Computer Organization

Cache Simulator Project Report

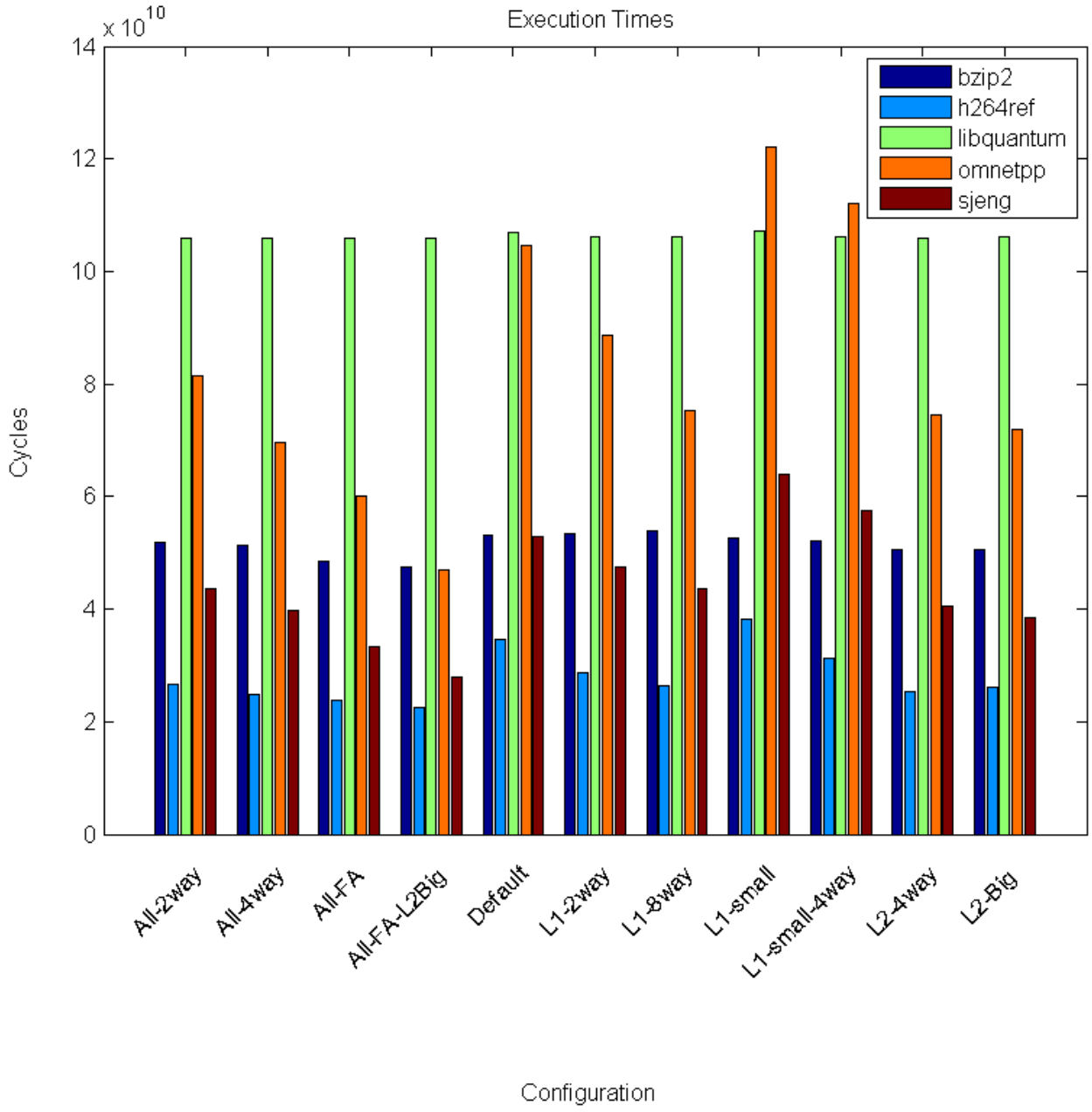
Parker Evans
Evan Schlomann
ECEN 4593

Introduction

The purpose of this project is to implement a simulator capable of simulating two levels of cache as well as a main memory. The simulator, while it does not actually run instructions given to it, attempts to give an estimate of the timing by giving hits and misses in the cache a set number of cycles. The amount of instructions, cycles, hits, misses, as well as other relevant data are collected for a variety of cache configurations and traces. The simulator is written in C as it has advantages in memory allocation and bit operations.

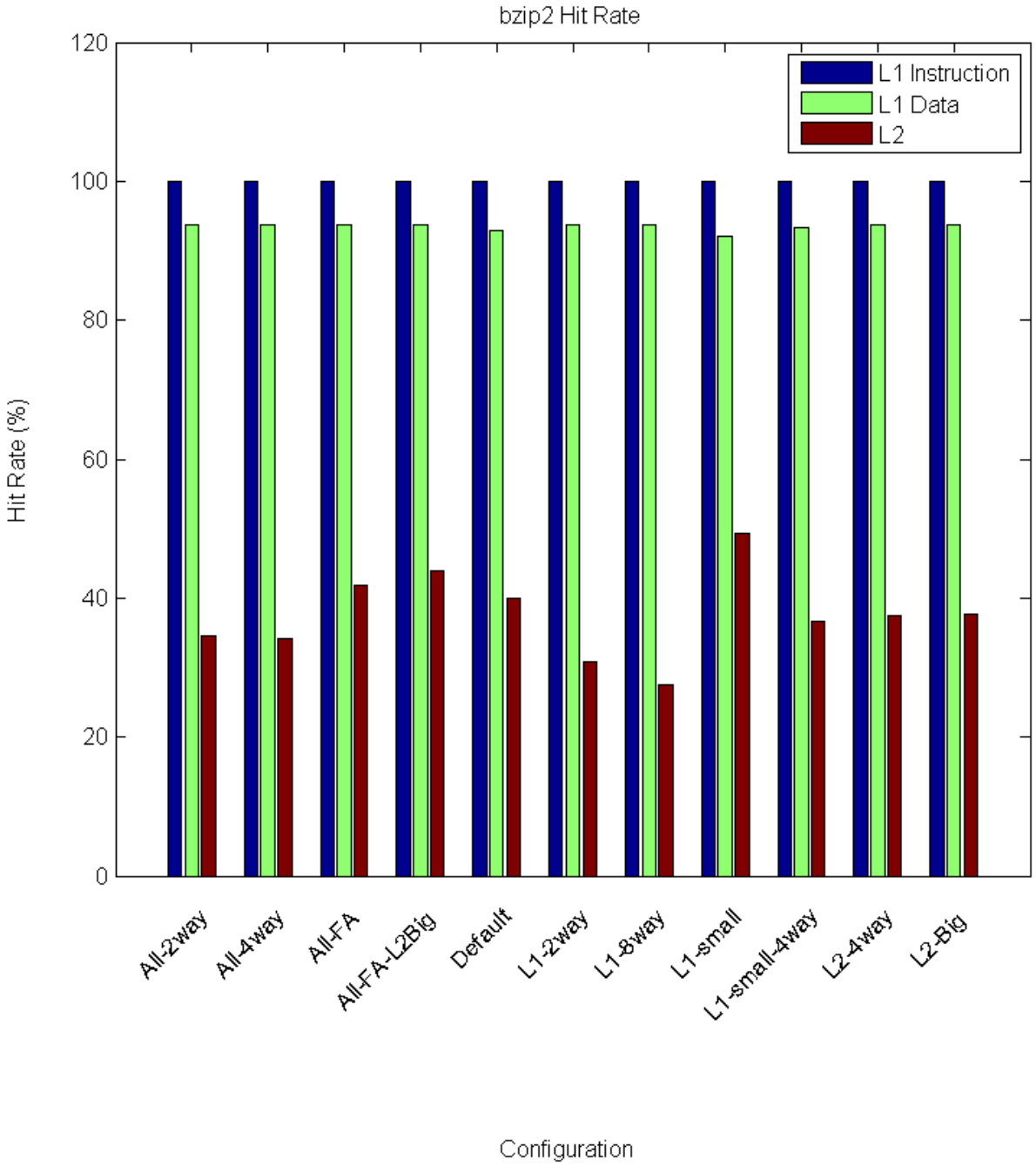
Results and Discussion

The first valuable metric is a comparison of the execution times between the different traces as well as the different configurations.

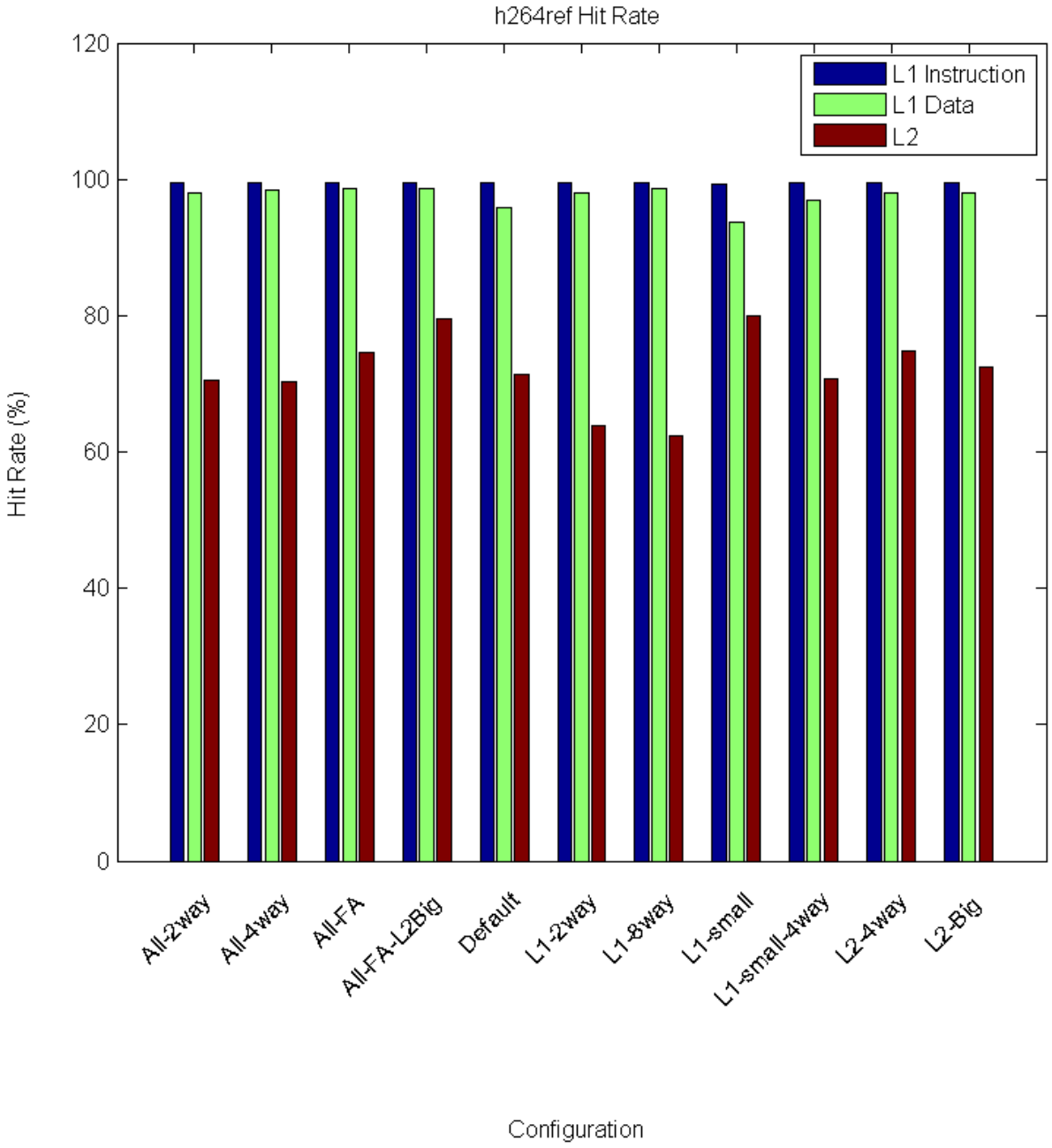


In this figure the columns of the bar chart are organized first by the configuration run, then the trace. Excepting the libquantum trace, the traces follow a regular pattern. The execution times go from smallest with the All-FA-L2Big configuration to the largest execution time L1-small. The amount of variation for different traces also differs greatly. Libquantum and bzip2 have a relatively small amount of variation between different configs, while omnetpp varies wildly. This behavior is likely due to the types of instructions contined within the trace. The omnetpp trace, for example, gains large benifits from increasing the associativity of both the L1 and L2 cache, as well as benefit from increasing the size of the L2 cache. The libquantum trace acts in a different fashion, not seeming to gain benefit from any sort of modifications made to the cache structure. Though libquatum was the smallest trace size at 4.2GB, it took the most cycles to run in 9 of the 11 configurations this is likely due to its hit rate which will be discussed later in this section.

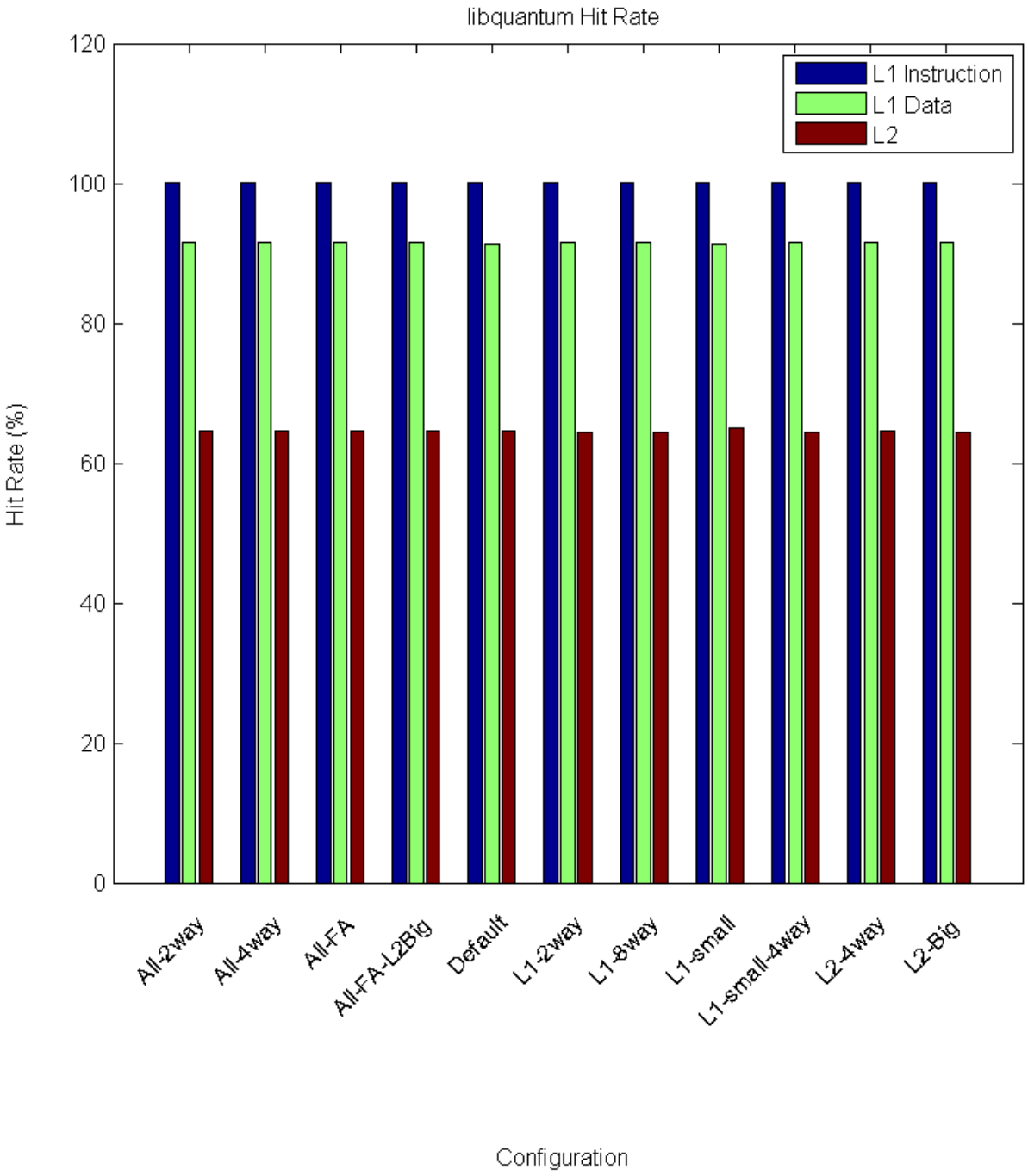
The next five images show comparisons of the hit rates for the L1 caches and the L2 cache for each of the different configurations with each of the five traces. This data sheds light on the results in the previous section.



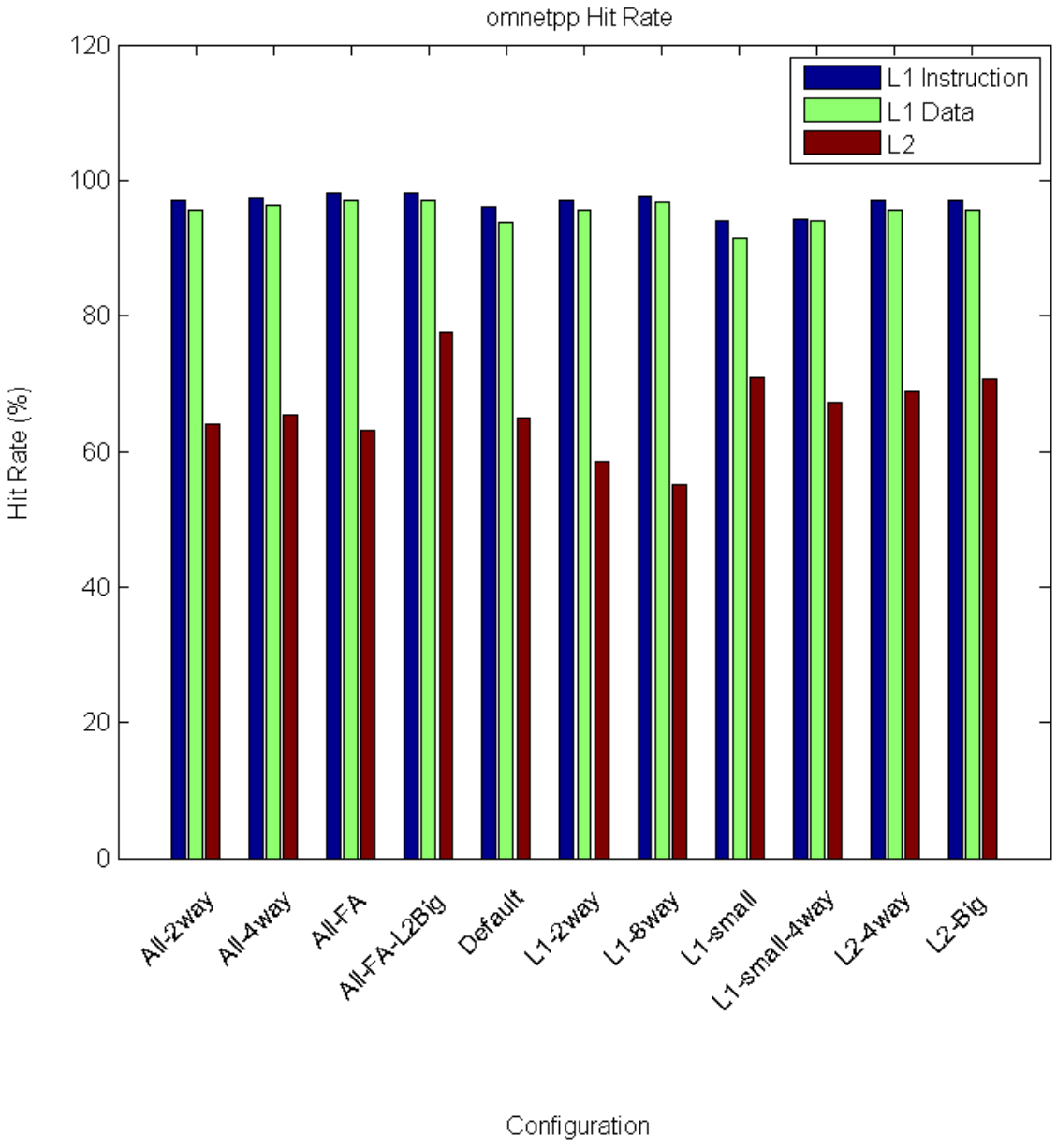
The hit rate results are different for the bzip2 trace as they have by far the lowest L2 cache hit rate at every configuration. The L1 instruction hit rate is near 100% while the L2 data hit rate hovers at around 93%. These relatively good hit rates keep the run time on the low side, but the under 50% L2 hit rate takes a toll on overall performance.



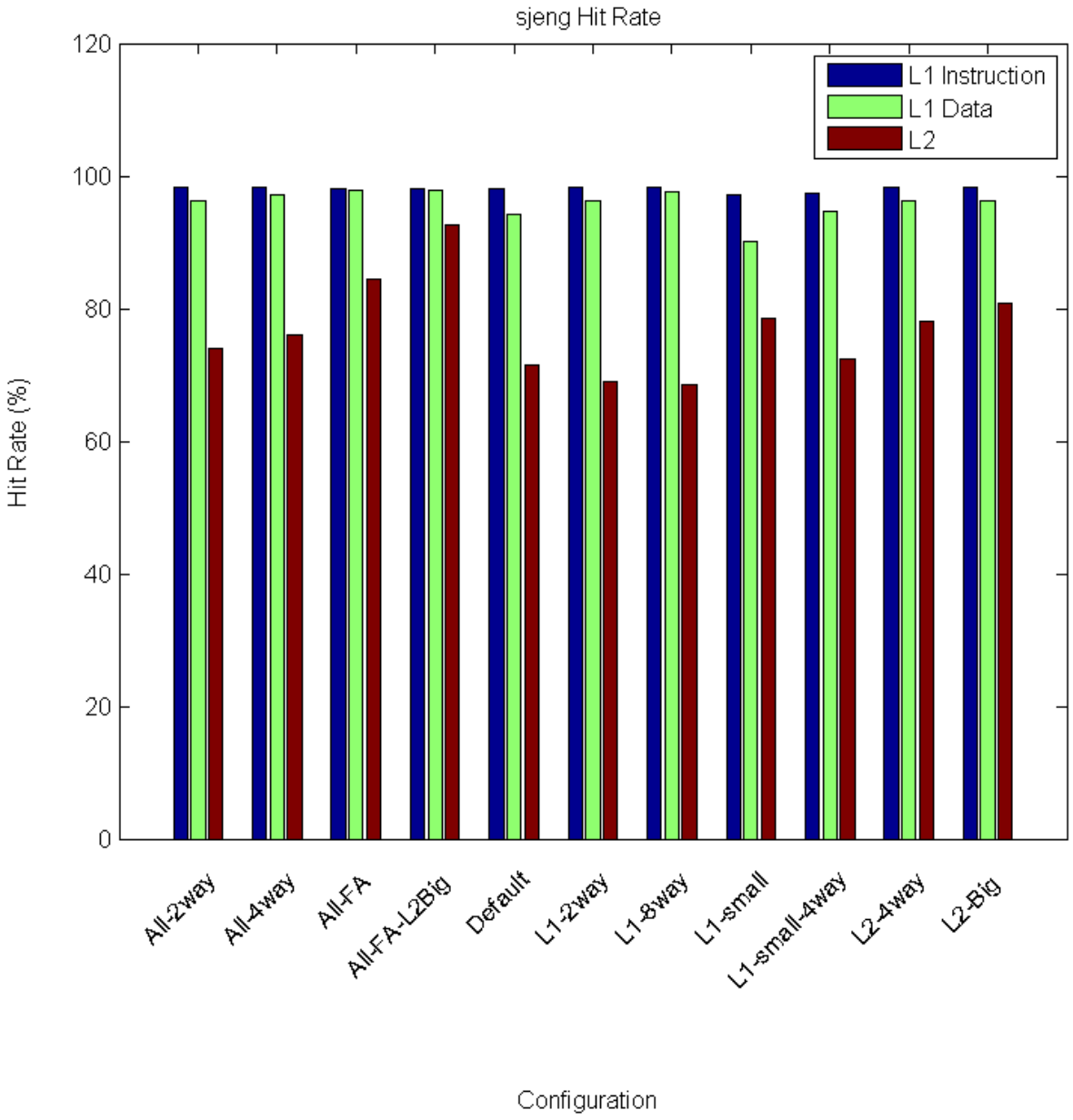
The h264ref trace had the best performace out of any trace at each of the different configurations. This makes some sense as it is the second smallest trace at 4.7GB, and therefore has fewer total references than the bigger traces. Only the libquantum traces is smaller, and it has other performance issues. The hit rate for the L1 instruction cache is near 100% for each of the configurations. The L1 data cache also consitently performes better than the rest of the traces. the h264ref trace additionally has a relatively good L2 hit rate whcih increases dramatically when the L2 cache is fully associative.



The libquantum trace had the most unusual characteristics. Though it was a small trace as discussed earlier, it took the most cycles to execute through the majority of the trials. The reason the exection time is so much longer has to do with the read instructions. Only 21.4% of the references were reads, but they took up 81.9% of the total execution time averaging 24.9 cycles per read instruction. Meanwhile, many of the other traces such as the sjeng trace have a smaller percentage of read instructions as well as a CPI of under 10. Another oddity with the libquantum trace is that for the different configurations, all of the exection times as well as the hit and miss rates were are near exactly the same. It is the only trace to exhibit this behavior. The trace that acts most similarly, bzip2, shares the characteristic of a low L2 cache hit rate, but libquantum's hit rate is not nearly as low.

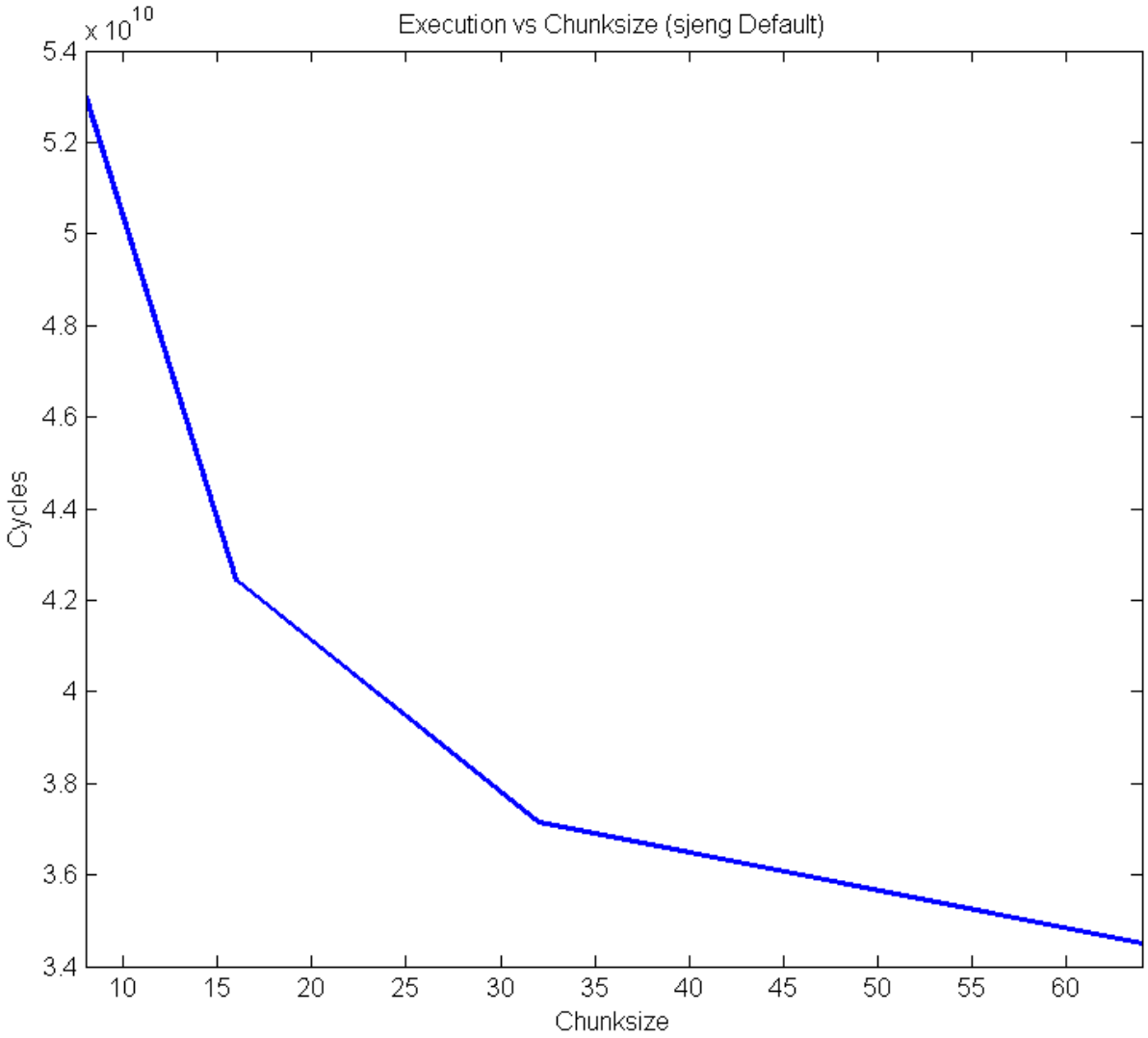


The omnetpp trace had the most variety in exection time. This is likely due to its lower than average L1i hit rate. Most traces are near 100% on the L1i hit rate, but the omnetpp hit rate is slightly lower, this means that more requests are going to L2 so a bigger or more associative L2 cache contributes greatly to the performance of the trace overall.

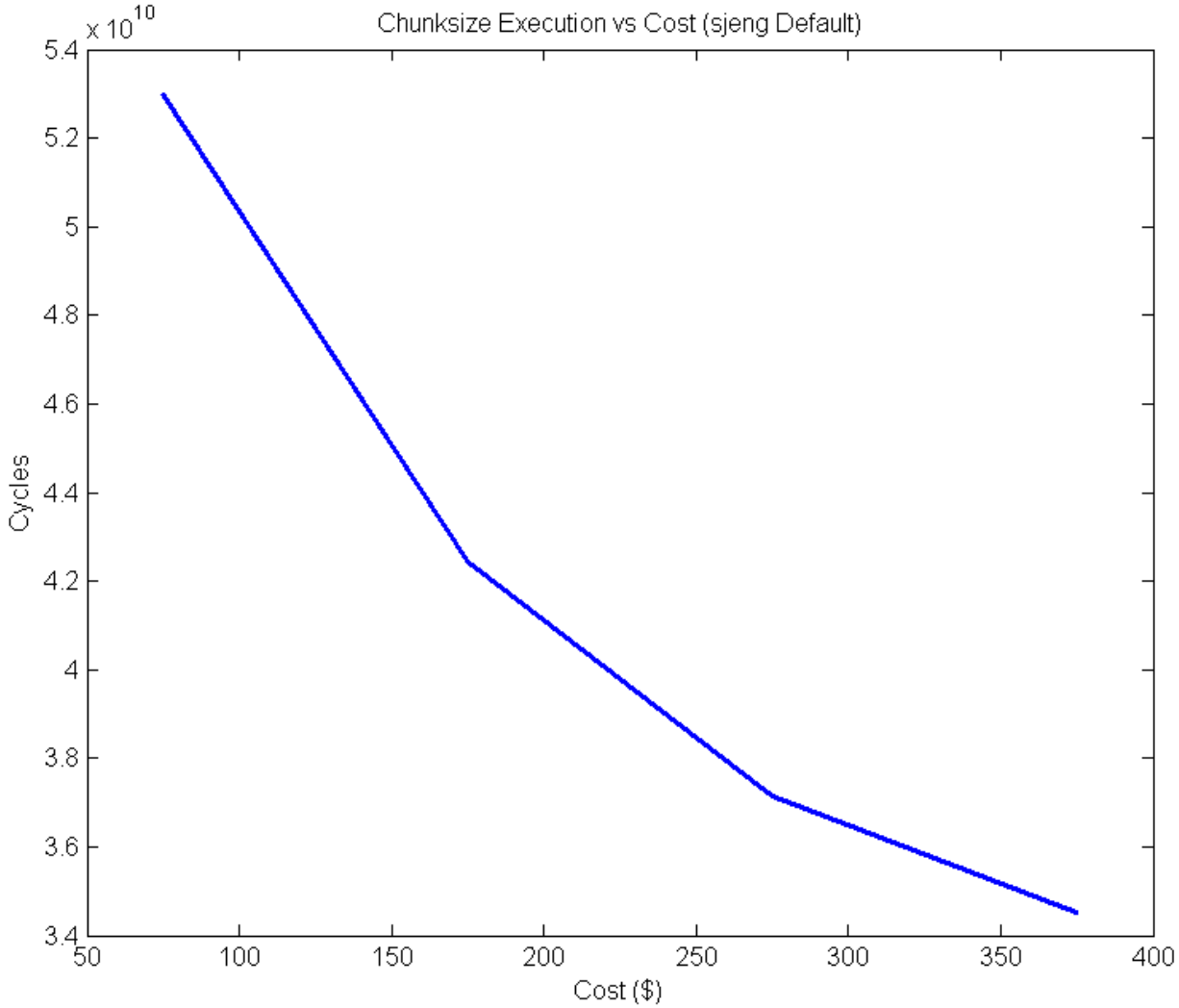


The sjeng trace has the highest average L2 hit rate of any of the traces in the ALL-FA-L2Big configuration at 92.6%, this is the only L2 hit rate that goes above 90% for any of the traces or configurations. This likely has to do with the relative consistency of the types of instructions. Even if the reference does not hit in the L1 caches, the odds are better for it hitting in the L2 if the references vary less.

The next two images deals again with the sjeng trace, but in this case the default configuration was used and the chunk size was modified. The chunk size comes into play during L2 misses. It defines the amount of information that can be carried from memory to the L2 cache at a time.

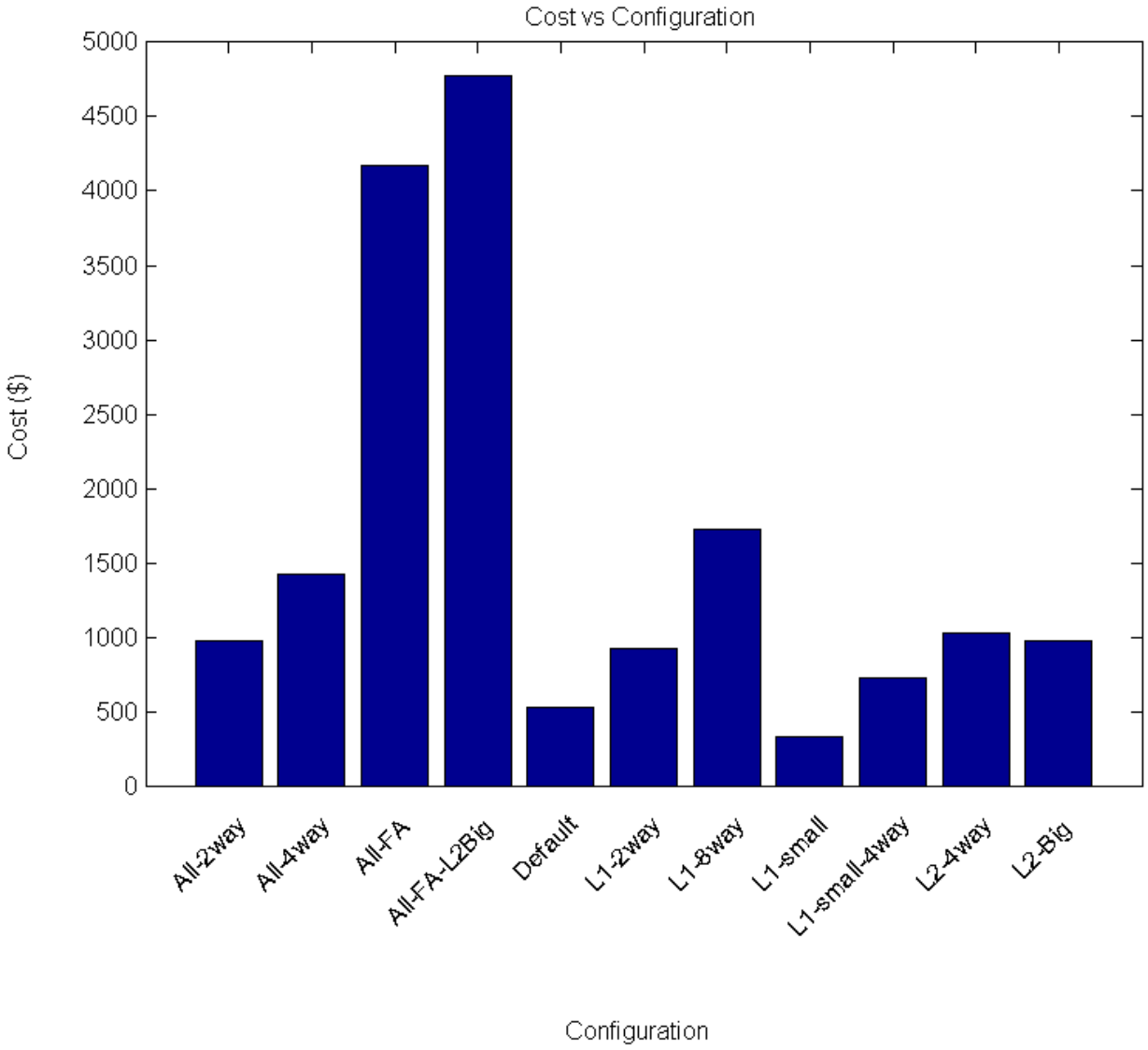


The default chunksize is 8 bytes. This takes around 53 billion cycles to perform. At the maximum chunk size, 64 bytes, the execution time goes down to 34.5 billion cycles. As the chunk size continues to increase, the effectiveness of each doubling goes down substatially. The increase from an 8 byte chunk size to a 16 byte chunk size is almost as large as the gap between a 16 byte chunk size and a 64 byte chunk size.

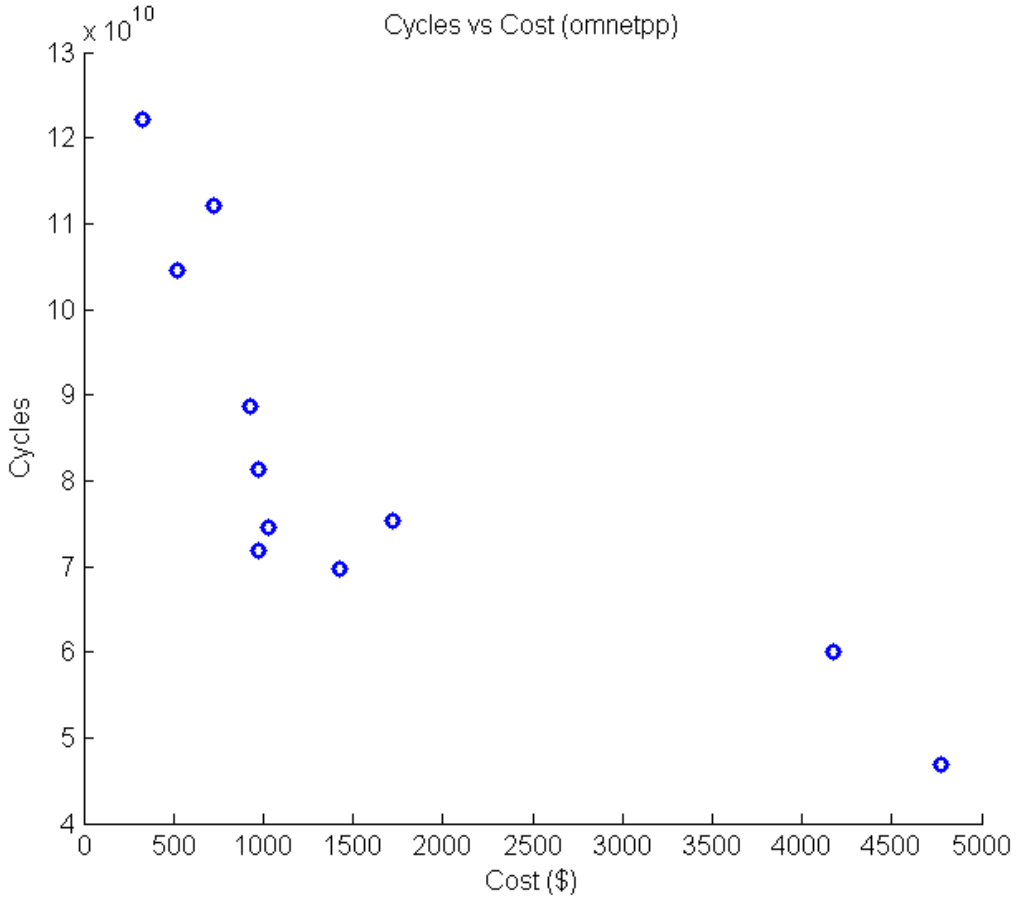


The next logical step is comparing the execution with the new chunksizes to the cost of the memory model. The cost graph is closer to linear than execution curve. A steeper slope on this graph is suggests a more efficieny in the system per dollar. Again the first doubling of the chunk size grants more utility than later doublings.

The final two images compare the costs of the different configurations and their speed.

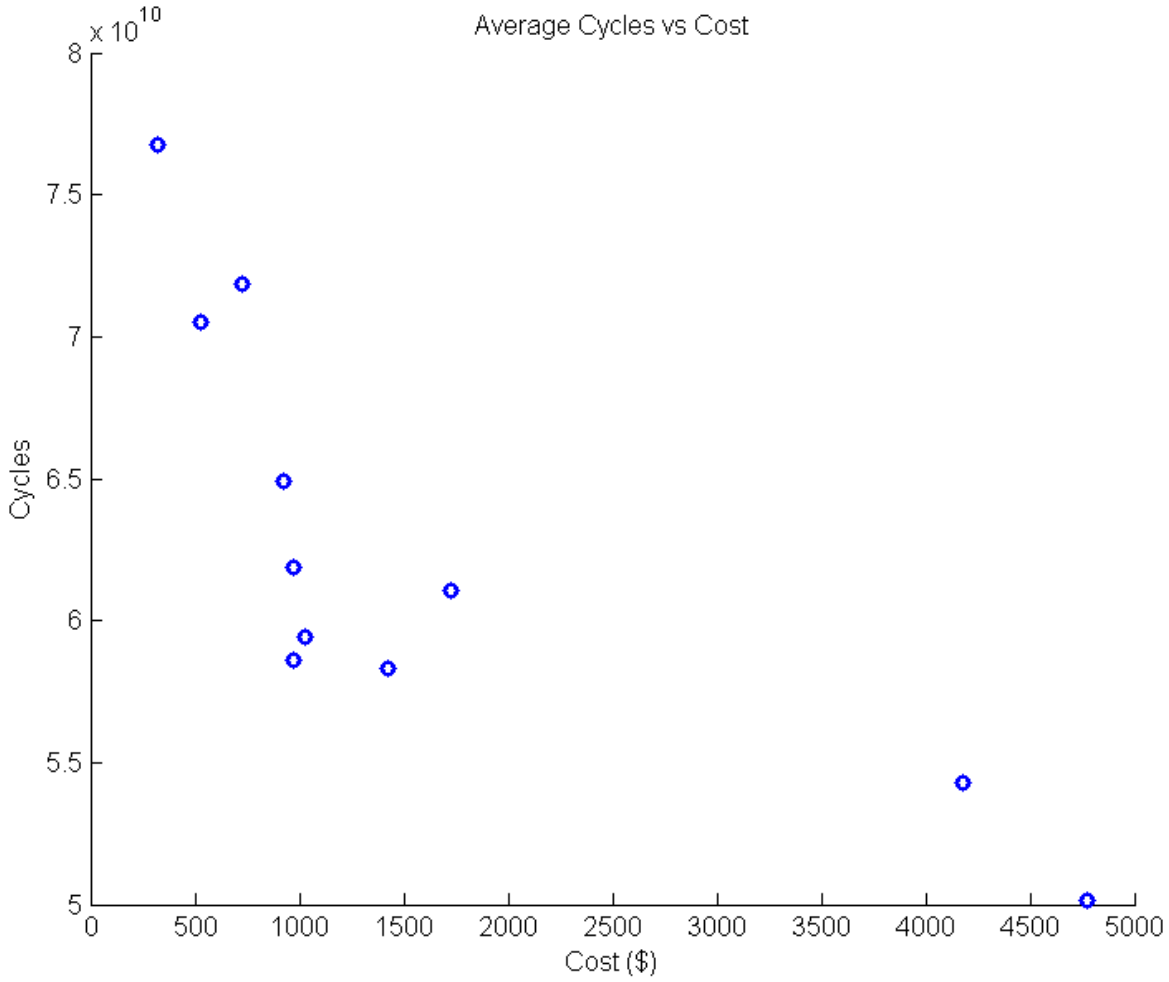


This figure clearly shows that making caches fully associative is incredibly expensive. Other modifications increase the price from default, but the expanding a system to fully associative comes with incredible cost.

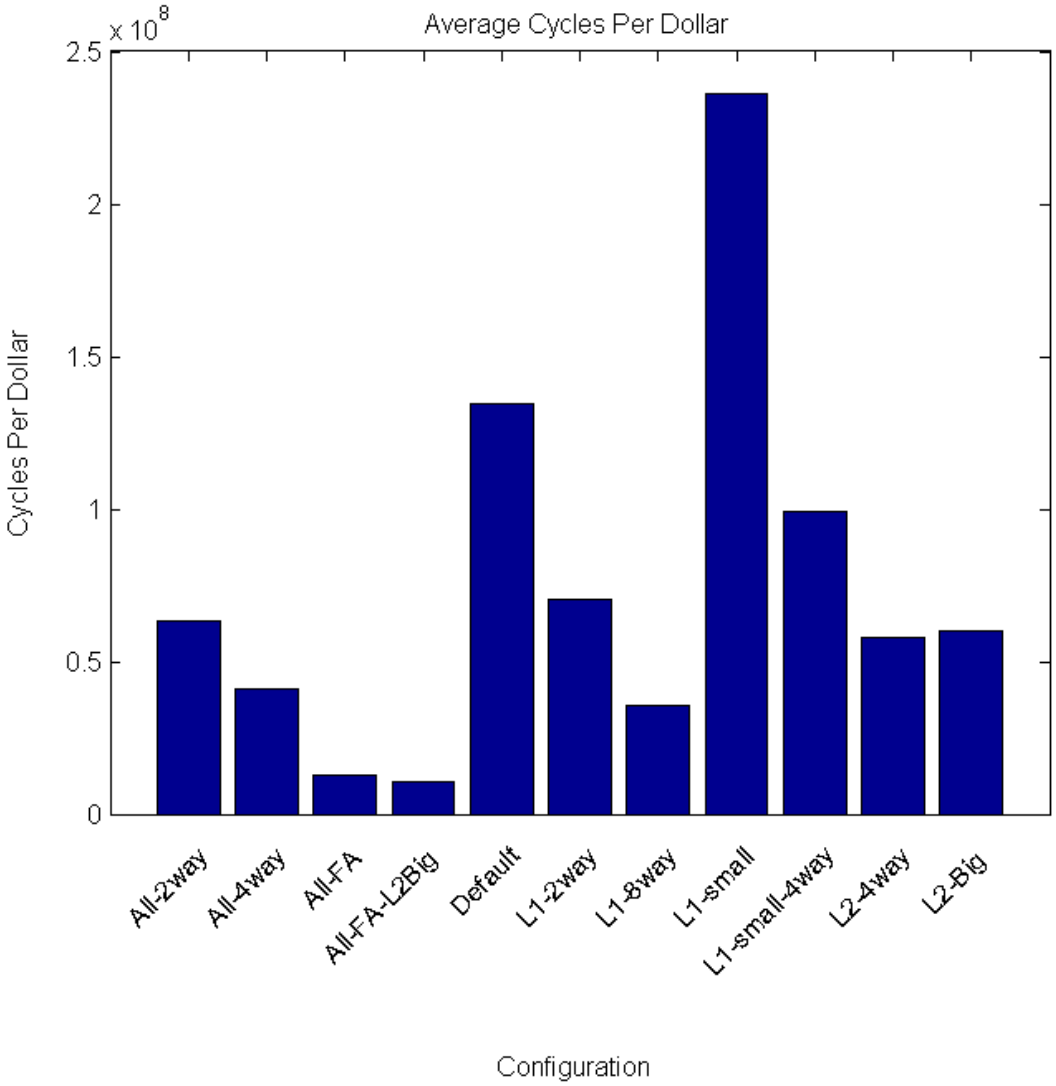


The plot in this image is a good place to start for choosing an optimal configuration. The trace used in this image is the omnetpp trace. It was chosen because it showed the greatest desparity in execution time for the different configurations, for this reason it highights the best choice in this scenario. The point with the lowest distance from the origin would be the most efficient implementation for the cost. In this case that point represents the L2-Big configuration.

The final two figures expand on the idea in the previous sectino in order to come to a conclusion about which configuration is more efficient on the whole. The total cycles are averaged for each configuration.



This figure shows a similar image to the scatter plot used for the omnetpp trace. This time the figure includes every trace. None the less it looks very similar. This is due to all of the traces following similar patterns in execution time for the different traces. Again the L2-Big configuration is closest to the origin, meaning that it is the most effieicnt configuration for the traces at hand.



This chart quantifies the optimization by plotting cycles vs cost. This plot is better for showing cost constraints. if the budget for the system is very very low, the L1-small trace does a good job of using each dollar very efficiently. The fully associtive traces on the other hand get very few cycles per dollar.

Conclusion

The original goal of making and testing a cache simulator was completed fully. The implimentation is clear and easy to follow, matches existing results up to 1 million traces, as well as provides meaningful and clear results. The simulations of the traces bzip2, h264ref, libquantum, omnetpp, and sjeng with the 11 different configurations provided extensive data to analyze the situation. Based on the concluding two graphs of the results section. The most effiecient configuration is L2-Big. It balances exection cycles and cost better than the remainder of the configurations.

Simulating a cache structure was a valuable experiment in gaining knowlege about how caches work as well as learning how their performance can change in different situations. Working in a language like C gave insights into how memory needs to be managed to guarantee correct execution when some configurations can take an hour or more to run. Appended at the end of this report are all of the results as well as all of the code used to write the simulator.

bzip2 with All-2way

```
---------------------------------------------------------------------
                         Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 2 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 51778560813; Total refs = 10000000073
Flush time = 391995579
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    20125896409   [38.9%]
  Writes =    19101145576   [36.9%]
  Inst.  =    12551518828   [24.2%]
  Total  =    51778560813

Average cycles per activity
  Read = 10.7; Write = 34.6; Inst. = 6.8
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527122  Miss Count = 555400
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2674;  Dirty Kickouts = 0;  Transfers = 555400
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2378127911  Miss Count = 160072485
  Total Requests = 2538200396
  Hit Rate = 93.7%  Miss Rate =  6.3%
  Kickouts = 154979866;  Dirty Kickouts = 62473522;  Transfers = 161308323
  Flush Kickouts = 1235838

Memory Level:  L2
  Hit Count = 77722140  Miss Count = 146615105
  Total Requests = 224337245
  Hit Rate = 34.6%  Miss Rate = 65.4%
  Kickouts = 136811720;  Dirty Kickouts = 55034623;  Transfers = 148667162
  Flush Kickouts = 2052057

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with All-4way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 4 : block size = 32
  Lcache size = 8192 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 51460524645; Total refs = 10000000073
Flush time = 385752378
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    19734975572   [38.3%]
  Writes =    19180835430   [37.3%]
  Inst.  =    12544713643   [24.4%]
  Total  =    51460524645

Average cycles per activity
  Read = 10.5; Write = 34.7; Inst. = 6.8
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527307  Miss Count = 555215
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 1778;  Dirty Kickouts = 0;  Transfers = 555215
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2380037622  Miss Count = 158162774
  Total Requests = 2538200396
  Hit Rate = 93.8%  Miss Rate =  6.2%
  Kickouts = 153066463;  Dirty Kickouts = 61597513;  Transfers = 159395041
  Flush Kickouts = 1232267

Memory Level:  L2
  Hit Count = 75870260  Miss Count = 145677509
  Total Requests = 221547769
  Hit Rate = 34.2%  Miss Rate = 65.8%
  Kickouts = 135799719;  Dirty Kickouts = 54270427;  Transfers = 147758102
  Flush Kickouts = 2080593

L1 cache cost (Icache $600) + (Dcache $600) = $1200
L2 cache cost = $150;  Memory cost = $75; Total cost = $1425
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with All-FA

```
---------------------------------------------------------------------
                         Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 32768 : ways = 512 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 48412620766; Total refs = 10000000073
Flush time = 356620034
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    18850904129   [38.9%]
  Writes =    17046226134   [35.2%]
  Inst.  =    12515490503   [25.9%]
  Total  =    48412620766

Average cycles per activity
  Read = 10.0; Write = 30.9; Inst. = 6.4
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527364  Miss Count = 555158
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 1619;  Dirty Kickouts = 0;  Transfers = 555158
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2380877774  Miss Count = 157322622
  Total Requests = 2538200396
  Hit Rate = 93.8%  Miss Rate =  6.2%
  Kickouts = 152225918;  Dirty Kickouts = 61293922;  Transfers = 158531019
  Flush Kickouts = 1208397

Memory Level:  L2
  Hit Count = 92347890  Miss Count = 128032209
  Total Requests = 220380099
  Hit Rate = 41.9%  Miss Rate = 58.1%
  Kickouts = 118091988;  Dirty Kickouts = 53783430;  Transfers = 130082402
  Flush Kickouts = 2050193

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $500;  Memory cost = $75; Total cost = $4175
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with All-FA-L2Big

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 65536 : ways = 1024 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 47430032780; Total refs = 10000000073
Flush time = 582675050
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    17998338985   [37.9%]
  Writes =    16690474845   [35.2%]
  Inst.  =    12741218950   [26.9%]
  Total  =    47430032780

Average cycles per activity
  Read = 9.6; Write = 30.2; Inst. = 6.3
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527364  Miss Count = 555158
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 1619;  Dirty Kickouts = 0;  Transfers = 555158
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2380877774  Miss Count = 157322622
  Total Requests = 2538200396
  Hit Rate = 93.8%  Miss Rate =  6.2%
  Kickouts = 152225918;  Dirty Kickouts = 61293922;  Transfers = 158531019
  Flush Kickouts = 1208397

Memory Level:  L2
  Hit Count = 97071248  Miss Count = 123308851
  Total Requests = 220380099
  Hit Rate = 44.0%  Miss Rate = 56.0%
  Kickouts = 104293882;  Dirty Kickouts = 51137949;  Transfers = 126793355
  Flush Kickouts = 3484504

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $1100;  Memory cost = $75; Total cost = $4775
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with Default

```
---------------------------------------------------------------------
                         Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 53174582036; Total refs = 10000000073
Flush time = 386727764
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    21895046645   [41.2%]
  Writes =    18718144578   [35.2%]
  Inst.  =    12561390813   [23.6%]
  Total  =    53174582036

Average cycles per activity
  Read = 11.6; Write = 33.9; Inst. = 7.0
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12094754632  Miss Count = 1327890
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 782782;  Dirty Kickouts = 0;  Transfers = 1327890
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2360100410  Miss Count = 178099986
  Total Requests = 2538200396
  Hit Rate = 93.0%  Miss Rate =  7.0%
  Kickouts = 173028492;  Dirty Kickouts = 69160663;  Transfers = 179312231
  Flush Kickouts = 1212245

Memory Level:  L2
  Hit Count = 99842778  Miss Count = 149958006
  Total Requests = 249800784
  Hit Rate = 40.0%  Miss Rate = 60.0%
  Kickouts = 140327599;  Dirty Kickouts = 57867454;  Transfers = 151908505
  Flush Kickouts = 1950499

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $75; Total cost = $525
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with L1-2way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 53379199640; Total refs = 10000000073
Flush time = 413636712
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    20850532975   [39.1%]
  Writes =    19954451746   [37.4%]
  Inst.  =    12574214919   [23.6%]
  Total  =    53379199640

Average cycles per activity
  Read = 11.1; Write = 36.1; Inst. = 7.1
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527122  Miss Count = 555400
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2674;  Dirty Kickouts = 0;  Transfers = 555400
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2378127911  Miss Count = 160072485
  Total Requests = 2538200396
  Hit Rate = 93.7%  Miss Rate =  6.3%
  Kickouts = 154979866;  Dirty Kickouts = 62473522;  Transfers = 161308323
  Flush Kickouts = 1235838

Memory Level:  L2
  Hit Count = 69099359  Miss Count = 155237886
  Total Requests = 224337245
  Hit Rate = 30.8%  Miss Rate = 69.2%
  Kickouts = 145607479;  Dirty Kickouts = 56092341;  Transfers = 157236190
  Flush Kickouts = 1998304

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $50;  Memory cost = $75; Total cost = $925
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with L1-8way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 8 : block size = 32
  Lcache size = 8192 : ways = 8 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 53980375392; Total refs = 10000000073
Flush time = 424333236
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    20481617396   [37.9%]
  Writes =    20913974915   [38.7%]
  Inst.  =    12584783081   [23.3%]
  Total  =    53980375392

Average cycles per activity
  Read = 10.9; Write = 37.9; Inst. = 7.1
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527357  Miss Count = 555165
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 1662;  Dirty Kickouts = 0;  Transfers = 555165
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2380527294  Miss Count = 157673102
  Total Requests = 2538200396
  Hit Rate = 93.8%  Miss Rate =  6.2%
  Kickouts = 152576404;  Dirty Kickouts = 61411716;  Transfers = 158893681
  Flush Kickouts = 1220579

Memory Level:  L2
  Hit Count = 60943291  Miss Count = 159917271
  Total Requests = 220860562
  Hit Rate = 27.6%  Miss Rate = 72.4%
  Kickouts = 150286864;  Dirty Kickouts = 55301590;  Transfers = 161920503
  Flush Kickouts = 2003232

L1 cache cost (Icache $800) + (Dcache $800) = $1600
L2 cache cost = $50;  Memory cost = $75; Total cost = $1725
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with L1-small

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 1 : block size = 32
  Lcache size = 4096 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 52591166997; Total refs = 10000000073
Flush time = 326624950
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    21895599242   [41.6%]
  Writes =    18192563490   [34.6%]
  Inst.  =    12503004265   [23.8%]
  Total  =    52591166997

Average cycles per activity
  Read = 11.6; Write = 32.9; Inst. = 7.0
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12094746130  Miss Count = 1336392
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 795934;  Dirty Kickouts = 0;  Transfers = 1336392
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2335018951  Miss Count = 203181445
  Total Requests = 2538200396
  Hit Rate = 92.0%  Miss Rate =  8.0%
  Kickouts = 200633399;  Dirty Kickouts = 78562930;  Transfers = 203824843
  Flush Kickouts = 643398

Memory Level:  L2
  Hit Count = 139961844  Miss Count = 143762321
  Total Requests = 283724165
  Hit Rate = 49.3%  Miss Rate = 50.7%
  Kickouts = 134131914;  Dirty Kickouts = 57422941;  Transfers = 145640606
  Flush Kickouts = 1878285

L1 cache cost (Icache $100) + (Dcache $100) = $200
L2 cache cost = $50;  Memory cost = $75; Total cost = $325
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with L1-small-4way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 4 : block size = 32
  Lcache size = 4096 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 52180556843; Total refs = 10000000073
Flush time = 342936137
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    20419940534   [39.1%]
  Writes =    19255854492   [36.9%]
  Inst.  =    12504761817   [24.0%]
  Total  =    52180556843

Average cycles per activity
  Read = 10.8; Write = 34.9; Inst. = 6.9
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095521668  Miss Count = 560854
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 14688;  Dirty Kickouts = 0;  Transfers = 560854
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2371554001  Miss Count = 166646395
  Total Requests = 2538200396
  Hit Rate = 93.4%  Miss Rate =  6.6%
  Kickouts = 164098043;  Dirty Kickouts = 65157952;  Transfers = 167302934
  Flush Kickouts = 656539

Memory Level:  L2
  Hit Count = 85318892  Miss Count = 147702848
  Total Requests = 233021740
  Hit Rate = 36.6%  Miss Rate = 63.4%
  Kickouts = 138072441;  Dirty Kickouts = 55700447;  Transfers = 149610932
  Flush Kickouts = 1908084

L1 cache cost (Icache $300) + (Dcache $300) = $600
L2 cache cost = $50;  Memory cost = $75; Total cost = $725
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with L2-4way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 50627255196; Total refs = 10000000073
Flush time = 377029930
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    19733727061   [39.0%]
  Writes =    18357491690   [36.3%]
  Inst.  =    12536036445   [24.8%]
  Total  =    50627255196

Average cycles per activity
  Read = 10.5; Write = 33.2; Inst. = 6.7
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527122  Miss Count = 555400
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2674;  Dirty Kickouts = 0;  Transfers = 555400
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2378127911  Miss Count = 160072485
  Total Requests = 2538200396
  Hit Rate = 93.7%  Miss Rate =  6.3%
  Kickouts = 154979866;  Dirty Kickouts = 62473522;  Transfers = 161308323
  Flush Kickouts = 1235838

Memory Level:  L2
  Hit Count = 84022371  Miss Count = 140314874
  Total Requests = 224337245
  Hit Rate = 37.5%  Miss Rate = 62.5%
  Kickouts = 130437084;  Dirty Kickouts = 54399063;  Transfers = 142382697
  Flush Kickouts = 2067823

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $150;  Memory cost = $75; Total cost = $1025
Flushes = 19908  :  Invalidates = 19908
```

bzip2 with L2-Big

```
---------------------------------------------------------------------
                         Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 65536 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 50524649832; Total refs = 10000000073
Flush time = 614934253
Inst refs = 7565217787; Data refs = 2434782286

Number of Reference Types: [Percentage]
  Reads  =     1882275327   [18.8%]
  Writes =      552506959   [ 5.5%]
  Inst.  =     7565217787   [75.7%]
  Total  =    10000000073

Total cycles for activities: [Percentage]
  Reads  =    19420267155   [38.4%]
  Writes =    18329566325   [36.3%]
  Inst.  =    12774816352   [25.3%]
  Total  =    50524649832

Average cycles per activity
  Read = 10.3; Write = 33.2; Inst. = 6.7
Ideal: Exec. Time = 17565217860; CPI = 2.3
Ideal mis-aligned: Exec. Time = 22199500705; CPI = 2.9

Memory Level:  L1i
  Hit Count = 12095527122  Miss Count = 555400
  Total Requests = 12096082522
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2674;  Dirty Kickouts = 0;  Transfers = 555400
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2378127911  Miss Count = 160072485
  Total Requests = 2538200396
  Hit Rate = 93.7%  Miss Rate =  6.3%
  Kickouts = 154979866;  Dirty Kickouts = 62473522;  Transfers = 161308323
  Flush Kickouts = 1235838

Memory Level:  L2
  Hit Count = 84346863  Miss Count = 139990382
  Total Requests = 224337245
  Hit Rate = 37.6%  Miss Rate = 62.4%
  Kickouts = 121764433;  Dirty Kickouts = 52710027;  Transfers = 143444646
  Flush Kickouts = 3454264

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 19908  :  Invalidates = 19908
```

h264ref with All-2way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 2 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 26651422424; Total refs = 10000000106
Flush time = 460681138
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     9273313717   [34.8%]
  Writes =     2620033212   [ 9.8%]
  Inst.  =    14758075495   [55.4%]
  Total  =    26651422424

Average cycles per activity
  Read = 3.4; Write = 4.5; Inst. = 4.0
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11160285560  Miss Count = 47791328
  Total Requests = 11208076888
  Hit Rate = 99.6%  Miss Rate =  0.4%
  Kickouts = 44719497;  Dirty Kickouts = 0;  Transfers = 47791328
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3940990857  Miss Count = 83744469
  Total Requests = 4024735326
  Hit Rate = 97.9%  Miss Rate =  2.1%
  Kickouts = 79223864;  Dirty Kickouts = 20672723;  Transfers = 85239755
  Flush Kickouts = 1495286

Memory Level:  L2
  Hit Count = 108531617  Miss Count = 45172189
  Total Requests = 153703806
  Hit Rate = 70.6%  Miss Rate = 29.4%
  Kickouts = 37062661;  Dirty Kickouts = 6420068;  Transfers = 47738132
  Flush Kickouts = 2565943

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 17710  :  Invalidates = 17710
```

h264ref with All-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 4 : block size = 32
  Lcache size = 8192 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 24871719204; Total refs = 10000000106
Flush time = 463933606
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     8151349554   [32.8%]
  Writes =     2452091937   [ 9.9%]
  Inst.  =    14268277713   [57.4%]
  Total  =    24871719204

Average cycles per activity
  Read = 3.0; Write = 4.2; Inst. = 3.7
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11159355696  Miss Count = 48721192
  Total Requests = 11208076888
  Hit Rate = 99.6%  Miss Rate =  0.4%
  Kickouts = 45599004;  Dirty Kickouts = 0;  Transfers = 48721192
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3962032699  Miss Count = 62702627
  Total Requests = 4024735326
  Hit Rate = 98.4%  Miss Rate =  1.6%
  Kickouts = 58175096;  Dirty Kickouts = 15514970;  Transfers = 64160586
  Flush Kickouts = 1457959

Memory Level:  L2
  Hit Count = 90182035  Miss Count = 38214713
  Total Requests = 128396748
  Hit Rate = 70.2%  Miss Rate = 29.8%
  Kickouts = 29865360;  Dirty Kickouts = 5001104;  Transfers = 40836577
  Flush Kickouts = 2621864

L1 cache cost (Icache $600) + (Dcache $600) = $1200
L2 cache cost = $150;  Memory cost = $75; Total cost = $1425
Flushes = 17710  :  Invalidates = 17710
```

h264ref with All-FA

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 32768 : ways = 512 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 23711427678; Total refs = 10000000106
Flush time = 461675702
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     7719132880   [32.6%]
  Writes =     2287924120   [ 9.6%]
  Inst.  =    13704370678   [57.8%]
  Total  =    23711427678

Average cycles per activity
  Read = 2.9; Write = 3.9; Inst. = 3.5
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11151277305  Miss Count = 56799583
  Total Requests = 11208076888
  Hit Rate = 99.5%  Miss Rate =  0.5%
  Kickouts = 53642596;  Dirty Kickouts = 0;  Transfers = 56799583
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3968992020  Miss Count = 55743306
  Total Requests = 4024735326
  Hit Rate = 98.6%  Miss Rate =  1.4%
  Kickouts = 51209295;  Dirty Kickouts = 13315680;  Transfers = 57201073
  Flush Kickouts = 1457767

Memory Level:  L2
  Hit Count = 95099351  Miss Count = 32216985
  Total Requests = 127316336
  Hit Rate = 74.7%  Miss Rate = 25.3%
  Kickouts = 23686675;  Dirty Kickouts = 4088172;  Transfers = 34854382
  Flush Kickouts = 2637397

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $500;  Memory cost = $75; Total cost = $4175
Flushes = 17710  :  Invalidates = 17710
```

h264ref with All-FA-L2Big

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 65536 : ways = 1024 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 22589842127; Total refs = 10000000106
Flush time = 634995681
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     6977778138   [30.9%]
  Writes =     2107720234   [ 9.3%]
  Inst.  =    13504343755   [59.8%]
  Total  =    22589842127

Average cycles per activity
  Read = 2.6; Write = 3.6; Inst. = 3.4
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11151277305  Miss Count = 56799583
  Total Requests = 11208076888
  Hit Rate = 99.5%  Miss Rate =  0.5%
  Kickouts = 53642596;  Dirty Kickouts = 0;  Transfers = 56799583
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3968992020  Miss Count = 55743306
  Total Requests = 4024735326
  Hit Rate = 98.6%  Miss Rate =  1.4%
  Kickouts = 51209295;  Dirty Kickouts = 13315680;  Transfers = 57201073
  Flush Kickouts = 1457767

Memory Level:  L2
  Hit Count = 101221104  Miss Count = 26095232
  Total Requests = 127316336
  Hit Rate = 79.5%  Miss Rate = 20.5%
  Kickouts = 11583346;  Dirty Kickouts = 2284380;  Transfers = 29916091
  Flush Kickouts = 3820859

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $1100;  Memory cost = $75; Total cost = $4775
Flushes = 17710  :  Invalidates = 17710
```

h264ref with Default

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 34732671255; Total refs = 10000000106
Flush time = 414875706
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =    14922428276   [43.0%]
  Writes =     3968789958   [11.4%]
  Inst.  =    15841453021   [45.6%]
  Total  =    34732671255

Average cycles per activity
  Read = 5.5; Write = 6.8; Inst. = 5.2
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11154941079  Miss Count = 53135809
  Total Requests = 11208076888
  Hit Rate = 99.5%  Miss Rate =  0.5%
  Kickouts = 50182353;  Dirty Kickouts = 0;  Transfers = 53135809
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3856443351  Miss Count = 168291975
  Total Requests = 4024735326
  Hit Rate = 95.8%  Miss Rate =  4.2%
  Kickouts = 163794819;  Dirty Kickouts = 43415363;  Transfers = 169848153
  Flush Kickouts = 1556178

Memory Level:  L2
  Hit Count = 190005299  Miss Count = 76394026
  Total Requests = 266399325
  Hit Rate = 71.3%  Miss Rate = 28.7%
  Kickouts = 68826710;  Dirty Kickouts = 13531568;  Transfers = 78641452
  Flush Kickouts = 2247426

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $75; Total cost = $525
Flushes = 17710  :  Invalidates = 17710
```

h264ref with L1-2way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 28743486090; Total refs = 10000000106
Flush time = 452031281
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =    10406599175   [36.2%]
  Writes =     3080548644   [10.7%]
  Inst.  =    15256338271   [53.1%]
  Total  =    28743486090

Average cycles per activity
  Read = 3.9; Write = 5.3; Inst. = 4.3
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11160285560  Miss Count = 47791328
  Total Requests = 11208076888
  Hit Rate = 99.6%  Miss Rate =  0.4%
  Kickouts = 44719497;  Dirty Kickouts = 0;  Transfers = 47791328
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3940990857  Miss Count = 83744469
  Total Requests = 4024735326
  Hit Rate = 97.9%  Miss Rate =  2.1%
  Kickouts = 79223864;  Dirty Kickouts = 20672723;  Transfers = 85239755
  Flush Kickouts = 1495286

Memory Level:  L2
  Hit Count = 98283939  Miss Count = 55419867
  Total Requests = 153703806
  Hit Rate = 63.9%  Miss Rate = 36.1%
  Kickouts = 47852551;  Dirty Kickouts = 9015338;  Transfers = 57769924
  Flush Kickouts = 2350057

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $50;  Memory cost = $75; Total cost = $925
Flushes = 17710  :  Invalidates = 17710
```

h264ref with L1-8way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 8 : block size = 32
  Lcache size = 8192 : ways = 8 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 26427381107; Total refs = 10000000106
Flush time = 471012673
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     8738448594   [33.1%]
  Writes =     2486986327   [ 9.4%]
  Inst.  =    15201946186   [57.5%]
  Total  =    26427381107

Average cycles per activity
  Read = 3.2; Write = 4.3; Inst. = 3.9
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11156202050  Miss Count = 51874838
  Total Requests = 11208076888
  Hit Rate = 99.5%  Miss Rate =  0.5%
  Kickouts = 48728868;  Dirty Kickouts = 0;  Transfers = 51874838
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3967117006  Miss Count = 57618320
  Total Requests = 4024735326
  Hit Rate = 98.6%  Miss Rate =  1.4%
  Kickouts = 53087007;  Dirty Kickouts = 13905536;  Transfers = 59054280
  Flush Kickouts = 1435960

Memory Level:  L2
  Hit Count = 77846779  Miss Count = 46987875
  Total Requests = 124834654
  Hit Rate = 62.4%  Miss Rate = 37.6%
  Kickouts = 39420559;  Dirty Kickouts = 6133891;  Transfers = 49388864
  Flush Kickouts = 2400989

L1 cache cost (Icache $800) + (Dcache $800) = $1600
L2 cache cost = $50;  Memory cost = $75; Total cost = $1725
Flushes = 17710  :  Invalidates = 17710
```

h264ref with L1-small

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 1 : block size = 32
  Lcache size = 4096 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 38210342618; Total refs = 10000000106
Flush time = 359939796
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =    16703383853   [43.7%]
  Writes =     4433241947   [11.6%]
  Inst.  =    17073716818   [44.7%]
  Total  =    38210342618

Average cycles per activity
  Read = 6.2; Write = 7.6; Inst. = 5.7
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11122010637  Miss Count = 86066251
  Total Requests = 11208076888
  Hit Rate = 99.2%  Miss Rate =  0.8%
  Kickouts = 84402319;  Dirty Kickouts = 0;  Transfers = 86066251
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3767873369  Miss Count = 256861957
  Total Requests = 4024735326
  Hit Rate = 93.6%  Miss Rate =  6.4%
  Kickouts = 254596737;  Dirty Kickouts = 65962984;  Transfers = 257514997
  Flush Kickouts = 653040

Memory Level:  L2
  Hit Count = 327114084  Miss Count = 82430148
  Total Requests = 409544232
  Hit Rate = 79.9%  Miss Rate = 20.1%
  Kickouts = 74862832;  Dirty Kickouts = 14915662;  Transfers = 84549511
  Flush Kickouts = 2119363

L1 cache cost (Icache $100) + (Dcache $100) = $200
L2 cache cost = $50;  Memory cost = $75; Total cost = $325
Flushes = 17710  :  Invalidates = 17710
```

h264ref with L1-small-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 4 : block size = 32
  Lcache size = 4096 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 31352624933; Total refs = 10000000106
Flush time = 369481158
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =    11583781668   [36.9%]
  Writes =     3274433575   [10.4%]
  Inst.  =    16494409690   [52.6%]
  Total  =    31352624933

Average cycles per activity
  Read = 4.3; Write = 5.6; Inst. = 4.7
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11142436555  Miss Count = 65640333
  Total Requests = 11208076888
  Hit Rate = 99.4%  Miss Rate =  0.6%
  Kickouts = 63953923;  Dirty Kickouts = 0;  Transfers = 65640333
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3902136074  Miss Count = 122599252
  Total Requests = 4024735326
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 120332251;  Dirty Kickouts = 28040657;  Transfers = 123209082
  Flush Kickouts = 609830

Memory Level:  L2
  Hit Count = 153586350  Miss Count = 63303722
  Total Requests = 216890072
  Hit Rate = 70.8%  Miss Rate = 29.2%
  Kickouts = 55736406;  Dirty Kickouts = 11028123;  Transfers = 65441227
  Flush Kickouts = 2137505

L1 cache cost (Icache $300) + (Dcache $300) = $600
L2 cache cost = $50;  Memory cost = $75; Total cost = $725
Flushes = 17710  :  Invalidates = 17710
```

h264ref with L2-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 25359907772; Total refs = 10000000106
Flush time = 459003409
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     8490793779   [33.5%]
  Writes =     2543128741   [10.0%]
  Inst.  =    14325985252   [56.5%]
  Total  =    25359907772

Average cycles per activity
  Read = 3.2; Write = 4.4; Inst. = 3.8
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11160285560  Miss Count = 47791328
  Total Requests = 11208076888
  Hit Rate = 99.6%  Miss Rate =  0.4%
  Kickouts = 44719497;  Dirty Kickouts = 0;  Transfers = 47791328
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3940990857  Miss Count = 83744469
  Total Requests = 4024735326
  Hit Rate = 97.9%  Miss Rate =  2.1%
  Kickouts = 79223864;  Dirty Kickouts = 20672723;  Transfers = 85239755
  Flush Kickouts = 1495286

Memory Level:  L2
  Hit Count = 115073093  Miss Count = 38630713
  Total Requests = 153703806
  Hit Rate = 74.9%  Miss Rate = 25.1%
  Kickouts = 30281360;  Dirty Kickouts = 5127123;  Transfers = 41245300
  Flush Kickouts = 2614587

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $150;  Memory cost = $75; Total cost = $1025
Flushes = 17710  :  Invalidates = 17710
```

h264ref with L2-Big

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 65536 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 26233475445; Total refs = 10000000106
Flush time = 597304377
Inst refs = 6730089151; Data refs = 3269910955

Number of Reference Types: [Percentage]
  Reads  =     2689845793   [26.9%]
  Writes =      580065162   [ 5.8%]
  Inst.  =     6730089151   [67.3%]
  Total  =    10000000106

Total cycles for activities: [Percentage]
  Reads  =     8949550344   [34.1%]
  Writes =     2778620242   [10.6%]
  Inst.  =    14505304859   [55.3%]
  Total  =    26233475445

Average cycles per activity
  Read = 3.3; Write = 4.8; Inst. = 3.9
Ideal: Exec. Time = 16730089257; CPI = 2.5
Ideal mis-aligned: Exec. Time = 21962901365; CPI = 3.3

Memory Level:  L1i
  Hit Count = 11160285560  Miss Count = 47791328
  Total Requests = 11208076888
  Hit Rate = 99.6%  Miss Rate =  0.4%
  Kickouts = 44719497;  Dirty Kickouts = 0;  Transfers = 47791328
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3940990857  Miss Count = 83744469
  Total Requests = 4024735326
  Hit Rate = 97.9%  Miss Rate =  2.1%
  Kickouts = 79223864;  Dirty Kickouts = 20672723;  Transfers = 85239755
  Flush Kickouts = 1495286

Memory Level:  L2
  Hit Count = 111378694  Miss Count = 42325112
  Total Requests = 153703806
  Hit Rate = 72.5%  Miss Rate = 27.5%
  Kickouts = 30301948;  Dirty Kickouts = 5996123;  Transfers = 45674468
  Flush Kickouts = 3349356

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 17710  :  Invalidates = 17710
```

libquantum with All-2way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 2 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 105993546379; Total refs = 16506492546
Flush time = 1267131606
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86764716341   [81.9%]
  Writes =     1197968976   [ 1.1%]
  Inst.  =    18030861062   [17.0%]
  Total  =   105993546379

Average cycles per activity
  Read = 24.6; Write = 2.4; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543262  Miss Count = 845791
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 4867;  Dirty Kickouts = 0;  Transfers = 845791
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262666441  Miss Count = 576950556
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568537628;  Dirty Kickouts = 234411130;  Transfers = 580300116
  Flush Kickouts = 3349560

Memory Level:  L2
  Hit Count = 526028656  Miss Count = 289528381
  Total Requests = 815557037
  Hit Rate = 64.5%  Miss Rate = 35.5%
  Kickouts = 272742873;  Dirty Kickouts = 125971714;  Transfers = 297120097
  Flush Kickouts = 7591716

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 32862  :  Invalidates = 32862
```

libquantum with All-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 4 : block size = 32
  Lcache size = 8192 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 105981944926; Total refs = 16506492546
Flush time = 1266900302
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86756713593   [81.9%]
  Writes =     1196009137   [ 1.1%]
  Inst.  =    18029222196   [17.0%]
  Total  =   105981944926

Average cycles per activity
  Read = 24.6; Write = 2.4; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543781  Miss Count = 845272
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2622;  Dirty Kickouts = 0;  Transfers = 845272
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262710581  Miss Count = 576906416
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568493488;  Dirty Kickouts = 234391405;  Transfers = 580255616
  Flush Kickouts = 3349200

Memory Level:  L2
  Hit Count = 526012854  Miss Count = 289479439
  Total Requests = 815492293
  Hit Rate = 64.5%  Miss Rate = 35.5%
  Kickouts = 272693097;  Dirty Kickouts = 125957694;  Transfers = 297070098
  Flush Kickouts = 7590659

L1 cache cost (Icache $600) + (Dcache $600) = $1200
L2 cache cost = $150;  Memory cost = $75; Total cost = $1425
Flushes = 32862  :  Invalidates = 32862
```

libquantum with All-FA

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 32768 : ways = 512 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 105980972137; Total refs = 16506492546
Flush time = 1305545541
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86716884320   [81.8%]
  Writes =     1196199205   [ 1.1%]
  Inst.  =    18067888612   [17.0%]
  Total  =   105980972137

Average cycles per activity
  Read = 24.6; Write = 2.4; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543917  Miss Count = 845136
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2003;  Dirty Kickouts = 0;  Transfers = 845136
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262710634  Miss Count = 576906363
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568493435;  Dirty Kickouts = 234391076;  Transfers = 580255703
  Flush Kickouts = 3349340

Memory Level:  L2
  Hit Count = 526017146  Miss Count = 289474769
  Total Requests = 815491915
  Hit Rate = 64.5%  Miss Rate = 35.5%
  Kickouts = 272684961;  Dirty Kickouts = 125714276;  Transfers = 297307677
  Flush Kickouts = 7832908

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $500;  Memory cost = $75; Total cost = $4175
Flushes = 32862  :  Invalidates = 32862
```

libquantum with All-FA-L2Big

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 65536 : ways = 1024 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 105971446887; Total refs = 16506492546
Flush time = 2443740677
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    85584741131   [80.8%]
  Writes =     1183802981   [ 1.1%]
  Inst.  =    19202902775   [18.1%]
  Total  =   105971446887

Average cycles per activity
  Read = 24.3; Write = 2.4; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543917  Miss Count = 845136
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2003;  Dirty Kickouts = 0;  Transfers = 845136
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262710634  Miss Count = 576906363
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568493435;  Dirty Kickouts = 234391076;  Transfers = 580255703
  Flush Kickouts = 3349340

Memory Level:  L2
  Hit Count = 526064776  Miss Count = 289427139
  Total Requests = 815491915
  Hit Rate = 64.5%  Miss Rate = 35.5%
  Kickouts = 256463250;  Dirty Kickouts = 118578619;  Transfers = 304385885
  Flush Kickouts = 14958746

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $1100;  Memory cost = $75; Total cost = $4775
Flushes = 32862  :  Invalidates = 32862
```

libquantum with Default

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 106995807090; Total refs = 16506492546
Flush time = 1253713938
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    87652194578   [81.9%]
  Writes =     1342072202   [ 1.3%]
  Inst.  =    18001540310   [16.8%]
  Total  =   106995807090

Average cycles per activity
  Read = 24.9; Write = 2.7; Inst. = 8.6
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620542015  Miss Count = 847038
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 8019;  Dirty Kickouts = 0;  Transfers = 847038
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6253479206  Miss Count = 586137791
  Total Requests = 6839616997
  Hit Rate = 91.4%  Miss Rate =  8.6%
  Kickouts = 577724863;  Dirty Kickouts = 236116182;  Transfers = 589493802
  Flush Kickouts = 3356011

Memory Level:  L2
  Hit Count = 533053879  Miss Count = 293403143
  Total Requests = 826457022
  Hit Rate = 64.5%  Miss Rate = 35.5%
  Kickouts = 276756312;  Dirty Kickouts = 127193832;  Transfers = 300913286
  Flush Kickouts = 7510143

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $75; Total cost = $525
Flushes = 32862  :  Invalidates = 32862
```

libquantum with L1-2way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 106138916031; Total refs = 16506492546
Flush time = 1271519267
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86876074725   [81.9%]
  Writes =     1243516735   [ 1.2%]
  Inst.  =    18019324571   [17.0%]
  Total  =   106138916031

Average cycles per activity
  Read = 24.6; Write = 2.5; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543262  Miss Count = 845791
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 4867;  Dirty Kickouts = 0;  Transfers = 845791
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262666441  Miss Count = 576950556
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568537628;  Dirty Kickouts = 234411130;  Transfers = 580300116
  Flush Kickouts = 3349560

Memory Level:  L2
  Hit Count = 525264100  Miss Count = 290292937
  Total Requests = 815557037
  Hit Rate = 64.4%  Miss Rate = 35.6%
  Kickouts = 273646106;  Dirty Kickouts = 126080806;  Transfers = 297886116
  Flush Kickouts = 7593179

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $50;  Memory cost = $75; Total cost = $925
Flushes = 32862  :  Invalidates = 32862
```

libquantum with L1-8way

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 8 : block size = 32
  Lcache size = 8192 : ways = 8 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 106119723825; Total refs = 16506492546
Flush time = 1272327657
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86878852789   [81.9%]
  Writes =     1220847119   [ 1.2%]
  Inst.  =    18020023917   [17.0%]
  Total  =   106119723825

Average cycles per activity
  Read = 24.6; Write = 2.5; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543922  Miss Count = 845131
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 2143;  Dirty Kickouts = 0;  Transfers = 845131
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262710638  Miss Count = 576906359
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568493431;  Dirty Kickouts = 234390955;  Transfers = 580255896
  Flush Kickouts = 3349537

Memory Level:  L2
  Hit Count = 525226657  Miss Count = 290265325
  Total Requests = 815491982
  Hit Rate = 64.4%  Miss Rate = 35.6%
  Kickouts = 273618494;  Dirty Kickouts = 125995847;  Transfers = 297858711
  Flush Kickouts = 7593386

L1 cache cost (Icache $800) + (Dcache $800) = $1600
L2 cache cost = $50;  Memory cost = $75; Total cost = $1725
Flushes = 32862  :  Invalidates = 32862
```

libquantum with L1-small

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 1 : block size = 32
  Lcache size = 4096 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 107058873826; Total refs = 16506492546
Flush time = 1226155310
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    87735756675   [82.0%]
  Writes =     1335196035   [ 1.2%]
  Inst.  =    17987921116   [16.8%]
  Total  =   107058873826

Average cycles per activity
  Read = 24.9; Write = 2.7; Inst. = 8.6
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620528006  Miss Count = 861047
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 69048;  Dirty Kickouts = 0;  Transfers = 861047
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6246272761  Miss Count = 593344236
  Total Requests = 6839616997
  Hit Rate = 91.3%  Miss Rate =  8.7%
  Kickouts = 589137772;  Dirty Kickouts = 238794548;  Transfers = 595009714
  Flush Kickouts = 1665478

Memory Level:  L2
  Hit Count = 541559895  Miss Count = 293105414
  Total Requests = 834665309
  Hit Rate = 64.9%  Miss Rate = 35.1%
  Kickouts = 276458583;  Dirty Kickouts = 127091825;  Transfers = 300607831
  Flush Kickouts = 7502417

L1 cache cost (Icache $100) + (Dcache $100) = $200
L2 cache cost = $50;  Memory cost = $75; Total cost = $325
Flushes = 32862  :  Invalidates = 32862
```

libquantum with L1-small-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 4 : block size = 32
  Lcache size = 4096 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 106064932697; Total refs = 16506492546
Flush time = 1243200044
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86829368659   [81.9%]
  Writes =     1232275937   [ 1.2%]
  Inst.  =    18003288101   [17.0%]
  Total  =   106064932697

Average cycles per activity
  Read = 24.6; Write = 2.5; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620542751  Miss Count = 846302
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 6062;  Dirty Kickouts = 0;  Transfers = 846302
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262691983  Miss Count = 576925014
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 572718550;  Dirty Kickouts = 236079319;  Transfers = 578587544
  Flush Kickouts = 1662530

Memory Level:  L2
  Hit Count = 525553677  Miss Count = 289959488
  Total Requests = 815513165
  Hit Rate = 64.4%  Miss Rate = 35.6%
  Kickouts = 273312657;  Dirty Kickouts = 125976043;  Transfers = 297547341
  Flush Kickouts = 7587853

L1 cache cost (Icache $300) + (Dcache $300) = $600
L2 cache cost = $50;  Memory cost = $75; Total cost = $725
Flushes = 32862  :  Invalidates = 32862
```

libquantum with L2-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 105983085852; Total refs = 16506492546
Flush time = 1266637692
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    86757451066   [81.9%]
  Writes =     1196597459   [ 1.1%]
  Inst.  =    18029037327   [17.0%]
  Total  =   105983085852

Average cycles per activity
  Read = 24.6; Write = 2.4; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543262  Miss Count = 845791
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 4867;  Dirty Kickouts = 0;  Transfers = 845791
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262666441  Miss Count = 576950556
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568537628;  Dirty Kickouts = 234411130;  Transfers = 580300116
  Flush Kickouts = 3349560

Memory Level:  L2
  Hit Count = 526078217  Miss Count = 289478820
  Total Requests = 815557037
  Hit Rate = 64.5%  Miss Rate = 35.5%
  Kickouts = 272692478;  Dirty Kickouts = 125959922;  Transfers = 297068679
  Flush Kickouts = 7589859

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $150;  Memory cost = $75; Total cost = $1025
Flushes = 32862  :  Invalidates = 32862
```

libquantum with L2-Big

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 65536 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 106065255517; Total refs = 16506492546
Flush time = 2409785835
Inst refs = 12487578510; Data refs = 4018914036

Number of Reference Types: [Percentage]
  Reads  =     3526260463   [21.4%]
  Writes =      492653573   [ 3.0%]
  Inst.  =    12487578510   [75.7%]
  Total  =    16506492546

Total cycles for activities: [Percentage]
  Reads  =    85686603467   [80.8%]
  Writes =     1217720961   [ 1.1%]
  Inst.  =    19160931089   [18.1%]
  Total  =   106065255517

Average cycles per activity
  Read = 24.3; Write = 2.5; Inst. = 8.5
Ideal: Exec. Time = 28994071056; CPI = 2.3
Ideal mis-aligned: Exec. Time = 35948584560; CPI = 2.9

Memory Level:  L1i
  Hit Count = 16620543262  Miss Count = 845791
  Total Requests = 16621389053
  Hit Rate = 100.0%  Miss Rate =  0.0%
  Kickouts = 4867;  Dirty Kickouts = 0;  Transfers = 845791
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 6262666441  Miss Count = 576950556
  Total Requests = 6839616997
  Hit Rate = 91.6%  Miss Rate =  8.4%
  Kickouts = 568537628;  Dirty Kickouts = 234411130;  Transfers = 580300116
  Flush Kickouts = 3349560

Memory Level:  L2
  Hit Count = 525604242  Miss Count = 289952795
  Total Requests = 815557037
  Hit Rate = 64.4%  Miss Rate = 35.6%
  Kickouts = 257143521;  Dirty Kickouts = 118855786;  Transfers = 304665639
  Flush Kickouts = 14712844

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 32862  :  Invalidates = 32862
```

omnetpp with All-2way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 2 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 81373730255; Total refs = 10000000076
Flush time = 655650593
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    36766983672   [45.2%]
  Writes =     8301078192   [10.2%]
  Inst.  =    36305668391   [44.6%]
  Total  =    81373730255

Average cycles per activity
  Read = 18.3; Write = 6.7; Inst. = 12.1
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11118333317  Miss Count = 341151756
  Total Requests = 11459485073
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 336631125;  Dirty Kickouts = 0;  Transfers = 341151756
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5675568402  Miss Count = 262045382
  Total Requests = 5937613784
  Hit Rate = 95.6%  Miss Rate =  4.4%
  Kickouts = 257498822;  Dirty Kickouts = 104790863;  Transfers = 264066286
  Flush Kickouts = 2020904

Memory Level:  L2
  Hit Count = 453912510  Miss Count = 256096395
  Total Requests = 710008905
  Hit Rate = 63.9%  Miss Rate = 36.1%
  Kickouts = 247005835;  Dirty Kickouts = 58840913;  Transfers = 259475499
  Flush Kickouts = 3379104

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with All-4way

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 4 : block size = 32
  Lcache size = 8192 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 69637819798; Total refs = 10000000076
Flush time = 646367100
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    31988061439   [45.9%]
  Writes =     7499970877   [10.8%]
  Inst.  =    30149787482   [43.3%]
  Total  =    69637819798

Average cycles per activity
  Read = 15.9; Write = 6.1; Inst. = 10.3
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11172241454  Miss Count = 287243619
  Total Requests = 11459485073
  Hit Rate = 97.5%  Miss Rate =  2.5%
  Kickouts = 282697412;  Dirty Kickouts = 0;  Transfers = 287243619
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5716738958  Miss Count = 220874826
  Total Requests = 5937613784
  Hit Rate = 96.3%  Miss Rate =  3.7%
  Kickouts = 216328266;  Dirty Kickouts = 84698427;  Transfers = 222939467
  Flush Kickouts = 2064641

Memory Level:  L2
  Hit Count = 388758250  Miss Count = 206123263
  Total Requests = 594881513
  Hit Rate = 65.4%  Miss Rate = 34.6%
  Kickouts = 197030163;  Dirty Kickouts = 49081448;  Transfers = 209459283
  Flush Kickouts = 3336020

L1 cache cost (Icache $600) + (Dcache $600) = $1200
L2 cache cost = $150;  Memory cost = $75; Total cost = $1425
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with All-FA

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 32768 : ways = 512 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 59984602959; Total refs = 10000000076
Flush time = 636853348
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    27380833221   [45.6%]
  Writes =     6955834764   [11.6%]
  Inst.  =    25647934974   [42.8%]
  Total  =    59984602959

Average cycles per activity
  Read = 13.6; Write = 5.6; Inst. = 8.9
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11255563323  Miss Count = 203921750
  Total Requests = 11459485073
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 199375365;  Dirty Kickouts = 0;  Transfers = 203921750
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5758964173  Miss Count = 178649611
  Total Requests = 5937613784
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 174103051;  Dirty Kickouts = 71366416;  Transfers = 180733918
  Flush Kickouts = 2084307

Memory Level:  L2
  Hit Count = 288308181  Miss Count = 167713903
  Total Requests = 456022084
  Hit Rate = 63.2%  Miss Rate = 36.8%
  Kickouts = 158620783;  Dirty Kickouts = 42647546;  Transfers = 171043730
  Flush Kickouts = 3329827

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $500;  Memory cost = $75; Total cost = $4175
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with All-FA-L2Big

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 65536 : ways = 1024 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 46942706986; Total refs = 10000000076
Flush time = 1089955143
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    21244616760   [45.3%]
  Writes =     5910128762   [12.6%]
  Inst.  =    19787961464   [42.2%]
  Total  =    46942706986

Average cycles per activity
  Read = 10.6; Write = 4.8; Inst. = 7.0
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11255563323  Miss Count = 203921750
  Total Requests = 11459485073
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 199375365;  Dirty Kickouts = 0;  Transfers = 203921750
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5758964173  Miss Count = 178649611
  Total Requests = 5937613784
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 174103051;  Dirty Kickouts = 71366416;  Transfers = 180733918
  Flush Kickouts = 2084307

Memory Level:  L2
  Hit Count = 354074600  Miss Count = 101947484
  Total Requests = 456022084
  Hit Rate = 77.6%  Miss Rate = 22.4%
  Kickouts = 83761244;  Dirty Kickouts = 26698185;  Transfers = 108358522
  Flush Kickouts = 6411038

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $1100;  Memory cost = $75; Total cost = $4775
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with Default

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 104538436058; Total refs = 10000000076
Flush time = 655735868
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    48347578206   [46.2%]
  Writes =    10805871311   [10.3%]
  Inst.  =    45384986541   [43.4%]
  Total  =   104538436058

Average cycles per activity
  Read = 24.0; Write = 8.7; Inst. = 15.5
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11001025643  Miss Count = 458459430
  Total Requests = 11459485073
  Hit Rate = 96.0%  Miss Rate =  4.0%
  Kickouts = 454141858;  Dirty Kickouts = 0;  Transfers = 458459430
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5568475800  Miss Count = 369137984
  Total Requests = 5937613784
  Hit Rate = 93.8%  Miss Rate =  6.2%
  Kickouts = 364591485;  Dirty Kickouts = 161396759;  Transfers = 371088233
  Flush Kickouts = 1950249

Memory Level:  L2
  Hit Count = 644430431  Miss Count = 346513991
  Total Requests = 990944422
  Hit Rate = 65.0%  Miss Rate = 35.0%
  Kickouts = 337480060;  Dirty Kickouts = 81547145;  Transfers = 349852702
  Flush Kickouts = 3338711

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $75; Total cost = $525
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with L1-2way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 88770499079; Total refs = 10000000076
Flush time = 695021224
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    40359651375   [45.5%]
  Writes =     9005916719   [10.1%]
  Inst.  =    39404930985   [44.4%]
  Total  =    88770499079

Average cycles per activity
  Read = 20.1; Write = 7.3; Inst. = 13.2
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11118333317  Miss Count = 341151756
  Total Requests = 11459485073
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 336631125;  Dirty Kickouts = 0;  Transfers = 341151756
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5675568402  Miss Count = 262045382
  Total Requests = 5937613784
  Hit Rate = 95.6%  Miss Rate =  4.4%
  Kickouts = 257498822;  Dirty Kickouts = 104790863;  Transfers = 264066286
  Flush Kickouts = 2020904

Memory Level:  L2
  Hit Count = 415208598  Miss Count = 294800307
  Total Requests = 710008905
  Hit Rate = 58.5%  Miss Rate = 41.5%
  Kickouts = 285766376;  Dirty Kickouts = 64669099;  Transfers = 298183822
  Flush Kickouts = 3383515

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $50;  Memory cost = $75; Total cost = $925
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with L1-8way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 8 : block size = 32
  Lcache size = 8192 : ways = 8 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 75258259293; Total refs = 10000000076
Flush time = 737455021
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    33335541562   [44.3%]
  Writes =     8004599600   [10.6%]
  Inst.  =    33918118131   [45.1%]
  Total  =    75258259293

Average cycles per activity
  Read = 16.6; Write = 6.5; Inst. = 11.2
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11196786230  Miss Count = 262698843
  Total Requests = 11459485073
  Hit Rate = 97.7%  Miss Rate =  2.3%
  Kickouts = 258152466;  Dirty Kickouts = 0;  Transfers = 262698843
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5740720184  Miss Count = 196893600
  Total Requests = 5937613784
  Hit Rate = 96.7%  Miss Rate =  3.3%
  Kickouts = 192347040;  Dirty Kickouts = 76592811;  Transfers = 198969396
  Flush Kickouts = 2075796

Memory Level:  L2
  Hit Count = 295793861  Miss Count = 242467189
  Total Requests = 538261050
  Hit Rate = 55.0%  Miss Rate = 45.0%
  Kickouts = 233433258;  Dirty Kickouts = 51782566;  Transfers = 245907321
  Flush Kickouts = 3440132

L1 cache cost (Icache $800) + (Dcache $800) = $1600
L2 cache cost = $50;  Memory cost = $75; Total cost = $1725
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with L1-small

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 1 : block size = 32
  Lcache size = 4096 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 122128687053; Total refs = 10000000076
Flush time = 502381764
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    54503971413   [44.6%]
  Writes =    11412021333   [ 9.3%]
  Inst.  =    56212694307   [46.0%]
  Total  =   122128687053

Average cycles per activity
  Read = 27.1; Write = 9.2; Inst. = 18.1
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 10787605967  Miss Count = 671879106
  Total Requests = 11459485073
  Hit Rate = 94.1%  Miss Rate =  5.9%
  Kickouts = 669606026;  Dirty Kickouts = 0;  Transfers = 671879106
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5431113100  Miss Count = 506500684
  Total Requests = 5937613784
  Hit Rate = 91.5%  Miss Rate =  8.5%
  Kickouts = 504227404;  Dirty Kickouts = 216081736;  Transfers = 507492305
  Flush Kickouts = 991621

Memory Level:  L2
  Hit Count = 989572804  Miss Count = 405880343
  Total Requests = 1395453147
  Hit Rate = 70.9%  Miss Rate = 29.1%
  Kickouts = 396846412;  Dirty Kickouts = 89897231;  Transfers = 408729324
  Flush Kickouts = 2848981

L1 cache cost (Icache $100) + (Dcache $100) = $200
L2 cache cost = $50;  Memory cost = $75; Total cost = $325
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with L1-small-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 4 : block size = 32
  Lcache size = 4096 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 112044583320; Total refs = 10000000076
Flush time = 522696693
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    47375946879   [42.3%]
  Writes =     9481535212   [ 8.5%]
  Inst.  =    55187101229   [49.3%]
  Total  =   112044583320

Average cycles per activity
  Read = 23.5; Write = 7.7; Inst. = 16.6
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 10797462979  Miss Count = 662022094
  Total Requests = 11459485073
  Hit Rate = 94.2%  Miss Rate =  5.8%
  Kickouts = 659748862;  Dirty Kickouts = 0;  Transfers = 662022094
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5577120317  Miss Count = 360493467
  Total Requests = 5937613784
  Hit Rate = 93.9%  Miss Rate =  6.1%
  Kickouts = 358220187;  Dirty Kickouts = 138937083;  Transfers = 361558962
  Flush Kickouts = 1065495

Memory Level:  L2
  Hit Count = 780952148  Miss Count = 381565991
  Total Requests = 1162518139
  Hit Rate = 67.2%  Miss Rate = 32.8%
  Kickouts = 372532060;  Dirty Kickouts = 75051307;  Transfers = 384425161
  Flush Kickouts = 2859170

L1 cache cost (Icache $300) + (Dcache $300) = $600
L2 cache cost = $50;  Memory cost = $75; Total cost = $725
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with L2-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 74513712911; Total refs = 10000000076
Flush time = 617973877
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    33604559809   [45.1%]
  Writes =     7889045508   [10.6%]
  Inst.  =    33020107594   [44.3%]
  Total  =    74513712911

Average cycles per activity
  Read = 16.7; Write = 6.4; Inst. = 11.0
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11118333317  Miss Count = 341151756
  Total Requests = 11459485073
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 336631125;  Dirty Kickouts = 0;  Transfers = 341151756
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5675568402  Miss Count = 262045382
  Total Requests = 5937613784
  Hit Rate = 95.6%  Miss Rate =  4.4%
  Kickouts = 257498822;  Dirty Kickouts = 104790863;  Transfers = 264066286
  Flush Kickouts = 2020904

Memory Level:  L2
  Hit Count = 488961342  Miss Count = 221047563
  Total Requests = 710008905
  Hit Rate = 68.9%  Miss Rate = 31.1%
  Kickouts = 211954463;  Dirty Kickouts = 52634419;  Transfers = 224340271
  Flush Kickouts = 3292708

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $150;  Memory cost = $75; Total cost = $1025
Flushes = 17759  :  Invalidates = 17759
```

omnetpp with L2-Big

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 65536 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 71873230048; Total refs = 10000000076
Flush time = 1040696919
Inst refs = 6748671723; Data refs = 3251328353

Number of Reference Types: [Percentage]
  Reads  =     2011922989   [20.1%]
  Writes =     1239405364   [12.4%]
  Inst.  =     6748671723   [67.5%]
  Total  =    10000000076

Total cycles for activities: [Percentage]
  Reads  =    32030079287   [44.6%]
  Writes =     7484735196   [10.4%]
  Inst.  =    32358415565   [45.0%]
  Total  =    71873230048

Average cycles per activity
  Read = 15.9; Write = 6.0; Inst. = 10.6
Ideal: Exec. Time = 16748671799; CPI = 2.5
Ideal mis-aligned: Exec. Time = 24145770580; CPI = 3.6

Memory Level:  L1i
  Hit Count = 11118333317  Miss Count = 341151756
  Total Requests = 11459485073
  Hit Rate = 97.0%  Miss Rate =  3.0%
  Kickouts = 336631125;  Dirty Kickouts = 0;  Transfers = 341151756
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 5675568402  Miss Count = 262045382
  Total Requests = 5937613784
  Hit Rate = 95.6%  Miss Rate =  4.4%
  Kickouts = 257498822;  Dirty Kickouts = 104790863;  Transfers = 264066286
  Flush Kickouts = 2020904

Memory Level:  L2
  Hit Count = 502326071  Miss Count = 207682834
  Total Requests = 710008905
  Hit Rate = 70.7%  Miss Rate = 29.3%
  Kickouts = 191072665;  Dirty Kickouts = 47529780;  Transfers = 213526599
  Flush Kickouts = 5843765

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 17759  :  Invalidates = 17759
```

sjeng with All-2way

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 2 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 43711871478; Total refs = 10000000109
Flush time = 589119361
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =    10477991568   [24.0%]
  Writes =     7023616473   [16.1%]
  Inst.  =    26210263437   [60.0%]
  Total  =    43711871478

Average cycles per activity
  Read = 5.5; Write = 9.7; Inst. = 5.9
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12329298534  Miss Count = 223077258
  Total Requests = 12552375792
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 218150206;  Dirty Kickouts = 0;  Transfers = 223077258
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3176318967  Miss Count = 121259542
  Total Requests = 3297578509
  Hit Rate = 96.3%  Miss Rate =  3.7%
  Kickouts = 116336783;  Dirty Kickouts = 53348235;  Transfers = 123513368
  Flush Kickouts = 2253826

Memory Level:  L2
  Hit Count = 296066515  Miss Count = 103872346
  Total Requests = 399938861
  Hit Rate = 74.0%  Miss Rate = 26.0%
  Kickouts = 94428259;  Dirty Kickouts = 23349036;  Transfers = 106597158
  Flush Kickouts = 2724812

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 19380  :  Invalidates = 19380
```

sjeng with All-4way

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 4 : block size = 32
  Lcache size = 8192 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 39766167947; Total refs = 10000000109
Flush time = 614964786
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     9539679685   [24.0%]
  Writes =     6034447296   [15.2%]
  Inst.  =    24192040966   [60.8%]
  Total  =    39766167947

Average cycles per activity
  Read = 5.0; Write = 8.3; Inst. = 5.4
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12332868791  Miss Count = 219507001
  Total Requests = 12552375792
  Hit Rate = 98.3%  Miss Rate =  1.7%
  Kickouts = 214571771;  Dirty Kickouts = 0;  Transfers = 219507001
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3200939266  Miss Count = 96639243
  Total Requests = 3297578509
  Hit Rate = 97.1%  Miss Rate =  2.9%
  Kickouts = 91680120;  Dirty Kickouts = 43578332;  Transfers = 99075132
  Flush Kickouts = 2435889

Memory Level:  L2
  Hit Count = 275257044  Miss Count = 86903421
  Total Requests = 362160465
  Hit Rate = 76.0%  Miss Rate = 24.0%
  Kickouts = 77136290;  Dirty Kickouts = 20032790;  Transfers = 89713063
  Flush Kickouts = 2809642

L1 cache cost (Icache $600) + (Dcache $600) = $1200
L2 cache cost = $150;  Memory cost = $75; Total cost = $1425
Flushes = 19380  :  Invalidates = 19380
```

sjeng with All-FA

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 32768 : ways = 512 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 33410718797; Total refs = 10000000109
Flush time = 587157778
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     7141934344   [21.4%]
  Writes =     5112140553   [15.3%]
  Inst.  =    21156643900   [63.3%]
  Total  =    33410718797

Average cycles per activity
  Read = 3.7; Write = 7.0; Inst. = 4.5
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12304531666  Miss Count = 247844126
  Total Requests = 12552375792
  Hit Rate = 98.0%  Miss Rate =  2.0%
  Kickouts = 242908776;  Dirty Kickouts = 0;  Transfers = 247844126
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3228649349  Miss Count = 68929160
  Total Requests = 3297578509
  Hit Rate = 97.9%  Miss Rate =  2.1%
  Kickouts = 63967624;  Dirty Kickouts = 35733591;  Transfers = 71351092
  Flush Kickouts = 2421932

Memory Level:  L2
  Hit Count = 299595204  Miss Count = 55333605
  Total Requests = 354928809
  Hit Rate = 84.4%  Miss Rate = 15.6%
  Kickouts = 45410892;  Dirty Kickouts = 14062768;  Transfers = 58016756
  Flush Kickouts = 2683151

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $500;  Memory cost = $75; Total cost = $4175
Flushes = 19380  :  Invalidates = 19380
```

sjeng with All-FA-L2Big

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 256 : block size = 32
  Lcache size = 8192 : ways = 256 : block size = 32
  L2-cache size = 65536 : ways = 1024 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 27897621005; Total refs = 10000000109
Flush time = 1038238739
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     4596772368   [16.5%]
  Writes =     4473806936   [16.0%]
  Inst.  =    18827041701   [67.5%]
  Total  =    27897621005

Average cycles per activity
  Read = 2.4; Write = 6.1; Inst. = 3.8
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12304531666  Miss Count = 247844126
  Total Requests = 12552375792
  Hit Rate = 98.0%  Miss Rate =  2.0%
  Kickouts = 242908776;  Dirty Kickouts = 0;  Transfers = 247844126
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3228649349  Miss Count = 68929160
  Total Requests = 3297578509
  Hit Rate = 97.9%  Miss Rate =  2.1%
  Kickouts = 63967624;  Dirty Kickouts = 35733591;  Transfers = 71351092
  Flush Kickouts = 2421932

Memory Level:  L2
  Hit Count = 328804740  Miss Count = 26124069
  Total Requests = 354928809
  Hit Rate = 92.6%  Miss Rate =  7.4%
  Kickouts = 7153607;  Dirty Kickouts = 6689001;  Transfers = 32211579
  Flush Kickouts = 6087510

L1 cache cost (Icache $1800) + (Dcache $1800) = $3600
L2 cache cost = $1100;  Memory cost = $75; Total cost = $4775
Flushes = 19380  :  Invalidates = 19380
```

sjeng with Default

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 52995995056; Total refs = 10000000109
Flush time = 578294593
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =    15529456925   [29.3%]
  Writes =     7799107927   [14.7%]
  Inst.  =    29667430204   [56.0%]
  Total  =    52995995056

Average cycles per activity
  Read = 8.1; Write = 10.7; Inst. = 7.2
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12315460746  Miss Count = 236915046
  Total Requests = 12552375792
  Hit Rate = 98.1%  Miss Rate =  1.9%
  Kickouts = 232095466;  Dirty Kickouts = 0;  Transfers = 236915046
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3108702506  Miss Count = 188876003
  Total Requests = 3297578509
  Hit Rate = 94.3%  Miss Rate =  5.7%
  Kickouts = 184181921;  Dirty Kickouts = 74706271;  Transfers = 190939943
  Flush Kickouts = 2063940

Memory Level:  L2
  Hit Count = 358958322  Miss Count = 143602938
  Total Requests = 502561260
  Hit Rate = 71.4%  Miss Rate = 28.6%
  Kickouts = 134852808;  Dirty Kickouts = 29789793;  Transfers = 146314021
  Flush Kickouts = 2711083

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $75; Total cost = $525
Flushes = 19380  :  Invalidates = 19380
```

sjeng with L1-2way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 47516369329; Total refs = 10000000109
Flush time = 610665984
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =    12212210317   [25.7%]
  Writes =     7301113771   [15.4%]
  Inst.  =    28003045241   [58.9%]
  Total  =    47516369329

Average cycles per activity
  Read = 6.4; Write = 10.0; Inst. = 6.5
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12329298534  Miss Count = 223077258
  Total Requests = 12552375792
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 218150206;  Dirty Kickouts = 0;  Transfers = 223077258
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3176318967  Miss Count = 121259542
  Total Requests = 3297578509
  Hit Rate = 96.3%  Miss Rate =  3.7%
  Kickouts = 116336783;  Dirty Kickouts = 53348235;  Transfers = 123513368
  Flush Kickouts = 2253826

Memory Level:  L2
  Hit Count = 275423782  Miss Count = 124515079
  Total Requests = 399938861
  Hit Rate = 68.9%  Miss Rate = 31.1%
  Kickouts = 115764949;  Dirty Kickouts = 25558089;  Transfers = 127263097
  Flush Kickouts = 2748018

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $50;  Memory cost = $75; Total cost = $925
Flushes = 19380  :  Invalidates = 19380
```

sjeng with L1-8way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 8 : block size = 32
  Lcache size = 8192 : ways = 8 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 43568384419; Total refs = 10000000109
Flush time = 678907579
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     9991303499   [22.9%]
  Writes =     6342048091   [14.6%]
  Inst.  =    27235032829   [62.5%]
  Total  =    43568384419

Average cycles per activity
  Read = 5.2; Write = 8.7; Inst. = 5.9
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12325674933  Miss Count = 226700859
  Total Requests = 12552375792
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 221765536;  Dirty Kickouts = 0;  Transfers = 226700859
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3215035005  Miss Count = 82543504
  Total Requests = 3297578509
  Hit Rate = 97.5%  Miss Rate =  2.5%
  Kickouts = 77582058;  Dirty Kickouts = 38144847;  Transfers = 85027259
  Flush Kickouts = 2483755

Memory Level:  L2
  Hit Count = 239715285  Miss Count = 110157680
  Total Requests = 349872965
  Hit Rate = 68.5%  Miss Rate = 31.5%
  Kickouts = 101407550;  Dirty Kickouts = 20840272;  Transfers = 112847150
  Flush Kickouts = 2689470

L1 cache cost (Icache $800) + (Dcache $800) = $1600
L2 cache cost = $50;  Memory cost = $75; Total cost = $1725
Flushes = 19380  :  Invalidates = 19380
```

sjeng with L1-small

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 1 : block size = 32
  Lcache size = 4096 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 63959998711; Total refs = 10000000109
Flush time = 417902919
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =    20529752551   [32.1%]
  Writes =     8411002895   [13.2%]
  Inst.  =    35019243265   [54.8%]
  Total  =    63959998711

Average cycles per activity
  Read = 10.8; Write = 11.6; Inst. = 8.7
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12184655652  Miss Count = 367720140
  Total Requests = 12552375792
  Hit Rate = 97.1%  Miss Rate =  2.9%
  Kickouts = 365252522;  Dirty Kickouts = 0;  Transfers = 367720140
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 2973403604  Miss Count = 324174905
  Total Requests = 3297578509
  Hit Rate = 90.2%  Miss Rate =  9.8%
  Kickouts = 321698344;  Dirty Kickouts = 121570257;  Transfers = 325178148
  Flush Kickouts = 1003243

Memory Level:  L2
  Hit Count = 639175255  Miss Count = 175293290
  Total Requests = 814468545
  Hit Rate = 78.5%  Miss Rate = 21.5%
  Kickouts = 166543160;  Dirty Kickouts = 34738579;  Transfers = 177599347
  Flush Kickouts = 2306057

L1 cache cost (Icache $100) + (Dcache $100) = $200
L2 cache cost = $50;  Memory cost = $75; Total cost = $325
Flushes = 19380  :  Invalidates = 19380
```

sjeng with L1-small-4way

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 4096 : ways = 4 : block size = 32
  Lcache size = 4096 : ways = 4 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 57618128067; Total refs = 10000000109
Flush time = 435046891
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =    16588367269   [28.8%]
  Writes =     7373846198   [12.8%]
  Inst.  =    33655914600   [58.4%]
  Total  =    57618128067

Average cycles per activity
  Read = 8.7; Write = 10.1; Inst. = 7.8
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12210515355  Miss Count = 341860437
  Total Requests = 12552375792
  Hit Rate = 97.3%  Miss Rate =  2.7%
  Kickouts = 339392543;  Dirty Kickouts = 0;  Transfers = 341860437
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3121957304  Miss Count = 175621205
  Total Requests = 3297578509
  Hit Rate = 94.7%  Miss Rate =  5.3%
  Kickouts = 173140439;  Dirty Kickouts = 69680495;  Transfers = 176771792
  Flush Kickouts = 1150587

Memory Level:  L2
  Hit Count = 425937732  Miss Count = 162374992
  Total Requests = 588312724
  Hit Rate = 72.4%  Miss Rate = 27.6%
  Kickouts = 153624862;  Dirty Kickouts = 30855400;  Transfers = 164703202
  Flush Kickouts = 2328210

L1 cache cost (Icache $300) + (Dcache $300) = $600
L2 cache cost = $50;  Memory cost = $75; Total cost = $725
Flushes = 19380  :  Invalidates = 19380
```

sjeng with L2-4way

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 32768 : ways = 4 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 40642222783; Total refs = 10000000109
Flush time = 583280378
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     9790456418   [24.1%]
  Writes =     6315918997   [15.5%]
  Inst.  =    24535847368   [60.4%]
  Total  =    40642222783

Average cycles per activity
  Read = 5.1; Write = 8.7; Inst. = 5.5
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12329298534  Miss Count = 223077258
  Total Requests = 12552375792
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 218150206;  Dirty Kickouts = 0;  Transfers = 223077258
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3176318967  Miss Count = 121259542
  Total Requests = 3297578509
  Hit Rate = 96.3%  Miss Rate =  3.7%
  Kickouts = 116336783;  Dirty Kickouts = 53348235;  Transfers = 123513368
  Flush Kickouts = 2253826

Memory Level:  L2
  Hit Count = 311872900  Miss Count = 88065961
  Total Requests = 399938861
  Hit Rate = 78.0%  Miss Rate = 22.0%
  Kickouts = 78298830;  Dirty Kickouts = 20596915;  Transfers = 90855504
  Flush Kickouts = 2789543

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $150;  Memory cost = $75; Total cost = $1025
Flushes = 19380  :  Invalidates = 19380
```

sjeng with L2-Big

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 2 : block size = 32
  Lcache size = 8192 : ways = 2 : block size = 32
  L2-cache size = 65536 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 8 : chunktime = 15

Execute time = 38392682409; Total refs = 10000000109
Flush time = 763553756
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     9162621177   [23.9%]
  Writes =     5899886935   [15.4%]
  Inst.  =    23330174297   [60.8%]
  Total  =    38392682409

Average cycles per activity
  Read = 4.8; Write = 8.1; Inst. = 5.2
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12329298534  Miss Count = 223077258
  Total Requests = 12552375792
  Hit Rate = 98.2%  Miss Rate =  1.8%
  Kickouts = 218150206;  Dirty Kickouts = 0;  Transfers = 223077258
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3176318967  Miss Count = 121259542
  Total Requests = 3297578509
  Hit Rate = 96.3%  Miss Rate =  3.7%
  Kickouts = 116336783;  Dirty Kickouts = 53348235;  Transfers = 123513368
  Flush Kickouts = 2253826

Memory Level:  L2
  Hit Count = 322584702  Miss Count = 77354159
  Total Requests = 399938861
  Hit Rate = 80.7%  Miss Rate = 19.3%
  Kickouts = 64452070;  Dirty Kickouts = 16389234;  Transfers = 81472199
  Flush Kickouts = 4118040

L1 cache cost (Icache $400) + (Dcache $400) = $800
L2 cache cost = $100;  Memory cost = $75; Total cost = $975
Flushes = 19380  :  Invalidates = 19380
```

sjeng with Default-16

```
------------------------------------------------------------------------
                          Simulation Results
------------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 16 : chunktime = 15

Execute time = 42429766216; Total refs = 10000000109
Flush time = 374694433
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =    11763616985   [27.7%]
  Writes =     5802399967   [13.7%]
  Inst.  =    24863749264   [58.6%]
  Total  =    42429766216

Average cycles per activity
  Read = 6.2; Write = 8.0; Inst. = 5.8
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12315460746  Miss Count = 236915046
  Total Requests = 12552375792
  Hit Rate = 98.1%  Miss Rate =  1.9%
  Kickouts = 232095466;  Dirty Kickouts = 0;  Transfers = 236915046
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3108702506  Miss Count = 188876003
  Total Requests = 3297578509
  Hit Rate = 94.3%  Miss Rate =  5.7%
  Kickouts = 184181921;  Dirty Kickouts = 74706271;  Transfers = 190939943
  Flush Kickouts = 2063940

Memory Level:  L2
  Hit Count = 358958322  Miss Count = 143602938
  Total Requests = 502561260
  Hit Rate = 71.4%  Miss Rate = 28.6%
  Kickouts = 134852808;  Dirty Kickouts = 29789793;  Transfers = 146314021
  Flush Kickouts = 2711083

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $175; Total cost = $625
Flushes = 19380  :  Invalidates = 19380
```

sjeng with Default-32

```
------------------------------------------------------------------
                        Simulation Results
------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 32 : chunktime = 15

Execute time = 37146651796; Total refs = 10000000109
Flush time = 272894353
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     9880697015   [26.6%]
  Writes =     4804045987   [12.9%]
  Inst.  =    22461908794   [60.5%]
  Total  =    37146651796

Average cycles per activity
  Read = 5.2; Write = 6.6; Inst. = 5.0
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12315460746  Miss Count = 236915046
  Total Requests = 12552375792
  Hit Rate = 98.1%  Miss Rate =  1.9%
  Kickouts = 232095466;  Dirty Kickouts = 0;  Transfers = 236915046
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3108702506  Miss Count = 188876003
  Total Requests = 3297578509
  Hit Rate = 94.3%  Miss Rate =  5.7%
  Kickouts = 184181921;  Dirty Kickouts = 74706271;  Transfers = 190939943
  Flush Kickouts = 2063940

Memory Level:  L2
  Hit Count = 358958322  Miss Count = 143602938
  Total Requests = 502561260
  Hit Rate = 71.4%  Miss Rate = 28.6%
  Kickouts = 134852808;  Dirty Kickouts = 29789793;  Transfers = 146314021
  Flush Kickouts = 2711083

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $275; Total cost = $725
Flushes = 19380  :  Invalidates = 19380
```

sjeng with Default-64

```
---------------------------------------------------------------------
                        Simulation Results
---------------------------------------------------------------------

Memory system:
  Dcache size = 8192 : ways = 1 : block size = 32
  Lcache size = 8192 : ways = 1 : block size = 32
  L2-cache size = 32768 : ways = 1 : block size = 64
  Memory ready time = 30 : chunksize = 64 : chunktime = 15

Execute time = 34505094586; Total refs = 10000000109
Flush time = 221994313
Inst refs = 7364538494; Data refs = 2635461615

Number of Reference Types: [Percentage]
  Reads  =     1907768017   [19.1%]
  Writes =      727693598   [ 7.3%]
  Inst.  =     7364538494   [73.6%]
  Total  =    10000000109

Total cycles for activities: [Percentage]
  Reads  =     8939237030   [25.9%]
  Writes =     4304868997   [12.5%]
  Inst.  =    21260988559   [61.6%]
  Total  =    34505094586

Average cycles per activity
  Read = 4.7; Write = 5.9; Inst. = 4.7
Ideal: Exec. Time = 17364538603; CPI = 2.4
Ideal mis-aligned: Exec. Time = 23214492795; CPI = 3.2

Memory Level:  L1i
  Hit Count = 12315460746  Miss Count = 236915046
  Total Requests = 12552375792
  Hit Rate = 98.1%  Miss Rate =  1.9%
  Kickouts = 232095466;  Dirty Kickouts = 0;  Transfers = 236915046
  Flush Kickouts = 0

Memory Level:  L1d
  Hit Count = 3108702506  Miss Count = 188876003
  Total Requests = 3297578509
  Hit Rate = 94.3%  Miss Rate =  5.7%
  Kickouts = 184181921;  Dirty Kickouts = 74706271;  Transfers = 190939943
  Flush Kickouts = 2063940

Memory Level:  L2
  Hit Count = 358958322  Miss Count = 143602938
  Total Requests = 502561260
  Hit Rate = 71.4%  Miss Rate = 28.6%
  Kickouts = 134852808;  Dirty Kickouts = 29789793;  Transfers = 146314021
  Flush Kickouts = 2711083

L1 cache cost (Icache $200) + (Dcache $200) = $400
L2 cache cost = $50;  Memory cost = $375; Total cost = $825
Flushes = 19380  :  Invalidates = 19380
```

cache.c:

```
#include "cache.h"


/*
    ******************************************************************************************

 * Decompose the address into a tag and an index and update the reference struct
 ******************************************************************************************
    */
void decomposeAddress( struct reference* ref, cache_TypeDef cache ) {
    // #define PRINT

    unsigned long long address = ref->address;
    #ifdef PRINT
        printf( "   Full address: %llx \n", address );
    #endif

    // Shift address over so that it only contains index and tag
    int bitsTag = TAG_SIZE[cache];
    int bitsIndex = INDEX_SIZE[cache];
    int bitsBlock = 48 - bitsTag - bitsIndex;
    address = address >> bitsBlock;

    // Initial address shifted to byte boundary
    unsigned long long initialAddress = (ref->address >> 2);

    // Mask the address to get the index
    unsigned long long initialIndex = (address & INDEX_MASK[cache]);

    // Shift address again to get tag
    unsigned long long initialTag = (address >> INDEX_SIZE[cache]);

    // Up to this point should be the base index and tag generated by the address
    // Depending on the number of bytes requested, it may be multiple references
    unsigned long long endAddress = ref->address;
    unsigned long long offset = (unsigned long long) (ref->numBytes - 1);
    endAddress = endAddress + offset;
    #ifdef PRINT
    printf( "   numBytes: %d \n", ref->numBytes );
        printf( "   End Address:  %llx \n", endAddress );
    #endif

    // Shift over by 3 (bytes)
    endAddress = (endAddress >> 2);

    // Check if same as initial address (only one reference necessary)
    if( endAddress == initialAddress ) {
        // Only one reference is needed
        ref->numReferences = 1;
        #ifdef PRINT
        printf( "   One reference \n" );
        #endif

        // Initialize reference
        ref->index = calloc( 1, sizeof(unsigned long long) );
        ref->tag = calloc( 1, sizeof(unsigned long long) );

        // Set reference
```

```
        ref->index[0] = initialIndex;
        ref->tag[0] = initialTag;

        #ifdef PRINT
            printf( "   Index 0: %lld \n", ref->index[0] );
            printf( "   Tag 0:   %lld \n", ref->tag[0] );
        #endif
    }
    // Otherwise need to calculate additional references
    else {
        // Number of references needed
        int increment = (int)(endAddress - initialAddress);
        ref->numReferences = increment + 1;
        #ifdef PRINT
        printf( "   Number of References: %d \n", increment+1 );
        #endif

        // Initialize references
        ref->index = calloc( increment+1, sizeof(unsigned long long) );
        ref->tag = calloc( increment+1, sizeof(unsigned long long) );

        // Set initial reference
        ref->index[0] = initialIndex;
        ref->tag[0] = initialTag;
        #ifdef PRINT
            printf( "   Index 0: %llx \n", ref->index[0] );
            printf( "   Tag 0:   %llx \n", ref->tag[0] );
        #endif

        // For each increment, assign tag and index to the reference
        int i;
        unsigned long long currentAddress = initialAddress;
        unsigned long long addr;
        unsigned long long index = initialIndex;
        unsigned long long tag = initialTag;
        for( i = 1; i < increment+1; i++ ) {

            // Increment address by 1 word
            currentAddress = currentAddress + 1;

            // Make addr
            addr = (currentAddress >> (bitsBlock-2));

            // Get associated index
            index = (addr & INDEX_MASK[cache]);

            // Get associated tag
            tag = (addr >> INDEX_SIZE[cache]);

            ref->index[i] = index;
            ref->tag[i] = tag;

            #ifdef PRINT
                printf( "   Index %d: %llx \n", i, ref->index[i] );
                printf( "   Tag %d:   %llx \n", i, ref->tag[i] );
            #endif

        }
    }
}
```

```
/*
    ************************************************************************************

 * Construct cache
 ************************************************************************************
    */
void constructCache( struct cache* cache, cache_TypeDef cacheType, char c ) {
    // Calculate number of blocks in the cache
    int numBlocks = CACHE_SIZE[cacheType] / BLOCK_SIZE[cacheType];

    // Create array of cacheBlocks
    struct cacheBlock* block = calloc( numBlocks, sizeof(struct cacheBlock) );

    // Initialize each cacheBlock
    int i;
    int associativity = ASSOC[cacheType];
    for( i=0; i<numBlocks; i++ ) {
        // Initialize valid to zero (all tags are invalid)
        bool* valid = (bool*) calloc( associativity, sizeof(bool) );

        // Initialize dirty to zero (all tags are NOT dirty)
        bool* dirty = (bool*) calloc( associativity, sizeof(bool) );

        // Initialize tags to zero (this is arbitrary)
        unsigned long long* tags = (unsigned long long*) calloc( associativity, sizeof(
            unsigned long long) );

        //Create the LRU instance and check the cache type
        LRU_inst* LRU = (LRU_inst*) calloc (1, sizeof(LRU_inst) );
        LRU->type = cacheType;

        // Link initialized arrays to associated cacheBlock
        block[i].valid = valid;
        block[i].dirty = dirty;
        block[i].tags = tags;
        block[i].LRU = LRU;

    }

    // Point cache struct to array
    cache->block = block;

    // Set cache type
    cache->type = cacheType;

    // If L1 cache, set L1_type
    if( cacheType == L1 ) {
        cache->L1_Type = c;
    }
}


/*
    ************************************************************************************

 * Add given reference to the cache
 ************************************************************************************
    */
```

```
void addCache( unsigned long long index, unsigned long long tag, struct cache* cache, char
    instructionType ) {
    // #define PRINT

    // Get associated block
    struct cacheBlock block = cache->block[index];

    // Check for any invalid blocks first
    int i;
    int associativity = ASSOC[cache->type];
    for( i=0; i<associativity; i++ ) {
        if( block.valid[i] == FALSE ) {
            #ifdef PRINT
            printf( "   Success, added %llx \n", tag );
            #endif

            // Add tag in this position
            block.valid[i] = TRUE;
            block.dirty[i] = FALSE;
            block.tags[i] = tag;

            // Indicate this index was recently used
            LRUpush ( block.LRU , i );

            // Don't ever need to writeback since data is invalid
            return;
        }
    }

    // If there are no invalid blocks, replace the least recently used block
    if ( (int) block.LRU->count == associativity) {
        #ifdef PRINT
        printf( "   Success, added %llx \n", tag );
        #endif

        // Get most least recently used index
        int tagIndex = LRUpop( block.LRU );

        // If block is dirty, reset and indicate writeback
        if( block.dirty[tagIndex] == TRUE ) {
            #ifdef PRINT
            printf( "   Dirty kickout \n" );
            #endif
            block.dirty[tagIndex] = FALSE;

            // If L1 cache, need to writeback to L2 cache
            if( cache->type == L1 ) {
                writeback( index, block.tags[tagIndex], instructionType );

                // Have to transfer block from L1 to L2
                if( instructionType == 'R' ) {
                    runResults.numReadCycles += config.L1_transfer_cycles;
                } else if( instructionType == 'W' ) {
                    runResults.numWriteCycles += config.L1_transfer_cycles;
                } else if( instructionType == 'F' ) {
                    runResults.flushTime += config.L2_transfer_cycles;
                } else {
                    runResults.numInstCycles += config.L2_transfer_cycles;
                }
```

```
                // Increment dirty kickout for L1 (assuming data cache)
                if( cache->L1_Type == 'I' ) {
                    runResults.l1i_dirtyKickouts++;
                } else {
                    runResults.l1d_dirtyKickouts++;
                }
            }
            // Otherwise need to access main memory
            else {
                // Increment dirty kickout for L2
                runResults.l2_dirtyKickouts++;

                // Have to transfer block from L1 to L2
                if( instructionType == 'R' ) {
                    runResults.numReadCycles += config.L2_transfer_cycles;
                } else if( instructionType == 'W' ) {
                    runResults.numWriteCycles += config.L2_transfer_cycles;
                } else if( instructionType == 'F' ) {
                    runResults.flushTime += config.L2_transfer_cycles;
                } else {
                    runResults.numInstCycles += config.L2_transfer_cycles;
                }
            }
        }

        // Indicate kickout has occurred
        if( cache->type == L1 ) {
            if( cache->L1_Type == 'I' ) {
                runResults.l1i_kickouts++;
            } else {
                runResults.l1d_kickouts++;
            }
        } else {
            runResults.l2_kickouts++;
        }

        // Overwrite associated tag
        block.tags[tagIndex] = tag;

        // Indicate index was recently used
        LRUpush ( block.LRU , tagIndex );

    }
}


/*
    ********************************************************************************

 * Query given cache for reference
 ********************************************************************************
    */
bool queryCache( unsigned long long index, unsigned long long tag, struct cache* cache ) {
    // #define PRINT

    // Get associated block
    struct cacheBlock block = cache->block[index];

    // Check tag(s)
    int i;
```

```
    int associativity = ASSOC[cache->type];
    bool hasTag = FALSE;
    for( i=0; i<associativity; i++ ) {
        // Check if tag is valid
        if( block.valid[i] == TRUE ) {
            if( block.tags[i] == tag ) {
                #ifdef PRINT
                printf( "   Tag %llx is contained in cache \n", tag );
                #endif
                hasTag = TRUE;
                LRUpush ( block.LRU , i );
            }
        }
    }

    return hasTag;
}


/*
    ***************************************************************************************

 * Set dirty bit for given reference
 ***************************************************************************************
    */
bool setDirty( unsigned long long index, unsigned long long tag, struct cache* cache ) {
    // #define PRINT

    // Get associated block
    struct cacheBlock block = cache->block[index];

    // Check tag(s)
    int i;
    int associativity = ASSOC[cache->type];
    for( i=0; i<associativity; i++ ) {
        // Check if tag is valid
        if( block.valid[i] == TRUE ) {
            if( block.tags[i] == tag ) {
                #ifdef PRINT
                printf( "   Tag %llx is set to dirty \n", tag );
                #endif

                // Set dirty bit
                block.dirty[i] = TRUE;

                // Update LRU
                LRUpush ( block.LRU , i );

                return TRUE;
            }
        }
    }
    return FALSE;
}


/*
    ***************************************************************************************

 * Construct L2_Tag and L2_Index based on L1_Tag and L1_Index
```

```
 ********************************************************************************************
    */
void constructL2Ref( struct L2_Reference* ref, unsigned long long index, unsigned long
    long tag ) {
    // #define PRINT

    // Reconstruct the original address from the index and tag
    unsigned long long newTag = (tag << INDEX_SIZE[L1]);
    unsigned long long address = ( newTag | index );

    // Get difference in tag+index address size between L1 and L2 (how much larger is L1 "
        address" than L2)
    int difference = (TAG_SIZE[L1] + INDEX_SIZE[L1]) - (TAG_SIZE[L2] + INDEX_SIZE[L2]);

    // Shift address to fit size of L2
    address = address >> difference;

    // Mask the address to get the index
    ref->L2_Index = (address & INDEX_MASK[L2]);
    #ifdef PRINT
        printf( "   Constructed Index: %llx \n", ref->L2_Index );
    #endif

    // Shift address again to get tag
    ref->L2_Tag = (address >> INDEX_SIZE[L2]);
    #ifdef PRINT
        printf( "   Constructed Tag: %llx \n", ref->L2_Tag );
    #endif
}


/*
    ********************************************************************************************

 * Writeback from L1 cache to L2 cache by setting corresponding data to dirty
 ********************************************************************************************
    */
void writeback( unsigned long long index, unsigned long long tag, char instructionType ) {
    // #define PRINT

    // Make L2_Reference struct
    struct L2_Reference ref;

    // Get L2 reference
    constructL2Ref( &ref, index, tag );

    // Set constructed block to dirty
    if( setDirty( ref.L2_Index, ref.L2_Tag, &L2_unified ) ) {
        runResults.l2_hit++;

        // Add hit time
        if( instructionType == 'R' ) {
            runResults.numReadCycles += config.L2_hit_time;
        } else if( instructionType == 'W' ) {
            runResults.numWriteCycles += config.L2_hit_time;
        } else if( instructionType == 'F' ) {
            runResults.flushTime += config.L2_hit_time;
        } else {
            // Do nothing
            printf( "Do nothing \n" );
```

```
        }
    } else {
        runResults.l2_miss++;

        // Need to add appropriate reference to L2 and mark dirty
        addCache( ref.L2_Index, ref.L2_Tag, &L2_unified, instructionType );
        setDirty( ref.L2_Index, ref.L2_Tag, &L2_unified );

        // Add miss time
        if( instructionType == 'R' ) {
            runResults.numReadCycles += config.L2_miss_time + config.L2_hit_time + config.
                L2_transfer_cycles;
        } else if( instructionType == 'W' ) {
            runResults.numWriteCycles += config.L2_miss_time + config.L2_hit_time + config
                .L2_transfer_cycles;
        } else if( instructionType == 'F' ){
            runResults.flushTime += config.L2_miss_time + config.L2_hit_time + config.
                L2_transfer_cycles;
        } else {
            // Do nothing
            printf( "Do nothing \n" );
        }
    }
}


/*
    ************************************************************************************

 * Flush the cache by setting all blocks to invalid
 ************************************************************************************
    */
void flush( struct cache* cache ) {

    // Calculate number of blocks in the cache
    int numBlocks = CACHE_SIZE[cache->type] / BLOCK_SIZE[cache->type];

    // Get associativity
    int associativity = ASSOC[cache->type];

    // Iterate through each cacheBlock
    int i, j;
    struct cacheBlock block;
    for( i = 0; i < numBlocks; i++ ) {
        block = cache->block[i];
        for( j = 0; j < associativity; j++ ) {
            if( block.valid[j] == TRUE ) {

                // If dirty, handle writeback
                if( block.dirty[j] == TRUE ) {
                    block.dirty[j] = FALSE;

                    if( cache->type == L1 ) {
                        writeback( i, block.tags[j], 'F' );

                        // Have to transfer from L1 to L2
                        runResults.flushTime += config.L1_transfer_cycles;

                        // Increment flush kickout for L1
                        if( cache->L1_Type == 'I' ) {
```

```
                        runResults.l1i_flushKickouts++;
                    } else {
                        runResults.l1d_flushKickouts++;
                    }
                }
                // Otherwise need to access main memory
                else {
                    // Increment dirty kickout for L2
                    runResults.l2_flushKickouts++;

                    // Have to transfer from L2 to main memory
                    runResults.flushTime += config.L2_transfer_cycles;

                }
            }
            // Invalidate block
            block.valid[j] = FALSE;

            // Handle LRU (for now, can just pop to remove nodes from LRU)
            LRUpop( block.LRU );
        }
    }
}


/*
    ************************************************************************************

 * Clears the LRU and frees the nodes. Should call when done with the entire program
 ************************************************************************************
    */
void LRUclear (LRU_inst* LRU) {
    // Check the cases where it is single item or null
    if (LRU -> first == LRU -> last) {
        if (!( LRU -> first == NULL )) {
            free (LRU -> first);
        }
    // If its multiple items iterate through them and free memory
    } else {
        while( LRU -> first -> next != LRU -> last ) {
            LRU -> first -> next = LRU -> first -> next -> next;
            free(LRU -> first -> next);
        }
        free( LRU -> first );
        free( LRU -> last );
    }
}


/*
    ************************************************************************************

 * Pushes a arrayIndex to the LRU should never push to something thats full (handled
    elsewhere).
 ************************************************************************************
    */
void LRUpush (LRU_inst* LRU, int arrayIndex) {
    // #define PRINT
```

```
//REMOVED SECTION WHERE IT CHECKS IF THE LRU IS FULL HERE (IT BROKE THINGS)


if (LRU -> first != NULL) {
    LRUnode * parser;
    parser = LRU->first;
    while ( parser->next != NULL ) {
        if ( parser -> arrayIndex == arrayIndex ) {
            if (LRU -> count == 1) {
                // printf("        LRU count is 1, and got hit : count %lu\n", LRU->
                    count);

            } else if ( parser == LRU->first ) {
                free( parser->next->prev );
                LRU -> first = parser -> next;
                LRU->count--;
                // printf("        LRU count decrimented : count %lu\n", LRU->count);
            } else {
                // free(parser -> prev -> next);
                parser -> prev -> next = parser -> next;
                // parser->next->prev = parser->prev;
                if (parser -> next != NULL) {
                    parser -> next -> prev = parser -> prev;
                }
                free( parser );
                LRU->count--;
                // printf("        LRU count decrimented : count %lu\n", LRU->count);
            }
        }
        parser = parser -> next;
    }
    if ( LRU -> last != NULL) {
        if ( LRU -> last -> arrayIndex == arrayIndex) {
            // printf("        Hit at the last index of LRU calling pop\n");
            LRUpop(LRU);
            // printf("        After pop called : count %lu\n", LRU->count);
        }
    }
}

// Make the node with the value of the arrayIndex
LRUnode * node = calloc( 1, sizeof(LRUnode) );
node->arrayIndex = arrayIndex;
// printf("        Creating node at %i\n", arrayIndex);

// Empty LRU case
if (LRU->first == NULL) {
    LRU->first = node;
    LRU->last = node;
    node -> next = NULL;
    node -> prev = NULL;
} else {
    LRU->first->prev = node;
    node->next = LRU->first;
    LRU->first = node;
}
LRU->count++;
// printf("        New count : %lu\n", LRU->count);
#ifdef PRINT
// printf("   The size of the LRU is %lu \n",LRU->count);
```

```
    // printf("   The last item in the list's tag is %i\n",LRU->last->arrayIndex);
    #endif
}


/*
    ***************************************************************************************

 * Pops an item off the LRU, the int of the last used arrayIndex that needs to be replaced
 ***************************************************************************************
    */
int LRUpop (LRU_inst* LRU) {
    // Return value
    int result;

    // Empty LRU case, doesn't need to pop, returns 0
    if( LRU -> count == 0 ) {
        result = 0;
    } else {
        if ( LRU -> count == 1 ) {
            // if there is only one item
            free( LRU -> first );
            LRU -> last = NULL;
            LRU -> first = NULL;
            result = 0;
        } else {
            // more than one item, really is only called when items = associativity
            // printf( "   In else \n" );
            // printf( "   Count: %d \n", LRU->count );
            result = LRU -> last -> arrayIndex;
            LRU -> last = LRU -> last -> prev;
            free ( LRU -> last -> next );
            LRU -> last -> next = NULL;
        }
        LRU->count--;
        // printf("        decrimented LRU counter : count %lu\n", LRU->count);
    }
    return result;
}
```

cache.h:

```
#ifndef CACHE_H
#define CACHE_H

#include <stdio.h>
#include <stdbool.h>
#include "configparse.h"
#include "print.h"


// Comment this line to suppress print statements
//#define PRINT

#define TRUE                    1
#define FALSE                   0


/*
    ************************************************************************************

 * Struct to represent each item in the LRU
 ************************************************************************************
    */
typedef struct LRUnode {
    struct LRUnode* prev;
    struct LRUnode* next;
    int arrayIndex;
    bool valid;
    bool dirty;
} LRUnode;

/*
    ************************************************************************************

 * Struct to represent LRU instance
 ************************************************************************************
    */
typedef struct LRU_inst {
    cache_TypeDef type;
    int count;
    LRUnode* first;
    LRUnode* last;
} LRU_inst;

/*
    ************************************************************************************

 * Struct to represent each block of the cache
 *      NOTE: Everything is represented as an array to support associative caches
 ************************************************************************************
    */
struct cacheBlock {
    bool* valid;                            // Pointer to valid array
    bool* dirty;                            // Pointer to dirty array
    unsigned long long* tags;               // Pointer to tag array
    LRU_inst* LRU;                          // Pointer to LRU
    // Block: would put data here but the simulation doesn't put actual data into the
        cache, just makes calls to it
};
```

```
/*
    ************************************************************************************

 * Struct to represent each cache
 ************************************************************************************
    */
struct cache {
    cache_TypeDef type;                     // Type of cache (L1 or L2)
    char L1_Type;                           // Data or Instruction cache (for L1 only)
    struct cacheBlock* block;               // Pointer to an array of cacheBlocks
};


/*
    ************************************************************************************

 * Define the three caches
 ************************************************************************************
    */
struct cache L1_instruction;
struct cache L1_data;
struct cache L2_unified;


/*
    ************************************************************************************

 * Struct to represent the information provided by the program trace
 ************************************************************************************
    */
struct reference {
    char type;                              // I, R, W (Instruction, Read Data, Write Data
        )
    unsigned long long address;             // 48 bits
    unsigned long long* tag;                // Array of L1 tags to access
    unsigned long long* index;              // Array of L1 indexes to access
    int numReferences;                      // Number of aligned references requested
    int numBytes;                           // Number of bytes requested
};


/*
    ************************************************************************************

 * Struct to represent index and tag of L2 cache
 ************************************************************************************
    */
struct L2_Reference {
    unsigned long long L2_Tag;              // L2 tag to access
    unsigned long long L2_Index;            // L2 index to access
};


/*
    ************************************************************************************

 * Clear an LRU
 ************************************************************************************
```

```
    */
void LRUclear (LRU_inst* LRU);


/*
    ************************************************************************************

 * Push an arrayindex onto the LRU
 ************************************************************************************
    */
void LRUpush (LRU_inst* LRU, int arrayIndex);


/*
    ************************************************************************************

 * Pop an array index off the LRU, used to add new tag with same index when tag[] is full
 ************************************************************************************
    */
int LRUpop (LRU_inst* LRU);


/*
    ************************************************************************************

 * Define reference used for all traces
 ************************************************************************************
    */
struct reference ref;


/*
    ************************************************************************************

 * Decompose the address into a tag and an index and update the reference struct
 ************************************************************************************
    */
void decomposeAddress( struct reference* ref, cache_TypeDef cache );


/*
    ************************************************************************************

 * Construct cache
 ************************************************************************************
    */
void constructCache( struct cache* cache, cache_TypeDef cacheType, char c );


/*
    ************************************************************************************

 * Add given reference to the cache
 ************************************************************************************
    */
void addCache( unsigned long long index, unsigned long long tag, struct cache* cache, char
    instructionType );


/*
```

```
   ***************************************************************************************************
 * Query given cache for reference
 ***************************************************************************************************
     */
bool queryCache( unsigned long long index, unsigned long long tag, struct cache* cache );


/*
   ***************************************************************************************************
 * Set dirty bit for given reference
 ***************************************************************************************************
     */
bool setDirty( unsigned long long index, unsigned long long tag, struct cache* cache );


/*
   ***************************************************************************************************
 * Construct L2_Tag and L2_Index based on L1_Tag and L1_Index
 ***************************************************************************************************
     */
void constructL2Ref( struct L2_Reference* ref, unsigned long long index, unsigned long
   long tag );


/*
   ***************************************************************************************************
 * Writeback from L1 cache to L2 cache by setting corresponding data to dirty
 ***************************************************************************************************
     */
void writeback( unsigned long long index, unsigned long long tag, char instructionType );


/*
   ***************************************************************************************************
 * Flush the cache by setting all blocks to invalid
 ***************************************************************************************************
     */
void flush( struct cache* cache );


#endif // CACHE_H
```

configparse.c:

```
#include "configparse.h"

/*
    ********************************************************************************************

 * Parses the given configuration file and fills in the values of the configuration struct
********************************************************************************************
    */
int configParse( char* configFile ) {
    int count = 0;
        int value = 0;
        FILE* fp;
        char buffer[BUFFER_SIZE];
    // Open configuration file
    if( (fp = fopen(configFile, "r")) != NULL ) {
        // Iterate over each line
        while( fgets(buffer, BUFFER_SIZE, fp) != NULL ) {
                    if( *buffer == '#' ) {
            #ifdef PRINT
                            printf( "%s", buffer );
            #endif
                    } else {
            count += 1;
                            // Convert string to an int
                            value = (int) strtol(buffer, (char**)NULL, 10);
            #ifdef PRINT
            printf( "%d: %d \n",count, value );
            #endif

                            // Basic implementation expects certain lines to contain
                               the correct variables and doesn't make any check
                            if( count == 1 ) {
                                    config.L1_block_size = value;
                            } else if( count ==2 ) {
                                    config.L1_cache_size = value;
                            } else if( count ==3 ) {
                                    config.L1_assoc = value;
                            } else if( count ==4 ) {
                                    config.L1_hit_time = value;
                            } else if( count ==5 ) {
                                    config.L1_miss_time = value;
                            } else if( count ==6 ) {
                                    config.L2_block_size = value;
                            } else if( count ==7 ) {
                                    config.L2_cache_size = value;
                            } else if( count ==8 ) {
                                    config.L2_assoc = value;
                            } else if( count ==9 ) {
                                    config.L2_hit_time = value;
                            } else if( count ==10 ) {
                                    config.L2_miss_time = value;
                            } else if( count ==11 ) {
                                    config.L2_transfer_time = value;
                            } else if( count ==12 ) {
                                    config.L2_bus_width = value;
                            } else if( count ==13 ) {
                        config.mem_sendaddr = value;
            } else if( count ==14 ) {
```

```
                    config.mem_ready = value;
                } else if( count ==15 ) {
                    config.mem_chunktime = value;
                } else if( count ==16 ) {
                    config.mem_chunksize = value;
                }
                        }
        }
    }
    fclose(fp);

    // Define the values of the arrays
    BLOCK_SIZE[0] = config.L1_block_size;
    BLOCK_SIZE[1] = config.L2_block_size;

    CACHE_SIZE[0] = config.L1_cache_size;
    CACHE_SIZE[1] = config.L2_cache_size;

    ASSOC[0] = config.L1_assoc;
    ASSOC[1] = config.L2_assoc;

    defineAddressParameters( 0 );
    defineAddressParameters( 1 );

    config.L1_transfer_cycles = config.L2_transfer_time * (config.L1_block_size/config.
        L2_bus_width);
    config.L2_transfer_cycles = config.mem_sendaddr + config.mem_ready + config.
        mem_chunktime * (config.L2_block_size/config.mem_chunksize);

        return 0;
}


/*
    ****************************************************************************************

 * Calculates and sets TAG_SIZE and INDEX_SIZE arrays for given cache
 ****************************************************************************************
    */
void defineAddressParameters( cache_TypeDef cache ) {
    // Calculate number of blocks in the cache
    int numBlocks = CACHE_SIZE[cache] / BLOCK_SIZE[cache];
    #ifdef PRINT
    printf( "Blocks in cache %d: %d \n", cache, numBlocks );
    #endif

    // Calculate number of bits used for the index (ignoring associativity)
    int bitsIndex = log(numBlocks) / log(2);
    #ifdef PRINT
    printf( "Number of index bits (%d): %d \n", cache, bitsIndex );
    #endif

    // Adjust for associativity
    int bitsAssociative = log(ASSOC[cache]) / log(2);
    bitsIndex = bitsIndex - bitsAssociative;
    #ifdef PRINT
    printf( "Number of associative bits (%d): %d \n", cache, bitsAssociative );
    printf( "Adjusted number of index bits (%d): %d \n", cache, bitsIndex );
    #endif
```

```
    // Calculate number of bits used to address each byte in the block
    int bitsBlock = log(BLOCK_SIZE[cache]) / log(2);
    #ifdef PRINT
    printf( "Number of block bits (%d): %d \n", cache, bitsBlock );
    #endif

    // Size of the address string
    int numAddress = 48;

    // Calculate how many bits are left over from the address
    int bitsTag = numAddress - bitsIndex - bitsBlock;
    #ifdef PRINT
    printf( "Number of tag bits (%d): %d \n", cache, bitsTag );
    #endif

    TAG_SIZE[cache] = bitsTag;
    INDEX_SIZE[cache] = bitsIndex;

    // Create bit mask to get index
    int i;
    unsigned long long mask = 0;
    unsigned long long tempMask = 0;
    for( i = 0; i < bitsIndex; i++ ) {
        tempMask = 1 << i;
        mask = (mask | tempMask);
    }
    #ifdef PRINT
    printf( "Index mask: %lld \n",mask );
    #endif
    INDEX_MASK[cache] = mask;

    // Create bit mask to get bytes
    mask = 0;
    tempMask = 0;
    for( i = 0; i < bitsBlock; i++ ) {
        tempMask = 1 << i;
        mask = (mask | tempMask);
    }
    #ifdef PRINT
    printf( "Byte mask: %lld \n",mask );
    #endif
    BYTE_MASK[cache] = mask;


}
```

configparse.h:

```
#ifndef CONFIGPARSE_H
#define CONFIGPARSE_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

// Comment this line to suppress print statements
// #define PRINT

#define BUFFER_SIZE                                     100
#define DELIM                                           "="


// Defines L1 and L2 parameters
typedef enum {
    L1 = 0,
    L2 = 1
} cache_TypeDef;


/*
    ********************************************************************************************

 * Parses the given configuration file and fills in the values of the configuration struct
*********************************************************************************************
    */
int configParse( char* );


/*
    ********************************************************************************************

 * Calculates and sets TAG_SIZE and INDEX_SIZE arrays for given cache
 *********************************************************************************************
    */
void defineAddressParameters( cache_TypeDef );


/*
    ********************************************************************************************

 * Struct used to access cache parameters
*********************************************************************************************
    */
struct configuration {
        // L1 parameters
        int L1_block_size;
        int L1_cache_size;
        int L1_assoc;
        int L1_hit_time;
        int L1_miss_time;
        // L2 parameters
        int L2_block_size;
        int L2_cache_size;
        int L2_assoc;
        int L2_hit_time;
```

```
        int L2_miss_time;
        int L2_transfer_time;
        int L2_bus_width;
        // Memory Parameters
        int mem_sendaddr;
        int mem_ready;
        int mem_chunktime;
        int mem_chunksize;
        // Calculated Parameters
        int L1_transfer_cycles;
        int L2_transfer_cycles;
};

struct configuration config;


/*
    ********************************************************************************************

 * Arrays used to index the configuration parameters using a single cache parameter
********************************************************************************************
    */
int BLOCK_SIZE[2];
int CACHE_SIZE[2];
int ASSOC[2];
int TAG_SIZE[2];
int INDEX_SIZE[2];
unsigned long long INDEX_MASK[2];
unsigned long long BYTE_MASK[2];


#endif // CONFIGPARSE_H
```

main.c:

```
// Contains main function to run the simulation
#include "main.h"

int main( int argc, char** argv ) {

        if( argc < 2 ) {
                configParse( "config" );
        } else {
                configParse( argv[1] );
        }

    // Initialize cache structures
    constructCache( &L1_instruction , L1, 'I' );
    constructCache( &L1_data , L1, 'D' );
    constructCache( &L2_unified , L2, '2' );

        while( scanf ("%c %llx %d\n",&ref.type,&ref.address,&ref.numBytes) == 3) {
                // printf("\n%c %llx %d\n\n",ref.type,ref.address,ref.numBytes);

        // Begin the statemachine
        stateMachine( &ref );

        }

        printCaches( "cacheResults.results" );

    printf ("calculating cost \n");
    runResults.l1_cost = 2 * (100 * (config.L1_cache_size/4096) + 100 * (log(config.
       L1_assoc) / log(2)) * (config.L1_cache_size/4096));
    runResults.l2_cost = (int)(50 * ((float) config.L2_cache_size/65536) + 50 * (log(
       config.L2_assoc) / log(2)) * ((float)config.L2_block_size/65536));
    runResults.mem_cost = 50 + 200 * (log ( 50 / config.mem_ready )) / log(2) + 25 + 100 *
        (log (config.mem_chunksize/16)) / log(2);

        runResults.m_config = config;
        if ( argc < 3) {
                printResults( "resultsFile.results" );
        } else {
                printResults( argv[2] );
        }

    return 1;
}
```

main.h:

```
#include <stdio.h>
#include <unistd.h>
#include "configparse.h"
#include "cache.h"
#include "print.h"
#include "statemachine.h"

// Comment this line to suppress print statements
#define PRINT
```

print.c:

```
#include "print.h"

int printResults ( char* resultsFile ) {
        FILE * fp;
        fp = fopen( resultsFile , "w");
        fprintf(fp,"
            ------------------------------------------------------------\n       
            ___________________Simulation Results_____________________________\n
            ------------------------------------------------------------\n\n");
        fprintf(fp,"Memory system:\n  Dcache size = %i : ways = %i : block size = %i \n  
            Lcache size = %i : ways = %i : block size = %i \n  L2-cache size = %i : ways = 
            %i : block size = %i \n  Memory ready time = %i : chunksize = %i : chunktime = 
            %i",
                runResults.m_config.L1_cache_size,
                runResults.m_config.L1_assoc,
                runResults.m_config.L1_block_size,
                runResults.m_config.L1_cache_size,
                runResults.m_config.L1_assoc,
                runResults.m_config.L1_block_size,
                runResults.m_config.L2_cache_size,
                runResults.m_config.L2_assoc,
                runResults.m_config.L2_block_size,
                runResults.m_config.mem_ready,
                runResults.m_config.mem_chunksize,
                runResults.m_config.mem_chunktime
        );
        runResults.numInstCycles += runResults.flushTime;
        unsigned long long totalCycle = runResults.numReadCycles + runResults.
            numWriteCycles + runResults.numInstCycles;
        unsigned long long totalRef = runResults.numReads + runResults.numWrites +
            runResults.numInst;
        fprintf(fp,"\n\nExecute time = %llu; Total refs = %llu \nFlush time = %llu \nInst 
            refs = %llu; Data refs = %llu",
                totalCycle,
                totalRef,
                runResults.flushTime,
                runResults.numInst,
                runResults.numReads + runResults.numWrites
        );
        fprintf(fp,"\n\nNumber of Reference Types: [Percentage]\n  Reads  = %14llu   [%4.1
            f%%]\n  Writes = %14llu   [%4.1f%%]\n  Inst.  = %14llu   [%4.1f%%]\n  Total  = 
            %14llu",
                runResults.numReads,
                ((float)runResults.numReads / (float)totalRef)*100,
                runResults.numWrites,
                ((float)runResults.numWrites / (float)totalRef)*100,
                runResults.numInst,
                ((float)runResults.numInst / (float)totalRef)*100,
                totalRef
        );
        fprintf(fp,"\n\nTotal cycles for activities: [Percentage]\n  Reads  = %14llu   
            [%4.1f%%]\n  Writes = %14llu   [%4.1f%%]\n  Inst.  = %14llu   [%4.1f%%]\n  
            Total  = %14llu",
                runResults.numReadCycles,
                ((float)runResults.numReadCycles / (float)totalCycle)*100,
                runResults.numWriteCycles,
                ((float)runResults.numWriteCycles / (float)totalCycle)*100,
                runResults.numInstCycles,
```

```
        ((float)runResults.numInstCycles / (float)totalCycle)*100,
        totalCycle
);
fprintf(fp,"\n\nAverage cycles per activity\n  Read = %.1f; Write = %.1f; Inst. = %.1f",
        (float)runResults.numReadCycles / (float)runResults.numReads,
        (float)runResults.numWriteCycles / (float)runResults.numWrites,
        (float)totalCycle / (float)runResults.numInst
);

unsigned long long l1i_total = runResults.l1i_hit + runResults.l1i_miss;
unsigned long long l1d_total = runResults.l1d_hit + runResults.l1d_miss;

fprintf(fp, "\nIdeal: Exec. Time = %llu; CPI = %.1f\nIdeal mis-aligned: Exec. Time = %llu; CPI = %.1f",
        totalRef + runResults.numInst,
        (float)(totalRef + runResults.numInst)/(float)runResults.numInst,
        l1i_total + l1d_total + runResults.numInst,
        (float)(l1i_total + l1d_total + runResults.numInst)/(float)runResults.
            numInst
);

fprintf(fp, "\n\nMemory Level:  L1i\n  Hit Count = %llu  Miss Count = %llu\n  Total Requests = %llu\n  Hit Rate = %4.1f%%  Miss Rate = %4.1f%%\n  Kickouts = %llu;  Dirty Kickouts = %llu;  Transfers = %llu\n  Flush Kickouts = %llu",
        runResults.l1i_hit,
        runResults.l1i_miss,
        l1i_total,
        ((float)runResults.l1i_hit/(float)l1i_total)*100,
        ((float)runResults.l1i_miss/(float)l1i_total)*100,
        runResults.l1i_kickouts,
        runResults.l1i_dirtyKickouts,
        runResults.l1i_miss + runResults.l1i_flushKickouts,
        runResults.l1i_flushKickouts
);

fprintf(fp, "\n\nMemory Level:  L1d\n  Hit Count = %llu  Miss Count = %llu\n  Total Requests = %llu\n  Hit Rate = %4.1f%%  Miss Rate = %4.1f%%\n  Kickouts = %llu;  Dirty Kickouts = %llu;  Transfers = %llu\n  Flush Kickouts = %llu",
        runResults.l1d_hit,
        runResults.l1d_miss,
        l1d_total,
        ((float)runResults.l1d_hit/(float)l1d_total)*100,
        ((float)runResults.l1d_miss/(float)l1d_total)*100,
        runResults.l1d_kickouts,
        runResults.l1d_dirtyKickouts,
        runResults.l1d_miss + runResults.l1d_flushKickouts,
        runResults.l1d_flushKickouts
);
unsigned long long l2_total = runResults.l2_hit + runResults.l2_miss;
fprintf(fp, "\n\nMemory Level:  L2\n  Hit Count = %llu  Miss Count = %llu\n  Total Requests = %llu\n  Hit Rate = %4.1f%%  Miss Rate = %4.1f%%\n  Kickouts = %llu;  Dirty Kickouts = %llu;  Transfers = %llu\n  Flush Kickouts = %llu",
        runResults.l2_hit,
        runResults.l2_miss,
        l2_total,
        ((float)runResults.l2_hit/(float)l2_total)*100,
        ((float)runResults.l2_miss/(float)l2_total)*100,
        runResults.l2_kickouts,
        runResults.l2_dirtyKickouts,
```

```
                runResults.l2_miss + runResults.l2_flushKickouts,
                runResults.l2_flushKickouts
        );
        //Need more here, cost calculations + memory cost
        int l1_cost = 2 * (100 * (config.L1_cache_size/4096) + 100 * (log(config.L1_assoc)
            / log(2)) * (config.L1_cache_size/4096));
    int l2_cost = (50 * (config.L2_cache_size/32768) + 50 * (log(config.L2_assoc) / log(2)
        ) * (config.L2_cache_size/32768));
    int mem_cost = 50 + 200 * (log ( 30 / config.mem_ready )) / log(2) + 25 + 100 * (log (
        config.mem_chunksize/8)) / log(2);
        fprintf(fp, "\n\nL1 cache cost (Icache $%i) + (Dcache $%i) = $%i\nL2 cache cost = $%i;  Memory cost = $%i; Total cost = $%i\nFlushes = %llu   :   Invalidates = %
            llu",
                l1_cost / 2,
                l1_cost / 2,
                l1_cost,
                l2_cost,
                mem_cost,
                (l1_cost + l2_cost + mem_cost),
                runResults.flushes,
                runResults.invalidates
        );



        fclose(fp);
        return 0;
}

void printCaches ( char* cacheFile) {
        FILE * fp;
        fp = fopen( cacheFile , "w");
        int i, j;
        int numBlocksL1 = CACHE_SIZE[L1] / BLOCK_SIZE[L1];
        int numBlocksL2 = CACHE_SIZE[L2] / BLOCK_SIZE[L2];

        fprintf(fp,  "Memory Level:  L1i\n");
        for ( i = 0; i < numBlocksL1; i++) {
                if ( L1_instruction.block[i].LRU -> count > 0 ) {
                        fprintf(fp,  "Index:%5x |", i );
                        for( j = 0; j < ASSOC[L1]; j++) {
                                if ( L1_instruction.block[i].LRU -> count > j ) {
                                        if ( L1_instruction.block[i].tags[j] == 0 ) {
                                                fprintf(fp, " V:%d D:%d Tag: - |",
                                                        L1_instruction.block[i].valid[j] ,
                                                        L1_instruction.block[i].dirty[j]
                                                );
                                        } else {
                                                fprintf(fp, " V:%d D:%d Tag:%13llx |",
                                                        L1_instruction.block[i].valid[j] ,
                                                        L1_instruction.block[i].dirty[j] ,
                                                        L1_instruction.block[i].tags[j]
                                                );
                                        } if ( j % 2 == 1) {
                                                fprintf(fp, "\n            |");
                                        }
                                }
                        }
                        fprintf(fp, "\n");
                }
```

```
        }
        fprintf(fp, "\n");

        fprintf(fp,  "Memory Level:  L1d\n");
        for ( i = 0; i < numBlocksL1; i++) {
                if ( L1_data.block[i].LRU -> count > 0 ) {
                        fprintf(fp,  "Index:%5x |", i );
                        for( j = 0; j < ASSOC[L1]; j++) {
                                if ( (int)L1_data.block[i].LRU -> count > j ) {
                                        if ( L1_data.block[i].tags[j] == 0 ) {
                                                fprintf(fp, " V:%d D:%d Tag: - |",
                                                        L1_data.block[i].valid[j] ,
                                                        L1_data.block[i].dirty[j]
                                                );
                                        } else {
                                                fprintf(fp, " V:%d D:%d Tag:%13llx |",
                                                        L1_data.block[i].valid[j] ,
                                                        L1_data.block[i].dirty[j] ,
                                                        L1_data.block[i].tags[j]
                                                );
                                        } if ( j % 2 == 1) {
                                                fprintf(fp, "\n            |");
                                        }

                                }
                        }
                        fprintf(fp, "\n");
                }
        }
        fprintf(fp, "\n");

        fprintf(fp,  "Memory Level:  L2\n");
        for ( i = 0; i < numBlocksL2; i++) {
                if ( L2_unified.block[i].LRU -> count > 0 ) {
                        fprintf(fp,  "Index:%5x |", i );
                        for( j = 0; j < ASSOC[L2]; j++) {
                                if ( (int)L2_unified.block[i].LRU -> count > j ) {
                                        if ( L2_unified.block[i].tags[j] == 0 ) {
                                                fprintf(fp, " V:%d D:%d Tag: - |",
                                                        L2_unified.block[i].valid[j] ,
                                                        L2_unified.block[i].dirty[j]
                                                );
                                        } else {
                                                fprintf(fp, " V:%d D:%d Tag:%13llx |",
                                                        L2_unified.block[i].valid[j] ,
                                                        L2_unified.block[i].dirty[j] ,
                                                        L2_unified.block[i].tags[j]
                                                );
                                        } if ( j % 2 == 1) {
                                                fprintf(fp, "\n            |");
                                        }
                                }
                        }
                        fprintf(fp, "\n");
                }
        }
        fprintf(fp, "\n");
}
```

print.h:

```
#ifndef PRINT_H
#define PRINT_H

#include "configparse.h"
#include "cache.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

/*
    ********************************************************************************************

 * Print cache to verify entries
 ********************************************************************************************
    */
void printCaches( char* cacheFile);


/*
    ********************************************************************************************

 * Function to print results to a file of choosing, uses the results structure
 ********************************************************************************************
    */
int printResults( char* );


/*
    ********************************************************************************************

 * Struct to hold calculated results
********************************************************************************************
    */
struct results {
        struct configuration m_config;
        unsigned long long exTime;
        unsigned long long flushTime;
        unsigned long long numReads;
        unsigned long long numWrites;
        unsigned long long numInst;
        unsigned long long numReadCycles;
        unsigned long long numWriteCycles;
        unsigned long long numInstCycles;
        unsigned long long l1i_hit;
        unsigned long long l1i_miss;
        unsigned long long l1i_total;
        float l1i_hitRate;
        float l1i_missRate;
        unsigned long long l1i_kickouts;
        unsigned long long l1i_dirtyKickouts;
        unsigned long long l1i_flushKickouts;
        unsigned long long l1d_hit;
        unsigned long long l1d_miss;
        unsigned long long l1d_total;
        float l1d_hitRate;
        float l1d_missRate;
        unsigned long long l1d_kickouts;
        unsigned long long l1d_dirtyKickouts;
```

```
        unsigned long long l1d_flushKickouts;
        unsigned long long l2_hit;
        unsigned long long l2_miss;
        unsigned long long l2_total;
        float l2_hitRate;
        float l2_missRate;
        unsigned long long l2_kickouts;
        unsigned long long l2_dirtyKickouts;
        unsigned long long l2_flushKickouts;
        unsigned int l1_cost;
        unsigned int l2_cost;
        unsigned int mem_cost;
        unsigned long long flushes;
        unsigned long long invalidates;
};

struct results runResults;


#endif // PRINT_H
```

statemachine.c:

```
#include "statemachine.h"


/*
    ********************************************************************************************

 * Makes call to state machine
 ********************************************************************************************
    */
void stateMachine( struct reference* ref ) {

    // Find which L1 cache to access
    struct cache L1_cache = cacheType( ref );

    // Define initial state
    struct state state;
    state.type = ref->type;
    state.iteration = 0;

    // Check instruction count
    if( setFlush == TRUE ) {
        state.next = FLUSH;
    } else {
        state.next = QUERY_L1;
    }

    // Define initial L1 reference
    decomposeAddress( ref, L1 );
    state.L1_Index = ref->index[0];
    state.L1_Tag = ref->tag[0];

    #ifdef PRINT
    printf( "Tag:   %13llx \n", ref->tag[0] );
    printf( "Index: %6llx \n", ref->index[0] );
    #endif

    // Start state machine
    while( TRUE ) {
        switch( state.next ) {
            case IDLE:
                // Increment L1 reference
                #ifdef PRINT
                printf( "Increment Reference \n" );
                #endif
                incrementL1( &state, ref );
                break;

            case QUERY_L1:
                // Query L1 cache for tag
                #ifdef PRINT
                printf( "Query L1 \n" );
                #endif
                queryL1( &state, &L1_cache );
                break;

            case QUERY_L2:
                // Query L2 cache for tag
                #ifdef PRINT
```

```
                printf( "Query L2 \n" );
                #endif
                queryL2( &state );
                break;

            case ADD_L1:
                // Add tag to L1 cache
                #ifdef PRINT
                printf( "Add L1 \n" );
                #endif
                addL1( &state, &L1_cache );
                break;

            case ADD_L2:
                // Add tag to L2 cache
                #ifdef PRINT
                printf( "Add L2 \n" );
                #endif
                addL2( &state );
                break;

            case HANDLE_WRITE:
                // Set dirty bit in L1 cache
                #ifdef PRINT
                printf( "Handle write case \n" );
                #endif
                handleWrite( &state, &L1_cache );
                break;

            case FLUSH:
                // Flush and invalidate all caches
                #ifdef PRINT
                printf( "Flush \n" );
                #endif
                flushCaches( &state );
                break;

            case TERMINATE:
                // Break out of state machine
                #ifdef PRINT
                printf( "
                    ********************************************************************************
                    n" );
                #endif
                terminate( ref );
                return;
        }
    }
}


/*
    ********************************************************************************

 * Selects correct L1 cache based on reference type
 ********************************************************************************
    */
struct cache cacheType( struct reference* ref ) {

    // Make sure setFlush is initially false
```

```
    setFlush = FALSE;

    if( ref->type == 'I' ) {
        runResults.numInst++;

        // Check number of instructions
        if( runResults.numInst % 380000 == 0 ) {
            setFlush = TRUE;
        }

        #ifdef PRINT
        printf( "L1 Instruction Cache: %c \n", ref->type );
        #endif

        return L1_instruction;

    } else if( (ref->type == 'R') || (ref->type == 'W') ) {
        if ( ref-> type == 'R' ) {
            runResults.numReads++;
        } else {
            runResults.numWrites++;
        }

        #ifdef PRINT
        printf( "L1 Data Cache: %c \n", ref->type );
        #endif

        return L1_data;

    } else {
        printf( "Unrecognized reference type" );
        return L1_instruction;
    }
}


/*
    ********************************************************************************

 * Set L1 tag and index based on iteration
 ********************************************************************************
    */
void incrementL1( struct state* state, struct reference* ref ) {

    // Increment iteration
    state->iteration++;

    // Check if there is another reference
    if( state->iteration == ref->numReferences ) {
        state->next = TERMINATE;
        return;
    }

    // Otherwise set new references
    state->L1_Index = ref->index[ state->iteration ];
    state->L1_Tag = ref->tag[ state->iteration ];

    // Transition back start of statemachine
    state->next = QUERY_L1;
    return;
```

```
}


/*
    ************************************************************************************************

 * Query L1 cache to get reference
 ************************************************************************************************
    */
void queryL1( struct state* state, struct cache* cache ) {

    // Query L1 cache
    bool hit = queryCache( state->L1_Index, state->L1_Tag, cache );

    // Transition based on result
    if( (hit == TRUE) && (state->type == 'W') ) {
        runResults.l1d_hit++;
        runResults.numWriteCycles += config.L1_hit_time;
        state->next = HANDLE_WRITE;
        return;
    }
    else if( hit == TRUE) {
        if (state-> type == 'R') {
            runResults.l1d_hit++;
            runResults.numReadCycles += config.L1_hit_time;
        } else {
            runResults.l1i_hit++;
            runResults.numInstCycles += config.L1_hit_time;
        }
        state->next = IDLE;
        return;
    }
    else {
        if (state->type == 'I') {
            runResults.l1i_miss++;
            runResults.numInstCycles += config.L1_miss_time + config.L1_hit_time + config.
                L1_transfer_cycles;

        } else if (state->type == 'W') {
            runResults.l1d_miss++;
            runResults.numWriteCycles += config.L1_miss_time + config.L1_hit_time + config
                .L1_transfer_cycles;
        } else {
            runResults.l1d_miss++;
            runResults.numReadCycles += config.L1_miss_time + config.L1_hit_time + config.
                L1_transfer_cycles;
        }
        // state->next = QUERY_L2;
        state->next = ADD_L1;
        return;
    }
}


/*
    ************************************************************************************************

 * Access L2 cache to get reference
 ************************************************************************************************
    */
```

```
void queryL2( struct state* state ) {

    // Decompose the address for the L2 cache
    struct L2_Reference L2_Ref;
    constructL2Ref( &L2_Ref, state->L1_Index, state->L1_Tag );

    // Update index and tag for L2 cache
    state->L2_Tag = L2_Ref.L2_Tag;
    state->L2_Index = L2_Ref.L2_Index;

    // Query L2 cache
    bool hit = queryCache( L2_Ref.L2_Index, L2_Ref.L2_Tag, &L2_unified );

    // Transition based on result
    if( (hit == TRUE) && (state->type == 'W') ) {
        runResults.l2_hit++;
        runResults.numWriteCycles += config.L2_hit_time;
        state->next = HANDLE_WRITE;
        return;
    } else if( hit == TRUE ) {
        if ( state->type == 'R') {
            runResults.numReadCycles += config.L2_hit_time;
        } else {
            runResults.numInstCycles += config.L2_hit_time;
        }
        runResults.l2_hit++;
        state->next = IDLE;
        return;
    } else {
        if ( state->type == 'R') {
            runResults.numReadCycles += config.L2_miss_time + config.L2_hit_time + config.
                L2_transfer_cycles;
        } else if ( state->type == 'W') {
            runResults.numWriteCycles += config.L2_miss_time + config.L2_hit_time + config
                .L2_transfer_cycles;
        } else {
            runResults.numInstCycles += config.L2_miss_time + config.L2_hit_time + config.
                L2_transfer_cycles;
        }

        runResults.l2_miss++;
        state->next = ADD_L2;
        return;
    }
}


/*
    ********************************************************************************

 * Add tag to L1 cache
 ********************************************************************************
    */
void addL1( struct state* state, struct cache* cache ) {

    // Add reference
    addCache( state->L1_Index, state->L1_Tag, cache, state->type );

    // Transition
    state->next = QUERY_L2;
```

```
    return;

}


/*
    ********************************************************************************

 * Add tag to L2 cache
 ********************************************************************************
    */
void addL2( struct state* state ) {

    // Add reference
    struct cache* cache = &L2_unified;
    addCache( state->L2_Index, state->L2_Tag, cache, state->type );

    // Transition
    if( state->type == 'W' ) {
        state->next = HANDLE_WRITE;
        return;
    } else {
        state->next = IDLE;
        return;
    }
}


/*
    ********************************************************************************

 * Set the dirty bit of tag in L1 cache
 ********************************************************************************
    */
void handleWrite( struct state* state, struct cache* cache) {

    // Set the dirty bit of given cache (L1)
    setDirty( state->L1_Index, state->L1_Tag, cache );

    // Transition
    state->next = IDLE;
    return;
}


/*
    ********************************************************************************

 * Flush the cache by invalidating all of the data
 ********************************************************************************
    */
void flushCaches( struct state* state ) {;

    // Flush the L1_Instruction cache
    flush( &L1_instruction );

    // Flush the L1_Data cache
    flush( &L1_data );

    // Flush the L2_Unified cache
```

```
    flush( &L2_unified );

    // Increment results
    runResults.flushes++;
    runResults.invalidates++;

    // Transition
    state->next = QUERY_L1;
    return;
}


/*
    ********************************************************************************

 * Frees memory used to hold references
 ********************************************************************************
    */
void terminate( struct reference* ref ) {

    // Free reference and index
    free( ref->index );
    free( ref->tag );
}
```

statemachine.h:

```
#ifndef STATEMACHINE_H
#define STATEMACHINE_H

#include <stdio.h>
#include "cache.h"
#include "print.h"


// Comment this line to disable print statements
// #define PRINT


/*
    ********************************************************************************************

 * Define states used in the state machine
 ********************************************************************************************
    */
#define IDLE                            0
#define QUERY_L1                        1
#define QUERY_L2                        2
#define ADD_L1                          3
#define ADD_L2                          4
#define HANDLE_WRITE                    5
#define FLUSH                           6
#define TERMINATE                       7


/*
    ********************************************************************************************

 * Struct to represent state and L1_Index and L1_Tag
 ********************************************************************************************
    */
 struct state {
    int next;                                   // Transition to this state
    int iteration;                              // Index of the current reference
    char type;                                  // Type of instruction
    unsigned long long L1_Tag;                  // L1 tag to access
    unsigned long long L1_Index;                // L1 index to access
    unsigned long long L2_Tag;                  // L2 tag of access
    unsigned long long L2_Index;                // L2 index to access
};


/*
    ********************************************************************************************

 * Flag for if first state should be set to FLUSH
 ********************************************************************************************
    */
bool setFlush;


/*
    ********************************************************************************************

 * Makes call to state machine
```

```
 ****************************************************************************
    */
void stateMachine( struct reference* ref );


/*
   ****************************************************************************

 * Selects correct L1 cache based on reference type
 ****************************************************************************
    */
struct cache cacheType( struct reference* ref );


/*
   ****************************************************************************

 * Set L1 tag and index based on iteration
 ****************************************************************************
    */
void incrementL1( struct state* state, struct reference* ref );


/*
   ****************************************************************************

 * Access L1 cache to get reference
 ****************************************************************************
    */
void queryL1( struct state* state, struct cache* cache );


/*
   ****************************************************************************

 * Access L2 cache to get reference
 ****************************************************************************
    */
void queryL2( struct state* state );


/*
   ****************************************************************************

 * Add tag to L1 cache
 ****************************************************************************
    */
void addL1( struct state* state, struct cache* cache );


/*
   ****************************************************************************

 * Add tag to L2 cache
 ****************************************************************************
    */
void addL2( struct state* state );


/*
```

```
    ********************************************************************************
 * Set the dirty bit of tag in L1 cache
 ********************************************************************************
    */
void handleWrite( struct state* state, struct cache* cache );


/*
    ********************************************************************************
 * Flush the cache by invalidating all of the data
 ********************************************************************************
    */
void flushCaches( struct state* state );


/*
    ********************************************************************************
 * Frees memory used to hold references
 ********************************************************************************
    */
void terminate( struct reference* ref );


#endif // STATEMACHINE_H
```

Makefile

```
# Makefile for the project

CC = gcc
CFLAGS = -I.
DEPS = cache.h configparse.h main.h print.h statemachine.h
OBJ = cache.o configparse.o main.o print.o statemachine.o
LIBS = -lm

%.o: %.c $(DEPS)
	$(CC) -O3 -Wall -Wextra -c -o $@ $< $(CFLAGS)

cacheSim.exe: $(OBJ)
	$(CC) -O3 -o $@ $^ $(CFLAGS) $(LIBS)

clean:
	rm -f *.o
	rm cacheSim.exe
```