EE524/CPTS561 PA2

OLUWAFEMI AJEIGBE

Wednesday 26th October, 2022

Introduction and Objective

Cache design choices affect the performance of a microprocessor. In the case of this assignment, we were asked to fine-tune the cache hierarchy on X86 architecture using the gem5 simulator. Also, the goal of a good cache design is pivotal in how sensible we made our design specification choices (cost, cache sizes, cache associativity.). To this end I carried out a test on five benchmark programs; 401.bzip2, 429.mcf, 456.hmmer, 458.sjeng and 470.lbm, and below are some of the data points generated after the test.

Part 2: Calculate CPI

Table 1: CPI Calculation for each Benchmark

B	L1D(kB)	L1I(kB)	L2(MB)	L1A	L2A	L2Miss	L1I Miss	L1D Miss	CPI
401.bzip2	128	128	1	2	4	0.659356	0.000001	0.013116	1.000000066
429.mcf	128	128	1	2	4	0.616319	0.000001	0.034822	1.000000062
456.hmmer	128	128	1	2	4	0.048838	0.000002	0.000482	1.000000005
458.sjeng	128	128	1	2	4	0.99414	0.000004	0.031002	1.0000001
470.lbm	128	128	1	2	4	0.997968	0.000001	0.033135	1.0000001
								Instructions	**500000000**
								Block size	**64 bytes**

Part 2.1: Calculate Cache Miss Types

Compulsory Miss: This is referred to as the cold start miss. This is the amount of misses we experience when we run a program for the first time. For this experiment, I increased the L1 cache size equally from 64kB to 512kB. I observed that the number of the L1 instruction cache misses reduced and stopped reducing when the cache size was at 256kB and this stayed the same even as I further increased the cache size to 512kB for all the five benchmark applications. At this point we get the number of compulsory misses for the application. *Please refer to Appendix for the data points.*

Capacity Miss: Making the cache a fully associative cache will make the cache have only two misses which is the compulsory misses and the capacity misses as there will not be any conflict misses if the cache is a fully associative cache.

Conflict Miss: When the cache is fully associative, there will not be any conflict misses.

$$C = C_1 + C_2 + C_3 \tag{1}$$

where:
C = total misses
C_1 = Compulsory misses
C_2 = Capacity misses
C_3 = Conflict misses

$$\begin{aligned} C_{64kB} &= C_1 + C_2 + C_3 \\ C_1 &= C_{512kB} \\ C_{64kB|512} &= C_1 + C_2 \\ C_2 &= C_{64kB|512} - C_{512kB} \\ C_3 &= C_{64kB} - C_{64kB|512} \end{aligned}$$

where:
C = total misses
C_{64kB} = number of misses when the cache size is 64kB
$C_{64kB|512}$ =number of misses when the cache size is 64kB and cache is fully associative (n=512)
C_{512kB} = number of misses when the cache size is 512kB

Part 3: Optimize CPI for each benchmark

1. Optimal Configuration for each Benchmarks. (***based on the data points in the appendix section***)

   1.1. **401.bzips**

      1.1.1. Block size: 64 bytes
      L1 Instruction cache: 256 KB
      L1 Data cache: 64 KB
      L1 Instruction and data cache associativity: 4
      L2 size: 4 MB
      L2 associativity: 4
      Number of instructions: 500000000
      CPI : **1.000000008**

   1.2. **429.mcf**

      1.2.1. Block size: 64 bytes
      L1 Instruction cache: 256 KB
      L1 Data cache: 64 KB
      L1 Instruction and data cache associativity: 4
      L2 size: 4 MB
      L2 associativity: 4
      Number of instructions: 500000000
      CPI : **1.000000036**

   1.3. **456.hmmer**

      1.3.1. Block size: 64 bytes
      L1 Instruction cache: 256 KB
      L1 Data cache: 64 KB
      L1 Instruction and data cache associativity: 4
      L2 size: 4 MB
      L2 associativity: 4
      Number of instructions: 500000000
      CPI : **1.000000002**

   1.4. **458.sjeng**

      1.4.1. Block size: 64 bytes
      L1 Instruction cache: 64 KB
      L1 Data cache: 128 KB
      L1 Instruction and data cache associativity: 2
      L2 size: 4 MB
      L2 associativity: 4
      Number of instructions: 500000000
      CPI : **1.000000094**

   1.5. **470.lbm**

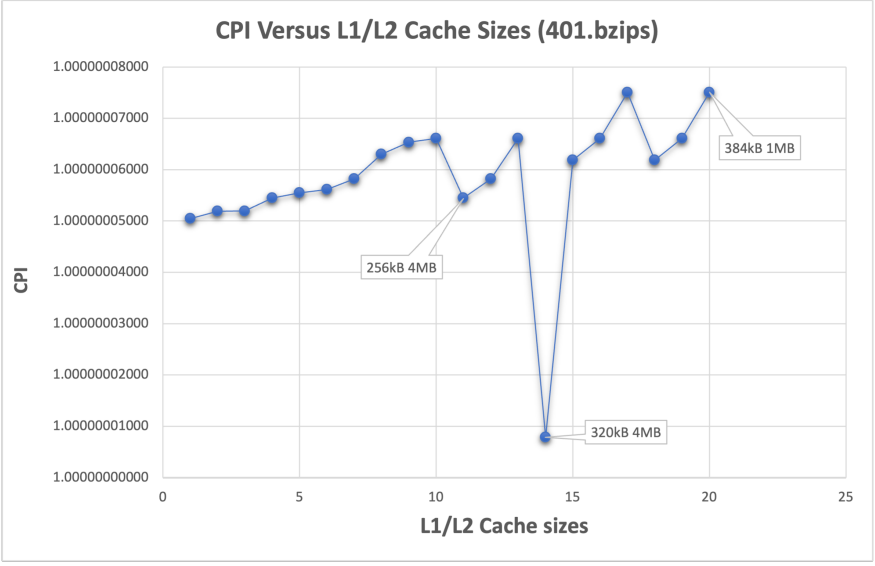
      1.5.1. Block size: 64 bytes
      L1 Instruction cache: 256 KB
      L1 Data cache: 64 KB
      L1 Instruction and data cache associativity: 4
      L2 size: 4 MB
      L2 associativity: 4
      Number of instructions: 500000000
      CPI: **1.00000010011112**

2. Reasoning for trade-offs between design choices

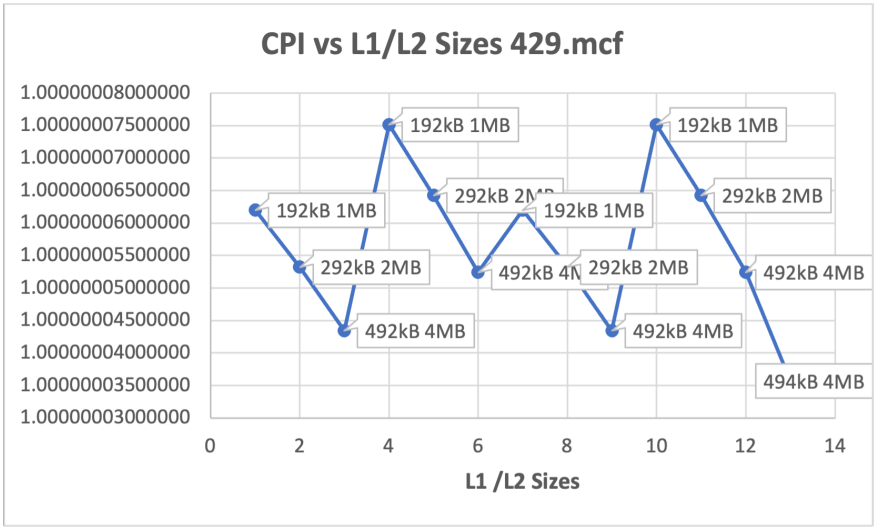
   Cycles per instruction, or CPI, is defined as the average number of CPU cycles required to retire an instruction, and therefore is an indicator of how much latency in the system affected the running application. In this experiment, I observed that the miss rate is not proportional to the increasing the cache size. It appears that as per the above stated optimal configurations, we observe that increasing the cache sizes (L1 and L2) to a point reduces the miss rate although, after a point the hit time increases which led to higher CPIs for those configurations. Also, we observe that increasing the associativity improves the CPI.

3. Graphs

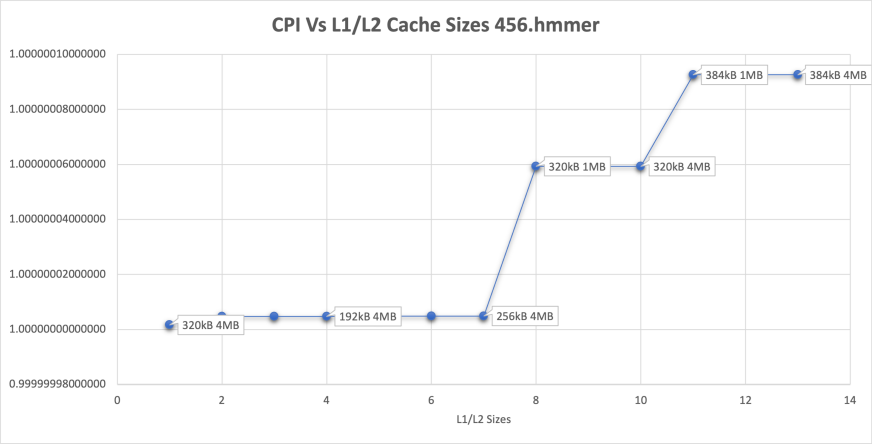
   3.1. **401.bzips**



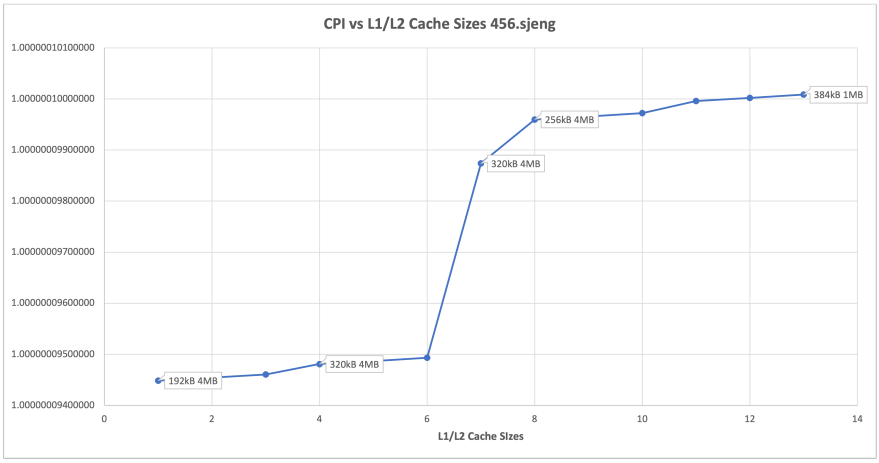
   3.2. **429.mcf**



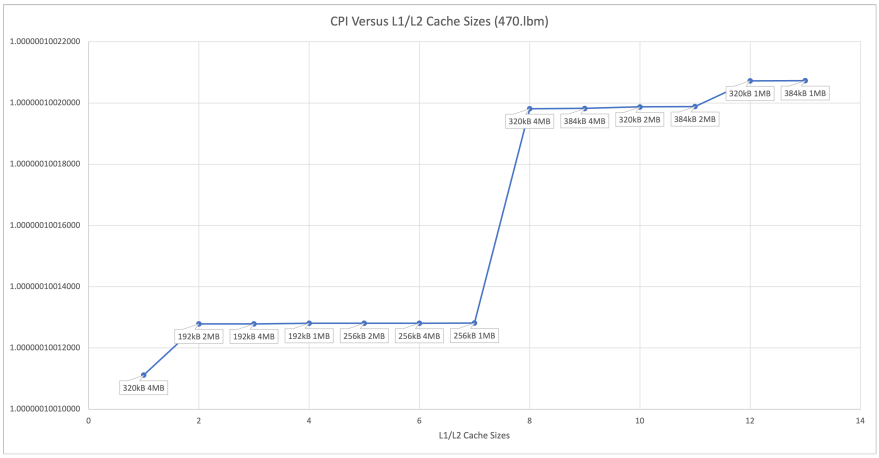
3.3. **456.hmmer**



3.4. **458.sjeng**



3.5. **470.lbm**



Part 4: Define cost function

Table 2: Cost Function

Cost Funtion	
Item	**Price/kB**
L1 Cache	$150
L2 Cache	$5

Table 3: 401.bzips

BenchMark	L1 Cache Size	L1 Cost	L2 Size(MB)	L2	L2 Cost	CPI	Total Cost
401.bzip2	320kB	$48,000	4MB	4	$20,480	**1.00000000783999**	$68,480
401.bzip2	128kB	$19,200	4MB	4	$20,480	1.00000005048363	$39,680
401.bzip2	192kB	$28,800	4MB	4	$20,480	1.00000005192567	$49,280
401.bzip2	192kB	$28,800	4MB	4	$20,480	1.00000005193127	$49,280
401.bzip2	256kB	$38,400	4MB	4	$20,480	1.00000005448827	$58,880
401.bzip2	192kB	$28,800	4MB	4	$20,480	1.00000005448937	$49,280
401.bzip2	192kB	$28,800	2MB	2	$10,240	1.00000005545807	$39,040
401.bzip2	192kB	$28,800	2MB	2	$10,240	1.00000005613427	$39,040
401.bzip2	256kB	$38,400	2MB	2	$10,240	1.00000005819027	$48,640
401.bzip2	192kB	$28,800	2MB	2	$10,240	1.00000005819137	$39,040
401.bzip2	384kB	$57,600	4MB	4	$20,480	1.00000006187345	$78,080
401.bzip2	320kB	$48,000	4MB	4	$20,480	1.00000006187475	$68,480
401.bzip2	192kB	$28,800	1MB	1	$5,120	1.00000006296427	**$33,920**
401.bzip2	192kB	$28,800	1MB	1	$5,120	1.00000006532017	$33,920
401.bzip2	256kB	$38,400	1MB	1	$5,120	1.00000006606677	$43,520
401.bzip2	192kB	$28,800	1MB	1	$5,120	1.00000006606697	$33,920
401.bzip2	384kB	$57,600	2MB	2	$10,240	1.00000006607925	$67,840
401.bzip2	320kB	$48,000	2MB	2	$10,240	1.00000006608035	$58,240
401.bzip2	384kB	$57,600	1MB	1	$5,120	1.00000007504995	$62,720
401.bzip2	320kB	$48,000	1MB	1	$5,120	1.00000007505005	$53,120

Table 4: 429.mcf

BenchMark	L1 Cache Size	L1 Cost	L2 Size(MB)	L2	L2 Cost	CPI	Total Cost
429.mcf	192kB	$28,800	1MB	1	$5,120	1.00000006198013	**$33,920**
429.mcf	192kB	$28,800	1MB	1	$5,120	1.00000006198023	$33,920
429.mcf	192kB	$28,800	1MB	1	$5,120	1.00000007513534	$33,920
429.mcf	192kB	$28,800	1MB	1	$5,120	1.00000007513544	$33,920
429.mcf	292kB	$43,800	2MB	2	$10,240	1.00000005321513	$54,040
429.mcf	292kB	$43,800	2MB	2	$10,240	1.00000005321563	$54,040
429.mcf	292kB	$43,800	2MB	2	$10,240	1.00000006425574	$54,040
429.mcf	292kB	$43,800	2MB	2	$10,240	1.00000006425634	$54,040
429.mcf	492kB	$73,800	4MB	4	$20,480	1.00000004342553	$94,280
429.mcf	492kB	$73,800	4MB	4	$20,480	1.00000004342573	$94,280
429.mcf	492kB	$73,800	4MB	4	$20,480	1.00000005240254	$94,280
429.mcf	492kB	$73,800	4MB	4	$20,480	1.00000005240284	$94,280
429.mcf	494kB	$74,100	4MB	4	$20,480	**1.00000003567171**	$94,580

Table 5: 456.hmmer

BenchMark	L1 Cache Size	L1 Cost	L2 Size(MB)	L2	L2 Cost	CPI	Total Cost
456.hmmer	192kB	$28,800	1MB	1	$5,120	1.00000000474829	**$33,920**
456.hmmer	192kB	$28,800	2MB	2	$10,240	1.00000000474829	$39,040
456.hmmer	256kB	$38,400	1MB	1	$5,120	1.00000000488864	$43,520
456.hmmer	256kB	$38,400	2MB	2	$10,240	1.00000000488864	$48,640
456.hmmer	192kB	$28,800	4MB	4	$20,480	1.00000000474829	$49,280
456.hmmer	320kB	$48,000	1MB	1	$5,120	1.00000005931537	$53,120
456.hmmer	320kB	$48,000	2MB	2	$10,240	1.00000005931537	$58,240
456.hmmer	256kB	$38,400	4MB	4	$20,480	1.00000000488864	$58,880
456.hmmer	384kB	$57,600	1MB	1	$5,120	1.00000009260642	$62,720
456.hmmer	384kB	$57,600	2MB	2	$10,240	1.00000009260642	$67,840
456.hmmer	320kB	$48,000	4MB	4	$20,480	**1.00000000167415**	$68,480
456.hmmer	320kB	$48,000	4MB	4	$20,480	1.00000005931537	$68,480
456.hmmer	384kB	$57,600	4MB	4	$20,480	1.00000009260642	$78,080

Table 6: 458.sjeng

BenchMark	L1 Cache Size	L1 Cost	L2 Size(MB)	L2	L2 Cost	CPI	Total Cost
458.sjeng	192kB	$28,800	4MB	4	$20,480	**1.00000009448515**	$49,280
458.sjeng	192kB	$28,800	2MB	2	$10,240	1.00000009454385	$39,040
458.sjeng	192kB	$28,800	1MB	1	$5,120	1.00000009460665	**$33,920**
458.sjeng	320kB	$48,000	4MB	4	$20,480	1.00000009480923	$68,480
458.sjeng	320kB	$48,000	2MB	2	$10,240	1.00000009486833	$58,240
458.sjeng	320kB	$48,000	1MB	1	$5,120	1.00000009493313	$53,120
458.sjeng	320kB	$48,000	4MB	4	$20,480	1.00000009873771	$68,480
458.sjeng	256kB	$38,400	4MB	4	$20,480	1.00000009960026	$58,880
458.sjeng	256kB	$38,400	2MB	2	$10,240	1.00000009966056	$48,640
458.sjeng	256kB	$38,400	1MB	1	$5,120	1.00000009972406	$43,520
458.sjeng	384kB	$57,600	4MB	4	$20,480	1.00000009996074	$78,080
458.sjeng	384kB	$57,600	2MB	2	$10,240	1.00000010002144	$67,840
458.sjeng	384kB	$57,600	1MB	1	$5,120	1.00000010008684	$62,720

Table 7: 470.lbm

BenchMark	L1 Cache Size	L1 Cost	L2 Size(MB)	L2	L2 Cost	CPI	Total Cost
470.lbm	320kB	$48,000	4MB	4	$20,480	**1.00000010011112**	$68,480
470.lbm	192kB	$28,800	2MB	2	$10,240	1.00000010012786	$39,040
470.lbm	192kB	$28,800	4MB	4	$20,480	1.00000010012786	$49,280
470.lbm	192kB	$28,800	1MB	1	$5,120	**1.00000010012806**	**$33,920**
470.lbm	256kB	$38,400	2MB	2	$10,240	1.00000010012806	$48,640
470.lbm	256kB	$38,400	4MB	4	$20,480	1.00000010012806	$58,880
470.lbm	256kB	$38,400	1MB	1	$5,120	1.00000010012816	$43,520
470.lbm	320kB	$48,000	4MB	4	$20,480	1.00000010019813	$68,480
470.lbm	384kB	$57,600	4MB	4	$20,480	1.00000010019823	$78,080
470.lbm	320kB	$48,000	2MB	2	$10,240	1.00000010019873	$58,240
470.lbm	384kB	$57,600	2MB	2	$10,240	1.00000010019883	$67,840
470.lbm	320kB	$48,000	1MB	1	$5,120	1.00000010020723	$53,120
470.lbm	384kB	$57,600	1MB	1	$5,120	1.00000010020733	$62,720

Part 5: Optimize caches for performance/cost

1. Evaluation Function
   As per the cost function in table 2 above, tables 3-7 above shows the various cost in dollars needed to setup a particular configuration for each of the five benchmarks. Looking at these data points, it appears that higher cost does not necessarily give better CPI performance and this is due to the fact that larger cache sizes will still give high miss rates. I propose that a point at which we have a reasonable CPI and minimal cost defines an optimal point. At which point, we do not spend too much money on setting up a processor configuration that will render a not-too-good CPI performance. I will identify the optimal design fo each benchmark next.

2. Optimal Design for each Benchmark

$$\text{Average}\{\text{CPI} * \text{Total cost}\} \tag{2}$$

   2.1. **401.bzips**
   Cost: $49,280
   CPI : 1.00000005192567

   2.2. **429.mcf**
   Cost: $54,040
   CPI : 1.00000006425634

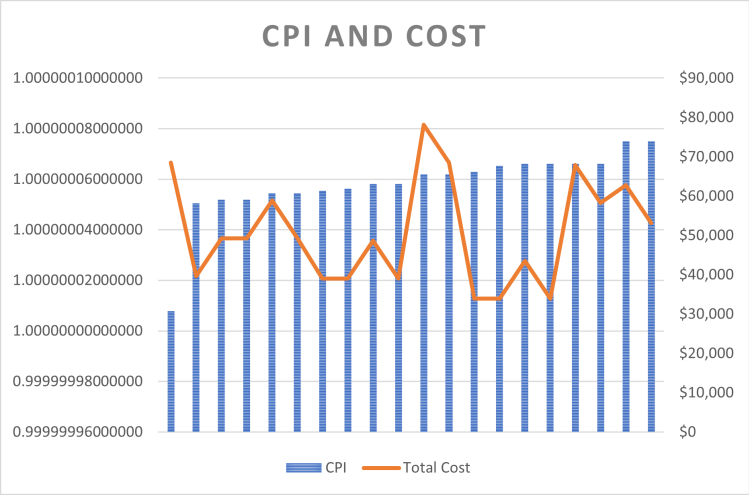
   2.3. **456.hmmer**
   Cost: $53,120
   CPI : 1.00000005931537

   2.4. **458.sjeng**
   Cost: $53,120
   CPI : 1.00000009493313

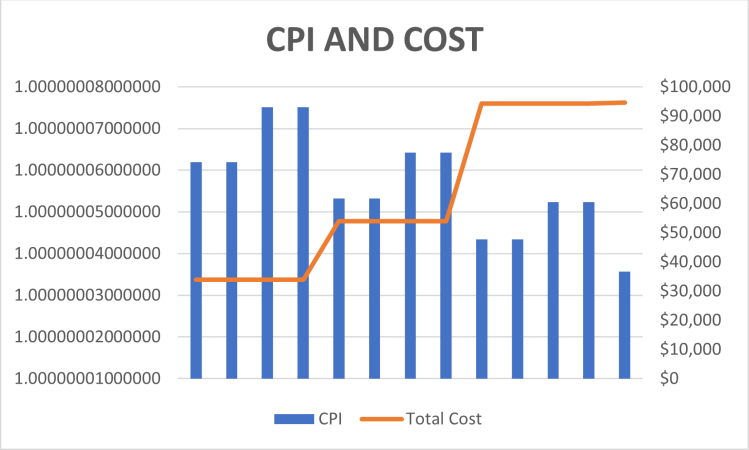
   2.5. **470.lbm**
   Cost: $53,120
   CPI : 1.00000010020723

3. Graphs

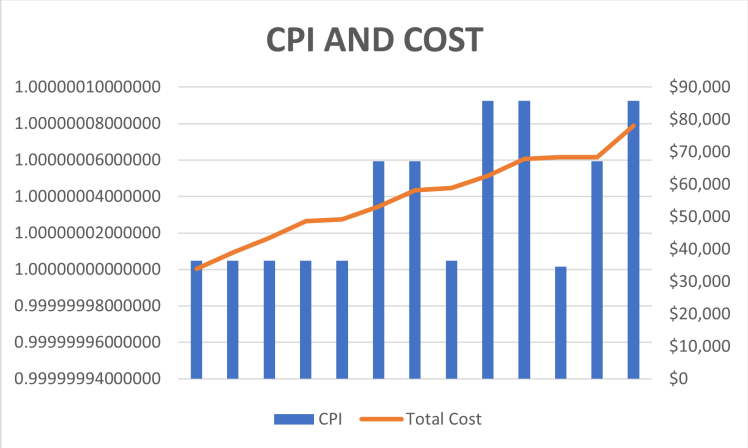
3.1. **401.bzips**



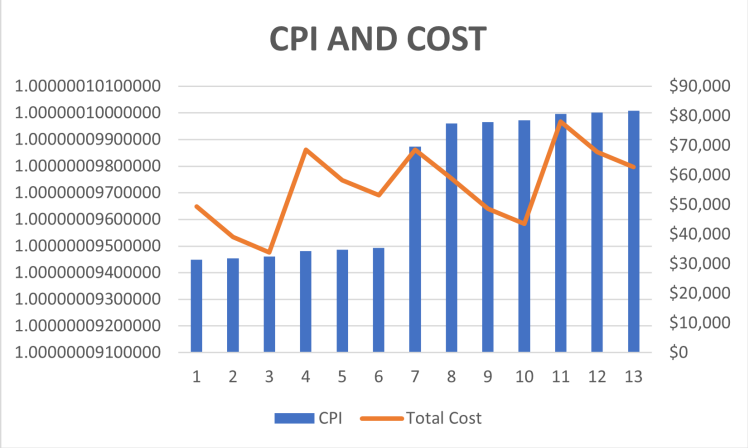
3.2. **429.mcf**



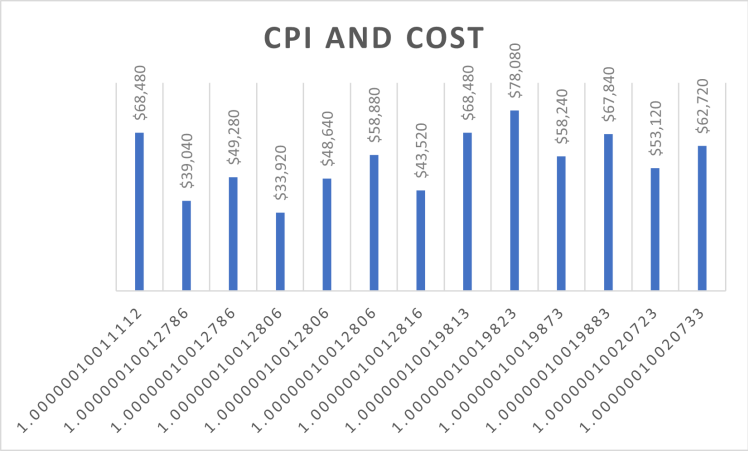
3.3. **456.hmmer**



3.4. **458.sjeng**



3.5. **470.lbm**



Appendix

Table 8: 401.bzips

L1 (kB)	L1A	L2 Miss	L1i Miss	L1D Miss	L1 T
64	4	645425	1002	2751083	2752085
64	8	642249	994	2694004	2694998
64	16	640039	987	2653053	2654040
64	512	639954	987	2551540	2552527
128	4	645325	988	2333259	2334247
128	8	642129	987	2282183	2283170
128	16	639956	987	2255590	2256577
128	512	639854	987	2166883	2167870
256	4	645288	987	1938952	1939939
256	8	642062	987	1899311	1900298
256	16	639864	987	1882269	1883256
256	512	639721	987	1804794	1805781
512	4	645827	987	1499335	1500322
512	8	641911	987	1495729	1496716
512	16	639553	987	1497074	1498061
512	512	639256	987	1425894	1426881

Table 9: 470.lbm

L1 (kB)	L1A	L2 Miss	L1i Miss	L1D Miss	L1 T
64	4	7147869	548	7163883	7164431
64	8	7147868	544	7163952	7164496
64	16	7147867	537	7163953	7164490

Table 10: 429.mcf

L1 (kB)	L1A	L2 Miss	L1i Miss	L1D Miss	L1 T
64	4	6303526	682	14698959	14699641
64	8	6255287	680	14627141	14627821
64	16	6252702	674	14659513	14660187
64	512	6354835	674	14767658	14768332
128	4	6303283	673	12503105	12503778
128	8	6255139	674	12431709	12432383
128	16	6252478	673	12358624	12359297
128	512	6354581	673	12216185	12216858
256	4	6309463	673	9624836	9625509
256	8	6259279	673	9220764	9221437
256	16	6254979	673	8987995	8988668
256	512	6351611	673	8842288	8842961
512	4	6317386	673	7949032	7949705
512	8	6285602	673	7907624	7908297
512	16	6279569	673	7867150	7867823
512	512	6358230	673	7821524	7822197

Table 11: 456.hmmer

L1 (kB)	L1A	L2 Miss	L1i Miss	L1D Miss	L1 T
64	4	6083	1545	480898	482443
64	8	6062	1507	484747	486254
64	16	6062	1502	485972	487474
64	512	6062	1494	487320	488814
128	4	6062	1416	362980	364396
128	8	6062	1406	367448	368854
128	16	6062	1399	373579	374978
128	512	6062	1385	379826	381211
256	4	6062	1375	136092	137467
256	8	6062	1371	105422	106793
256	16	6062	1371	84072	85443
256	512	6062	1371	94805	96176
512	4	6062	1371	4525	5896
512	8	6062	1371	4346	5717
512	16	6062	1371	4324	5695
512	512	6062	1371	4301	5672

Table 12: 458.sjeng

L1 (kB)	L1A	L2 Miss	L1i Miss	L1D Miss	L1 T
64	4	8376802	989847	8746760	9736607
64	8	8376216	653758	8601013	9254771
64	16	8375985	390878	8520253	8911131
64	512	8375638	276484	8502790	8779274
128	4	8376182	90731	8497795	8588526
128	8	8375604	3631	8433543	8437174
128	16	8375427	2631	8418680	8421311
128	512	8375064	1791	8413307	8415098
256	4	8375951	1886	8402279	8404165
256	8	8375451	1599	8398519	8400118
256	16	8375338	1599	8394021	8395620
256	512	8375050	1599	8393033	8394632
512	4	8375846	1599	8385884	8387483
512	8	8375451	1599	8384755	8386354
512	16	8375344	1599	8384214	8385813
512	512	8375016	1599	8383766	8385365

Table 13: 401.bzips

L1D(kB)	L1I(kB)	L2(MB)	L1A	L2A	L2 Miss	L1I Miss	L1D Miss	CPI
64	256	4	4	4	7.7606%	0.0001%	0.7938%	1.000000008
512	64	4	4	4	72.2655%	0.0001%	0.9865%	1.000000072
256	128	4	2	4	61.7580%	0.0001%	1.1544%	1.000000062
256	64	4	2	4	61.7593%	0.0001%	1.1544%	1.000000062
256	128	2	2	4	65.9638%	0.0001%	1.1544%	1.000000066
256	64	2	2	4	65.9649%	0.0001%	1.1544%	1.000000066
256	128	1	2	4	74.9345%	0.0001%	1.1544%	1.000000075
256	64	1	2	4	74.9346%	0.0001%	1.1544%	1.000000075
128	128	4	2	4	54.3571%	0.0001%	1.3116%	1.000000054
128	64	4	2	4	54.3582%	0.0001%	1.3116%	1.000000054
128	128	2	2	4	58.0591%	0.0001%	1.3116%	1.000000058
128	64	2	2	4	58.0602%	0.0001%	1.3116%	1.000000058
128	128	1	2	4	65.9356%	0.0001%	1.3116%	1.000000066
128	64	1	2	4	65.9358%	0.0001%	1.3116%	1.000000066
128	64	4	1	1	51.7880%	0.0001%	1.3766%	1.000000052
128	64	4	1	4	51.7936%	0.0001%	1.3766%	1.000000052
128	64	2	1	4	55.3204%	0.0001%	1.3766%	1.000000055
128	64	2	1	1	55.9966%	0.0001%	1.3766%	1.000000056
128	64	1	1	4	62.8266%	0.0001%	1.3766%	1.000000063
128	64	1	1	1	65.1825%	0.0001%	1.3766%	1.000000065
64	64	4	4	4	50.3420%	0.0001%	1.4162%	1.00000005

Table 14: 429.mcf

L1D(kB)	L1I(kB)	L2(MB)	L1A	L2A	L2 Miss	L1I Miss	L1D Miss	CPI
128	64	1	2	4	61.632%	0.0001%	3.4822%	1.000000062
128	64	2	2	4	52.867%	0.0001%	3.4822%	1.000000053
128	64	4	2	4	43.078%	0.0001%	3.4822%	1.000000043
256	64	1	2	4	74.848%	0.0001%	2.8783%	1.000000075
256	64	2	2	4	63.969%	0.0001%	2.8783%	1.000000064
256	64	4	2	4	52.115%	0.0001%	2.8783%	1.000000052
128	128	1	2	4	61.632%	0.0001%	3.4822%	1.000000062
128	128	2	2	4	52.867%	0.0001%	3.4822%	1.000000053
128	128	4	2	4	43.077%	0.0001%	3.4822%	1.000000043
256	128	1	2	4	74.848%	0.0001%	2.8783%	1.000000075
256	128	2	2	4	63.968%	0.0001%	2.8783%	1.000000064
256	128	4	2	4	52.115%	0.0001%	2.8783%	1.000000052
64	256	4	4	4	35.246%	0.0001%	4.2560%	1.000000036

Table 15: 456.hmmer

L1D(kB)	L1I(kB)	L2(MB)	L1A	L2A	L2 Miss	L1I Miss	L1D Miss	CPI
128	64	1	2	4	4.74%	0.0007%	0.0482%	1.000000005
128	64	2	2	4	4.74%	0.0007%	0.0482%	1.000000005
128	64	4	2	4	4.74%	0.0007%	0.0482%	1.000000005
256	64	1	2	4	59.32%	0.0007%	0.0020%	1.000000059
256	64	2	2	4	59.32%	0.0007%	0.0020%	1.000000059
256	64	4	2	4	59.32%	0.0007%	0.0020%	1.000000059
64	256	4	4	4	1.66%	0.0002%	0.1433%	1.000000002
128	128	1	2	4	4.88%	0.0002%	0.0482%	1.000000005
128	128	2	2	4	4.88%	0.0002%	0.0482%	1.000000005
128	128	4	2	4	4.88%	0.0002%	0.0482%	1.000000005
256	128	1	2	4	92.61%	0.0002%	0.0020%	1.000000093
256	128	2	2	4	92.61%	0.0002%	0.0020%	1.000000093
256	128	4	2	4	92.61%	0.0002%	0.0020%	1.000000093

Table 16: 458.sjeng

L1D(kB)	L1I(kB)	L2(MB)	L1A	L2A	L2 Miss	L1I Miss	L1D Miss	CPI
128	64	4	2	4	94.1683%	0.0683%	3.1002%	1.000000094
128	64	2	2	4	94.2270%	0.0683%	3.1002%	1.000000095
128	64	1	2	4	94.2898%	0.0683%	3.1002%	1.000000095
256	64	4	2	4	94.4935%	0.0683%	3.0890%	1.000000095
256	64	2	2	4	94.5526%	0.0683%	3.0890%	1.000000095
256	64	1	2	4	94.6174%	0.0683%	3.0890%	1.000000095
64	256	4	4	4	98.4249%	0.0002%	3.1279%	1.000000099
128	128	4	2	4	99.2902%	0.0004%	3.1002%	1.0000001
128	128	2	2	4	99.3505%	0.0004%	3.1002%	1.0000001
128	128	1	2	4	99.4140%	0.0004%	3.1002%	1.0000001
256	128	4	2	4	99.6518%	0.0004%	3.0890%	1.0000001
256	128	2	2	4	99.7125%	0.0004%	3.0890%	1.0000001
256	128	1	2	4	99.7779%	0.0004%	3.0890%	1.0000001

Table 17: 470.lbm

L1D(kB)	L1I(kB)	L2(MB)	L1A	L2A	L2 Miss	L1I Miss	L1D Miss	CPI
128	64	1	2	4	99.7967%	0.0001%	3.3135%	1.0000001
128	64	2	2	4	99.7965%	0.0001%	3.3135%	1.0000001
128	64	4	2	4	99.7965%	0.0001%	3.3135%	1.0000001
256	64	1	2	4	99.8761%	0.0001%	3.3112%	1.0000001
256	64	2	2	4	99.8676%	0.0001%	3.3112%	1.0000001
256	64	4	2	4	99.8670%	0.0001%	3.3112%	1.0000001
128	128	1	2	4	99.7968%	0.0001%	3.3135%	1.0000001
128	128	2	2	4	99.7967%	0.0001%	3.3135%	1.0000001
128	128	4	2	4	99.7967%	0.0001%	3.3135%	1.0000001
256	128	1	2	4	99.8762%	0.0001%	3.3112%	1.0000001
256	128	2	2	4	99.8677%	0.0001%	3.3112%	1.0000001
256	128	4	2	4	99.8671%	0.0001%	3.3112%	1.0000001
64	256	4	4	4	99.7797%	0.0001%	3.3141%	1.0000001