COMPUTER ARCHITECHTURE

Assignment 3

CACHES

SUBMISSION BY:

Siddharth Chauhan (IMT2021046)

Kushal Partani (IMT2021062)

Arjun Subhedar (IMT2021069)

INSTRUCTIONS TO RUN THE PROGRAMS:

1. **On the terminal, type the command ‘python3 filename.py’.**
2. **As we have made use of file handling to read the trace files, the trace files that are to be tested should be in the same directory (path) as that of the codes that are being executed.**
3. **Once the code is run, a table displaying results with be printed and five additional .txt files will be created which would contain the cache implementation of the five traces that are given.**

Question-1

a) Implementing a 4-way set associative cache of size 512kb. Block size: 4 bytes. Assume a 32-bit address.

4waySA

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483893	61	23110	60711	515683	93.8354
swim	280825	0	17649	48228	303193	92.6225
twolf	476844	1	5747	17009	482824	98.7615
mcf	7508	654116	16454	210	727230	1.0324
gzip	320883	157792	1211	2475	481044	66.7055

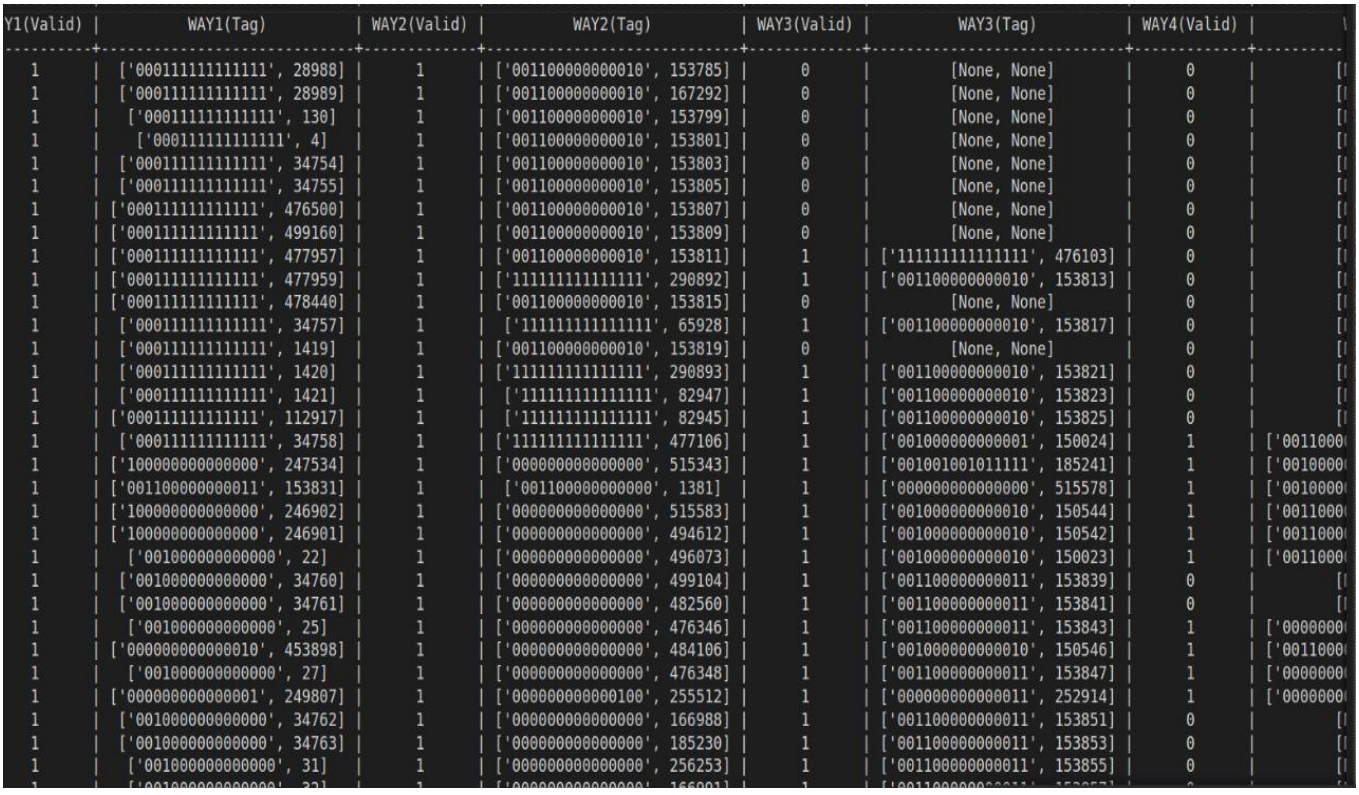
Here we are implementing a 4-way cache of 512kbytes. The given block size is 4 bytes and we are assuming a 32-bit address as given in the question.

Here the number of tag bits are 15 bits, the number of index bits are 15 bits and offset are 2 bits.

So, the total number of rows in the cache are (2^15)-1 i.e., 32767.The given number of rows in the table is number of rows partially or completely filled in the cache. In the table empty stands for the number of blocks unfilled in the partially filled rows. This number is same as number of 0 valid bit in the partially filled row in the cache.

As we can see in the table that the hit rate varies significantly from one trace to another. In ‘gcc.trace’ the hit rate is nearly 93% and in ‘mcf.trace’ the hit rate is nearly 1%.

Here we have implemented the LRU (Least Recently Used) replacement policy as we thought that would be the best policy to increase the hit rate. We have implemented LRU by keeping a PC counter which we will update every time we get a hit and when there is a miss and the cache size is full, we evicted the value having the least PC counter which was the Least Recently used. The evictions in the table denote the same.



Y1(Valid)	WAY1(Tag)	WAY2(Valid)	WAY2(Tag)	WAY3(Valid)	WAY3(Tag)	WAY4(Valid)	\
1	['000111111111111', 28988]	1	['001100000000010', 153785]	0	[None, None]	0	[I
1	['000111111111111', 28989]	1	['001100000000010', 167292]	0	[None, None]	0	[I
1	['000111111111111', 130]	1	['001100000000010', 153799]	0	[None, None]	0	[I
1	['000111111111111', 4]	1	['001100000000010', 153801]	0	[None, None]	0	[I
1	['000111111111111', 34754]	1	['001100000000010', 153803]	0	[None, None]	0	[I
1	['000111111111111', 34755]	1	['001100000000010', 153805]	0	[None, None]	0	[I
1	['000111111111111', 476500]	1	['001100000000010', 153807]	0	[None, None]	0	[I
1	['000111111111111', 499160]	1	['001100000000010', 153809]	0	[None, None]	0	[I
1	['000111111111111', 477957]	1	['001100000000010', 153811]	1	['111111111111111', 476103]	0	[I
1	['000111111111111', 477959]	1	['111111111111111', 290892]	1	['001100000000010', 153813]	0	[I
1	['000111111111111', 478440]	1	['001100000000010', 153815]	0	[None, None]	0	[I
1	['000111111111111', 34757]	1	['111111111111111', 65928]	1	['001100000000010', 153817]	0	[I
1	['000111111111111', 1419]	1	['001100000000010', 153819]	0	[None, None]	0	[I
1	['000111111111111', 1420]	1	['111111111111111', 290893]	1	['001100000000010', 153821]	0	[I
1	['000111111111111', 1421]	1	['111111111111111', 82947]	1	['001100000000010', 153823]	0	[I
1	['000111111111111', 112917]	1	['111111111111111', 82945]	1	['001100000000010', 153825]	0	[I
1	['000111111111111', 34758]	1	['111111111111111', 477106]	1	['001000000000001', 150024]	1	['0011000(
1	['100000000000000', 247534]	1	['000000000000000', 515343]	1	['001001001011111', 185241]	1	['0010000(
1	['001100000000011', 153831]	1	['001100000000000', 1381]	1	['000000000000000', 515578]	1	['0010000(
1	['100000000000000', 246902]	1	['000000000000000', 515583]	1	['001000000000010', 150544]	1	['0011000(
1	['100000000000000', 246901]	1	['000000000000000', 494612]	1	['001000000000010', 150542]	1	['0011000(
1	['001000000000000', 22]	1	['000000000000000', 496073]	1	['001000000000010', 150023]	1	['0011000(
1	['001000000000000', 34760]	1	['000000000000000', 499104]	1	['001100000000011', 153839]	0	[I
1	['001000000000000', 34761]	1	['000000000000000', 482560]	1	['001100000000011', 153841]	0	[I
1	['001000000000000', 25]	1	['000000000000000', 476346]	1	['001100000000011', 153843]	1	['0000000(
1	['000000000000010', 453898]	1	['000000000000000', 484106]	1	['001000000000010', 150546]	1	['0011000(
1	['001000000000000', 27]	1	['000000000000000', 476348]	1	['001100000000011', 153847]	1	['0000000(
1	['000000000000001', 249807]	1	['000000000000100', 255512]	1	['000000000000011', 252914]	1	['0000000(
1	['001000000000000', 34762]	1	['000000000000000', 166988]	1	['001100000000011', 153851]	0	[I
1	['001000000000000', 34763]	1	['000000000000000', 185230]	1	['001100000000011', 153853]	0	[I
1	['001000000000000', 31]	1	['000000000000000', 256253]	1	['001100000000011', 153855]	0	[I

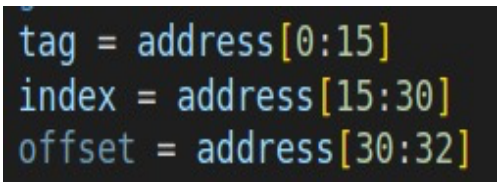
This is the 4-way cache of the trace file ‘gcc’.Here we show the valid bit, tag bits and the PC counter which we are using for the replacement policy.

The Nones here denote the garbage value as the valid bit is 0 and these are empty slots in the table.

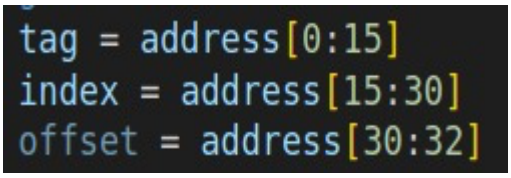
b) Increase the cache size to 2048kB and repeat the experiment. Note the change in hit/miss rates.

Trace File	Evictions (512kB)	Hit-Rate (512kB)	Evictions (2048kB)	Hit-Rate (2048kB)
gcc	61	93.8354%	38	93.8356%
swim	0	92.6225%	0	92.6225%
twolf	1	98.7615%	0	98.7615%
mcf	654116	1.0324%	457508	1.0324%
gzip	157792	66.7055%	153184	66.7055%

Here we can see that as we increase the size of the cache the evictions reduce as the index bits in 2048kB is 17 and there is a larger space to store the values. The hit rate tends to increase. Like it increases in gcc by a small value and in the rest of the trace files the hit rate is not increasing till the 4 decimals.



a)



b)

‘a’ stands for the division of the bits into index, tag and offset in 2048kB cache

‘b’ stands for the division of the bits into index, tag and offset in

512kB cache.

C) Keeping the cache size at 512kB, vary the block size from 1 byte, 4 bytes, 8 bytes, 16 bytes. Note that the number of cache lines will reduce, if you increase the block size keeping the cache size same.

Now we are keeping the size of the cache and number of ways constant I.e.,512kB and 4 ways.

We are changing the block size.

As we change the block size, we will be changing the offset, tag and index bits.

In 4-way cache when the block size is 1 byte the number of tag bits is 16 and the number of index bits are also 16.

And as the block size is 1 byte the number of bits in offset is 0.

When the block size is 2 byte the number of tag bits is 15 and number of index bits is also 16 and the byte offset is 1 bit.

We calculated the byte offset by (log (block size) to the base 2)

When the block size is 4 byte the number of tag bits is 15 and the number if index bits is 15 bits and the byte offset is 2 bits.

When the block size is 8 byte the number of tag bits is 15 and the number of index bits is 14 bits and the byte offset is 3 bits.

Here the byte offset is divided into block offset and byte offset I.e., the block offset is 1 bit and the byte offset is 2 bits.

When the block size is 16 byte the number of tag bits is 15, the number of index bits is 13 bits and the number of the byte offset bits are 4 bits.

These bytes offset bits include block offset and byte offset. The block offset consists of 2 bits and the byte offset consists of 2 bits.

4waySA_1byte

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	480610	58	26475	70885	515683	93.1987
swim	280588	0	17896	48979	303193	92.5444
twolf	475470	0	7118	21118	482824	98.4769
mcf	7451	654112	16515	393	727230	1.0246
gzip	320875	157792	1219	2499	481044	66.7039

4waySA_2byte

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	482806	61	24244	64160	515683	93.6246
swim	280737	0	17745	48524	303193	92.5935
twolf	476358	1	6230	18455	482824	98.6608
mcf	7481	654114	16483	297	727230	1.0287
gzip	320876	157792	1218	2496	481044	66.7041

4waySA_4byte

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483893	61	23110	60711	515683	93.8354
swim	280825	0	17649	48228	303193	92.6225
twolf	476844	1	5747	17009	482824	98.7615
mcf	7508	654116	16454	210	727230	1.0324
gzip	320883	157792	1211	2475	481044	66.7055

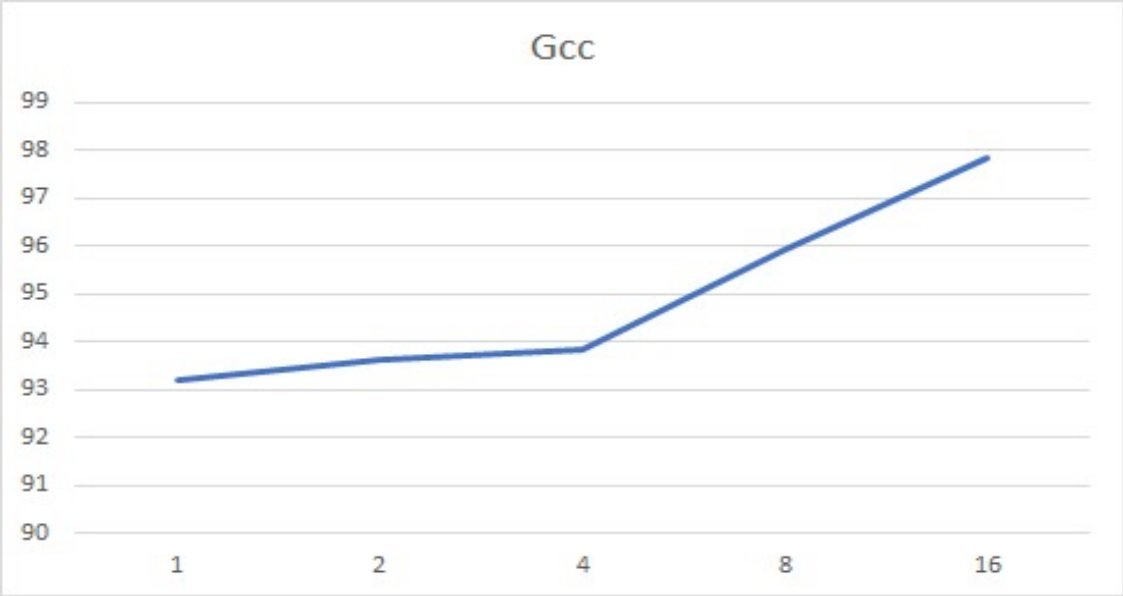
4waySA_8byte

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	494676	66	14114	35515	515683	95.9264
swim	283377	0	14995	40164	303193	93.4642
twolf	477319	1	5232	15424	482824	98.8598
mcf	7551	654143	16384	0	727230	1.0383
gzip	320891	157793	1201	2444	481044	66.7072

4waySA_16byte

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	504465	67	7202	17657	515683	97.8246
swim	291770	0	7934	20313	303193	96.2324
twolf	479869	1	2749	8042	482824	99.388
mcf	367273	327189	8192	0	727230	50.503
gzip	321268	157797	822	1309	481044	66.7856

Block Size	gcc.trace
1	93.1987
2	93.6246
4	93.8354
8	95.9264
16	97.8246

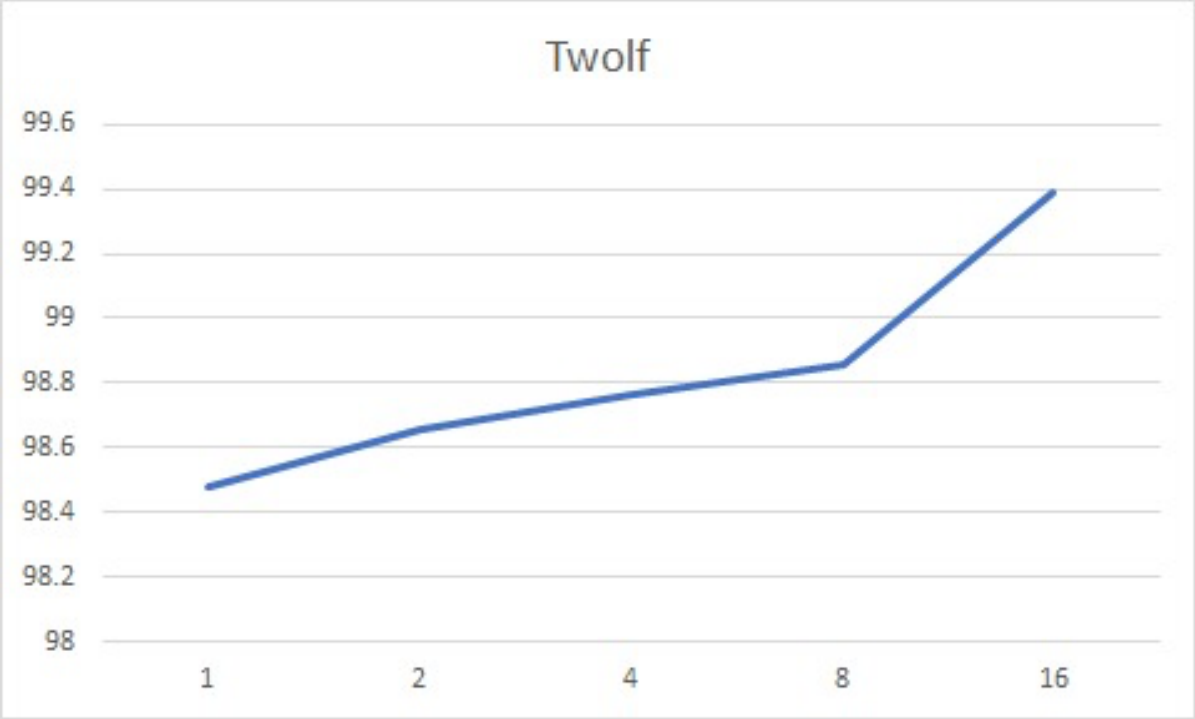


This is the variation of the hit rate in the trace Gcc when we change the blocksize.The given hit rate is in percentage.

Block Size	swim.trace
1	92.5444
2	92.5935
4	92.6225
8	93.4642
16	96.2324

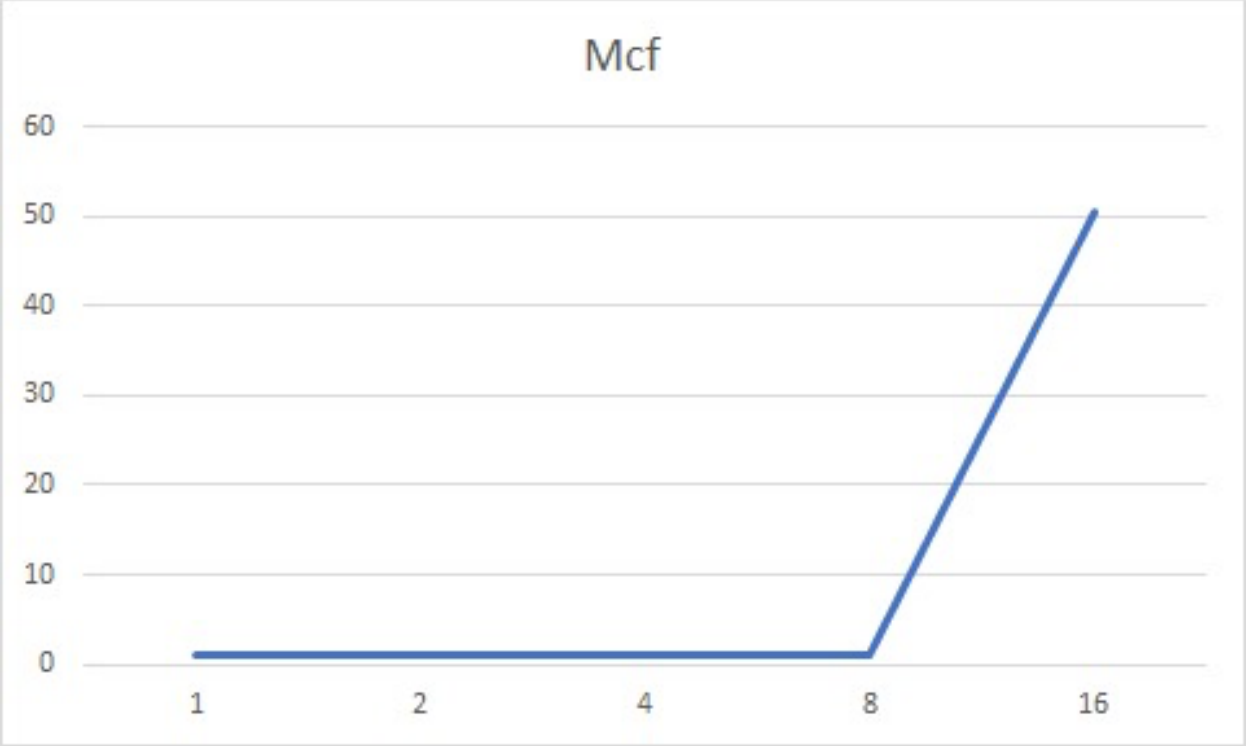
This is the variation of the hit rate in the trace Swim when we change the block size.The given hit rate is in percentage.

Block Size	twolf.trace
1	98.4769
2	98.6603
4	98.7615
8	98.8598
16	99.3880



This is the variation of the hit rate in the trace Twolf when we change the block size.The given hit rate is in percentage.

Block Size	mcf.trace
1	1.0246
2	1.0287
4	1.0324
8	1.0383
16	50.5030



This is the variation of the hit rate in the trace Mcf when we change the block size.The given hit rate is in percentage.

Block Size	gzip.trace
1	66.7039
2	66.7041
4	66.7055
8	66.7072
16	66.7856

This is the variation of the hit rate in the trace Gzip when we change the block size.The given hit rate is in percentage.

d) Vary the associativity from 1-way to 16-way, for a fixed cache size of 512kB

We are keeping the block size constant I.e., 4 bytes and the cache size as 512kB.

We are changing the associativity of the cache from 1 way to 16 way.

1waySA

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483878	213	29303	27014	515683	93.8325
swim	280825	8	20263	18166	303193	92.6225
twolf	476841	7	5932	5888	482824	98.7608
mcf	7507	654117	32838	70	727230	1.0323
gzip	320883	157794	1595	823	481044	66.7055

2waySA

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483878	213	29303	27014	515683	93.8325
swim	280825	8	20263	18166	303193	92.6225
twolf	476841	7	5932	5888	482824	98.7608
mcf	7507	654117	32838	70	727230	1.0323
gzip	320883	157794	1595	823	481044	66.7055

4waySA

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483893	61	23110	60711	515683	93.8354
swim	280825	0	17649	48228	303193	92.6225
twolf	476844	1	5747	17009	482824	98.7615
mcf	7508	654116	16454	210	727230	1.0324
gzip	320883	157792	1211	2475	481044	66.7055

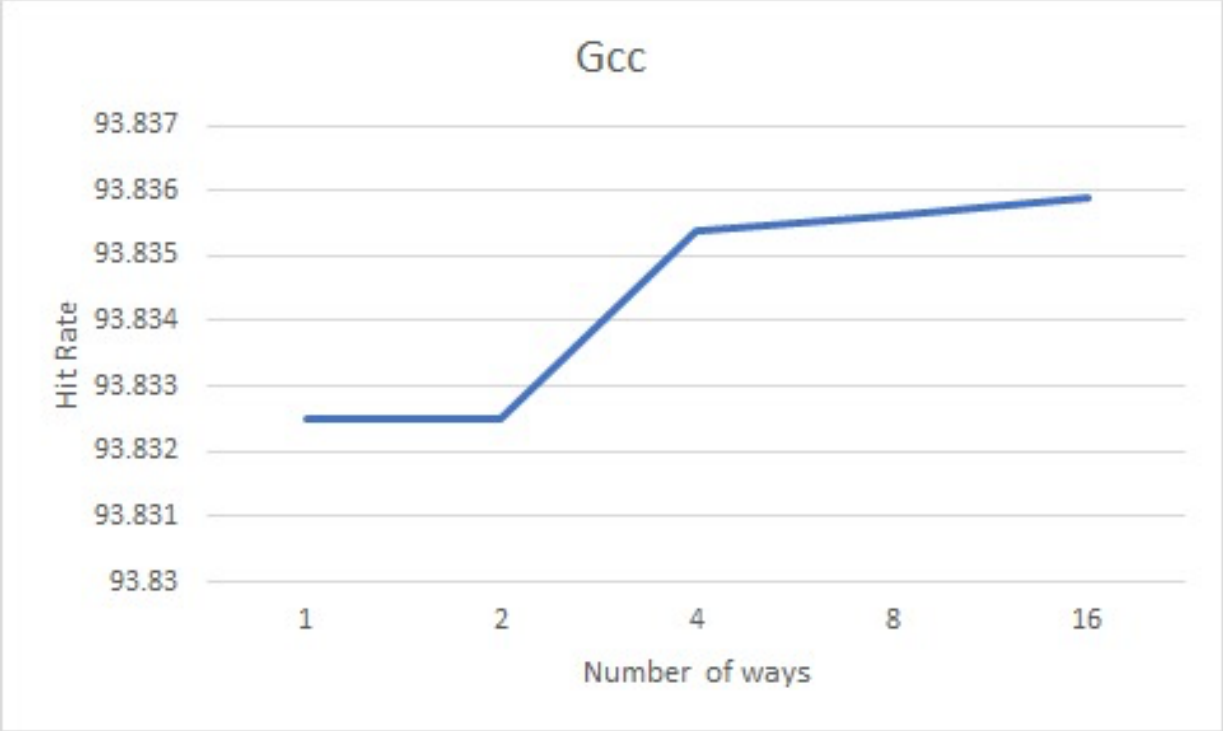
8waySA

FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483894	41	15020	88412	515683	93.8356
swim	280825	0	10489	61544	303193	92.6225
twolf	476844	0	5473	37804	482824	98.7615
mcf	7508	654116	8262	490	727230	1.0324
gzip	320883	157792	1017	5767	481044	66.7055

16waySA

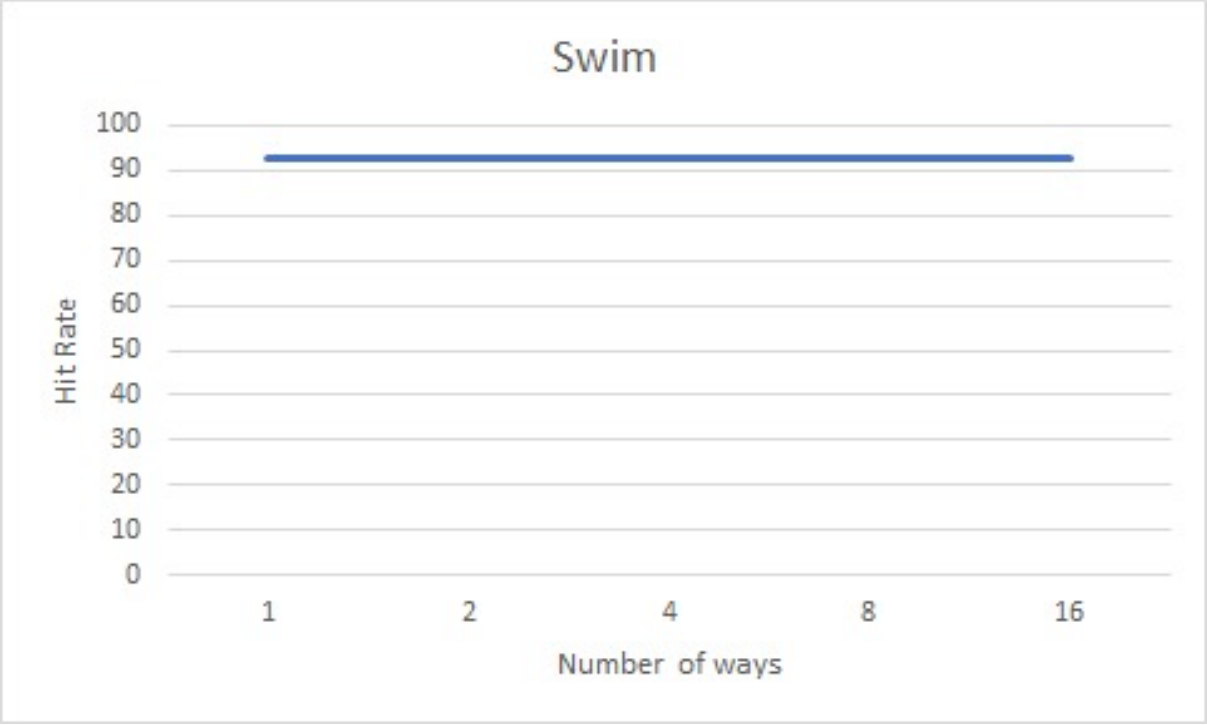
FILE NAME	HITS	EVICTIONS	ROWS	EMPTY	INSTRUCTIONS COUNT	HITRATE
gcc	483896	0	23110	337973	515683	93.8359
swim	280825	0	17649	260016	303193	92.6225
twolf	476844	0	5747	85972	482824	98.7615
mcf	7508	457508	16454	1050	727230	1.0324
gzip	320883	153184	1211	12399	481044	66.7055

Number of Ways	gcc.trace
1	93.8325
2	93.8325
4	93.8354
8	93.8356
16	93.8359



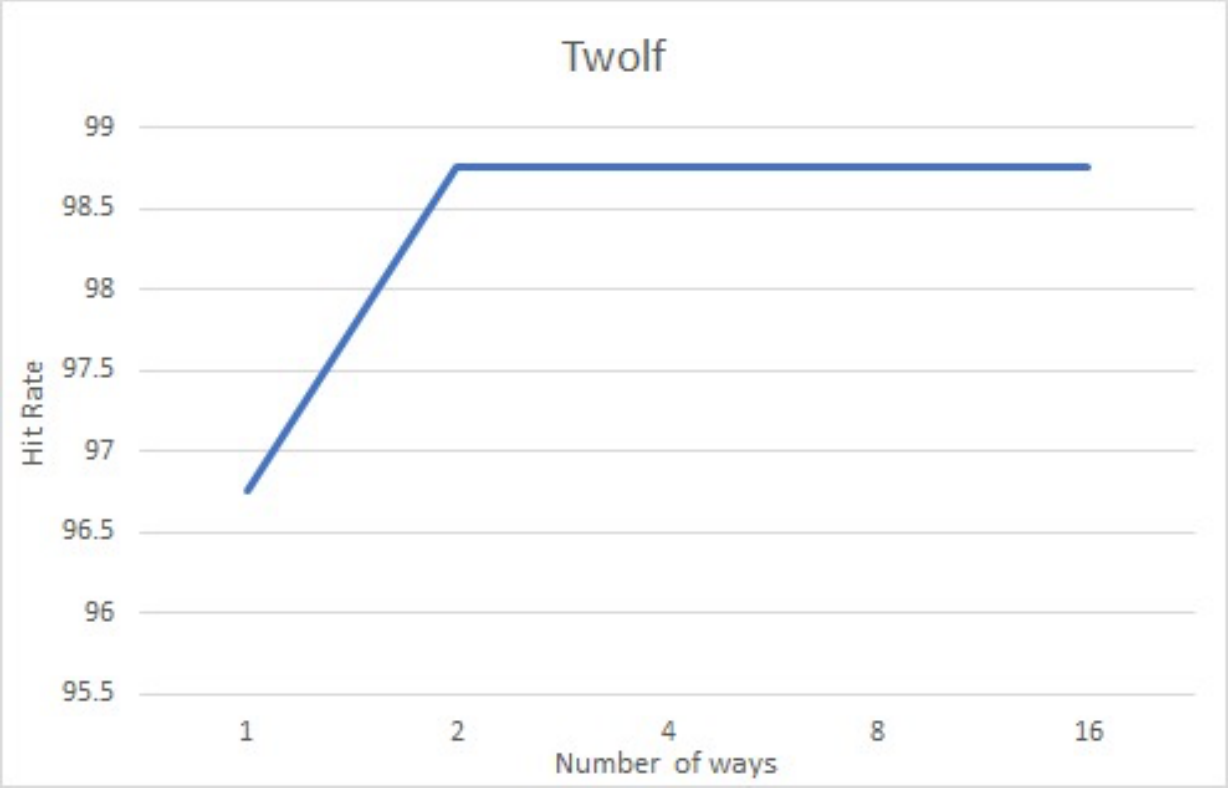
This is the variation of the hit rate in the trace Gcc when we change the number of ways.The given hit rate is in percentage.

Number of Ways	swim.trace
1	92.6225
2	92.6225
4	92.6225
8	92.6225
16	92.6225



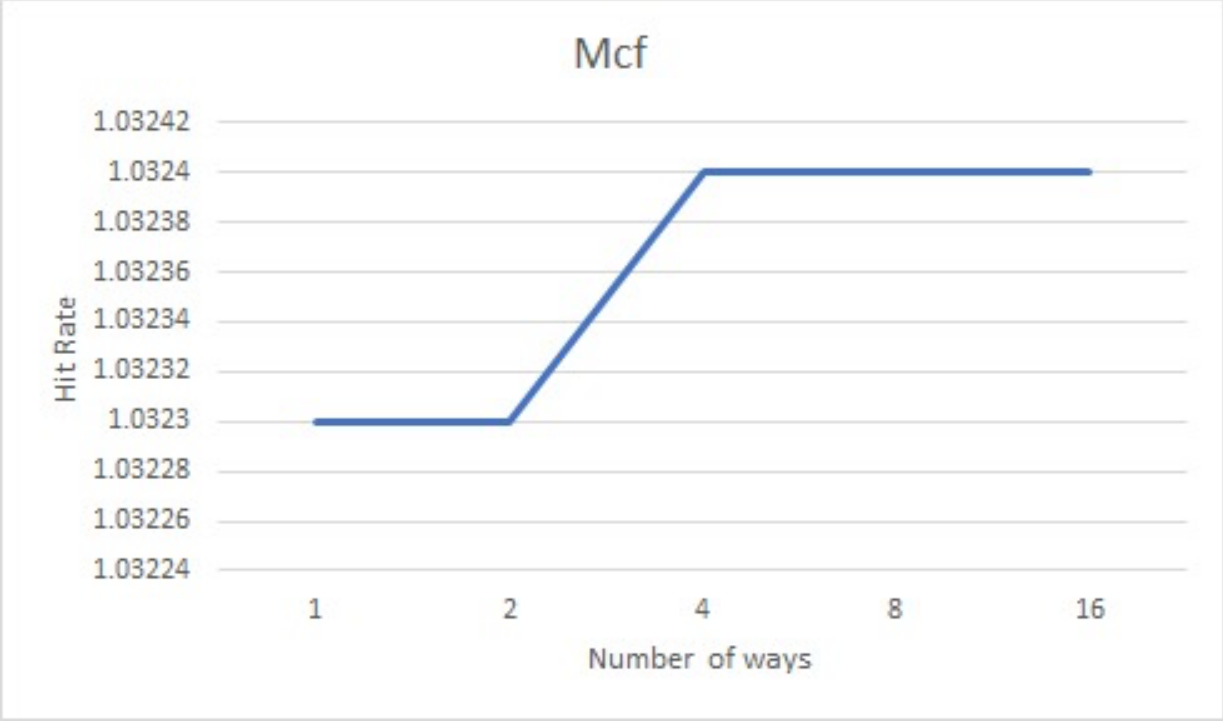
This is the variation of the hit rate in the trace Swim when we change the number of ways.The given hit rate is in percentage.

Number of Ways	twolf.trace
1	96.7608
2	98.7608
4	98.7615
8	98.7615
16	98.7615



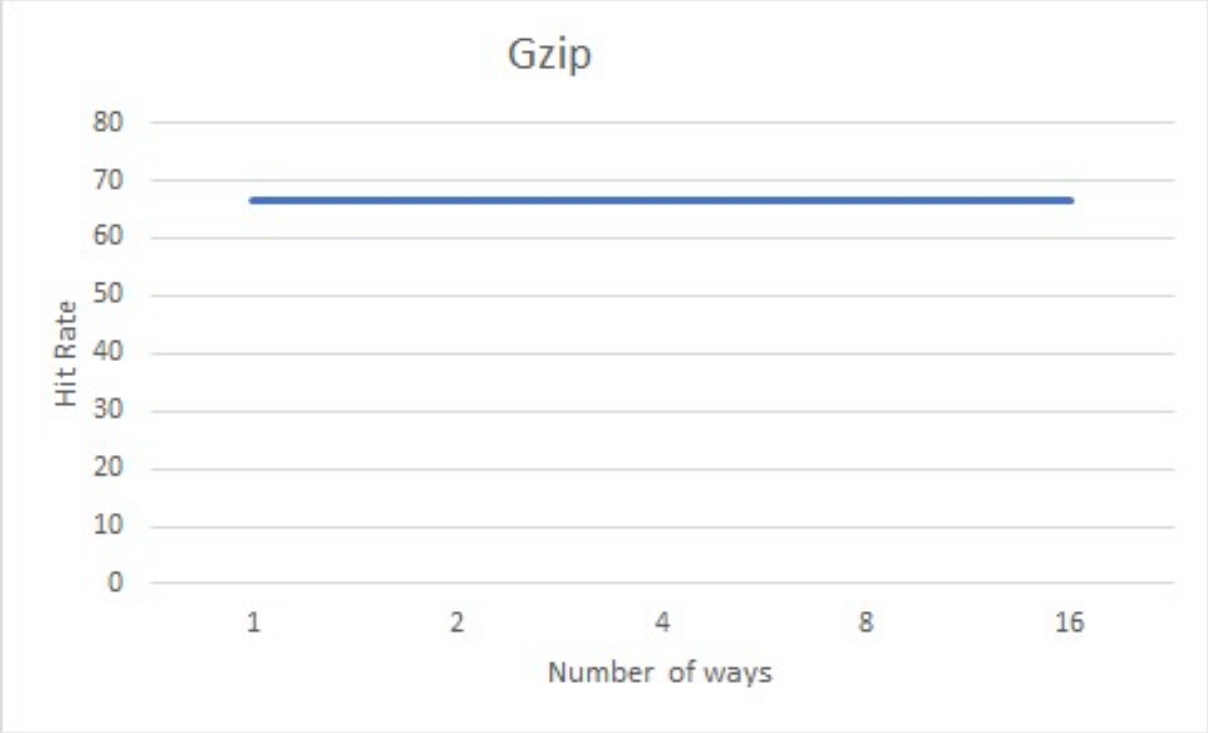
This is the variation of the hit rate in the trace Twolf when we change the number of ways.The given hit rate is in percentage.

Number of Ways	mcf.trace
1	1.0323
2	1.0323
4	1.0324
8	1.0324
16	1.0324



This is the variation of the hit rate in the trace Mcf when we change the number of ways.The given hit rate is in percentage.

Number of Ways	Gzip trace
1	66.7055
2	66.7055
4	66.7055
8	66.7055
16	66.7055



This is the variation of the hit rate in the trace Gzip when we change the number of ways.The given hit rate is in percentage.

This is the implementation of 1-way cache of 512kB and 4-byte block size.

INDEX	WAY1(Valid)	WAY1(Tag)
11111111111011100	1	['0001111111111', 127]
11111111111011110	1	['0001111111111', 28630]
11111111111100110	1	['0001111111111', 200153]
11111111111101000	1	['0001111111111', 4]
11111111111101010	1	['0001111111111', 145]
11111111111101100	1	['0001111111111', 1486]
11111111111101110	1	['1111111111111', 207922]
11111111111110000	1	['0001111111111', 8]
11111111111110010	1	['0001111111111', 9]
11111111111110100	1	['0001111111111', 10]
11111111111110110	1	['0001111111111', 11]
11111111111111000	1	['0001111111111', 12]
11111111111111010	1	['0001111111111', 13]
11111111111111100	1	['0001111111111', 14]
11111111111111110	1	['1111111111111', 469239]
00000000000000000	1	['0000000000000', 478740]
00000000000000010	1	['0000000000000', 478741]
00000000000000100	1	['0000000000000', 478753]
00000000000000110	1	['0000000000000', 225075]
00000000000001000	1	['0000000000000', 478744]
00000000000001010	1	['0000000000000', 478748]
00000000000001100	1	['0000000000000', 478750]
00000000000001110	1	['0000000000000', 478752]
00000000000010000	1	['0000000000000', 199979]
00000000000010010	1	['0000000000000', 199980]
00000000000010100	1	['0000000000000', 478747]
00000000000010110	1	['0000000000000', 478746]
00000000000011000	1	['0000000000000', 114753]
00000000000011010	1	['0000000000000', 114754]
00000000000011100	1	['0010000000000', 30]
00000000000011110	1	['0000000000000', 2768]
00000000000100000	1	['0000000000000', 2755]
00000000000100010	1	['0000000000000', 392277]

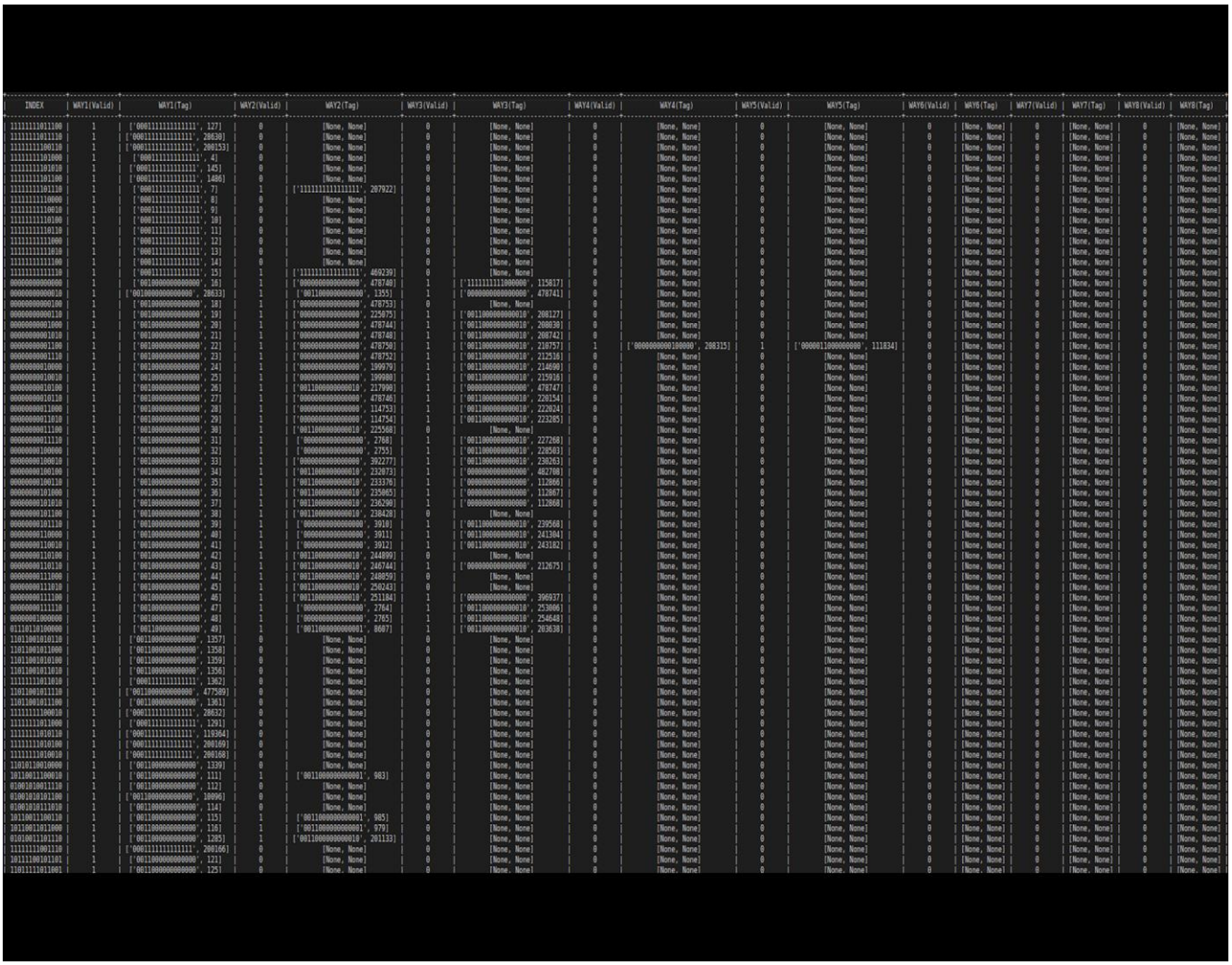
This is the implementation of 2-way cache of 512kB and 4-byte block size.

INDEX	WAY1(Valid)	WAY1(Tag)	WAY2(Valid)	WAY2(Tag)
1111111111011100	1	['00011111111111', 127]	0	[None, None]
1111111111011110	1	['00011111111111', 28630]	0	[None, None]
1111111111100110	1	['00011111111111', 200153]	0	[None, None]
1111111111101000	1	['00011111111111', 4]	0	[None, None]
1111111111101010	1	['00011111111111', 145]	0	[None, None]
1111111111101100	1	['00011111111111', 1486]	0	[None, None]
1111111111101110	1	['00011111111111', 7]	1	['11111111111111', 207922]
1111111111110000	1	['00011111111111', 8]	0	[None, None]
1111111111110010	1	['00011111111111', 9]	0	[None, None]
1111111111110100	1	['00011111111111', 10]	0	[None, None]
1111111111110110	1	['00011111111111', 11]	0	[None, None]
1111111111111000	1	['00011111111111', 12]	0	[None, None]
1111111111111010	1	['00011111111111', 13]	0	[None, None]
1111111111111100	1	['00011111111111', 14]	0	[None, None]
1111111111111110	1	['00011111111111', 15]	1	['11111111111111', 469239]
0000000000000000	1	['11111111110000', 115817]	1	['00000000000000', 478740]
0000000000000010	1	['00000000000000', 478741]	1	['00100000000000', 28633]
0000000000000100	1	['00100000000000', 18]	1	['00000000000000', 478753]
0000000000000110	1	['00100000000000', 19]	1	['00000000000000', 225075]
0000000000001000	1	['00100000000000', 20]	1	['00000000000000', 478744]
0000000000001010	1	['00100000000000', 21]	1	['00000000000000', 478748]
0000000000001100	1	['00000000000000', 478750]	1	['00000000001000', 208315]
0000000000001110	1	['00100000000000', 23]	1	['00000000000000', 478752]
0000000000010000	1	['00100000000000', 24]	1	['00000000000000', 199979]
0000000000010010	1	['00100000000000', 25]	1	['00000000000000', 199980]
0000000000010100	1	['00100000000000', 26]	1	['00000000000000', 478747]
0000000000010110	1	['00100000000000', 27]	1	['00000000000000', 478746]
0000000000011000	1	['00100000000000', 28]	1	['00000000000000', 114753]
0000000000011010	1	['00100000000000', 29]	1	['00000000000000', 114754]
0000000000011100	1	['00100000000000', 30]	0	[None, None]
0000000000011110	1	['00100000000000', 31]	1	['00000000000000', 2768]
0000000000100000	1	['00100000000000', 32]	1	['00000000000000', 2755]
0000000000100010	1	['00100000000000', 33]	1	['00000000000000', 392277]
0000000000100100	1	['00100000000000', 34]	1	['00000000000000', 482708]
0000000000100110	1	['00100000000000', 35]	1	['00000000000000', 112866]

This is the implementation of 4-way cache of 512kB and 4-byte block size.

INDEX	WAY1(Valid)	WAY1(Tag)	WAY2(Valid)	WAY2(Tag)	WAY3(Valid)	WAY3(Tag)	WAY4(Valid)	WAY4(Tag)
11111111101110000	1	['000111111111111', 127]	0	[None, None]	0	[None, None]	0	[None, None]
11111111101111000	1	['000111111111111', 28630]	0	[None, None]	0	[None, None]	0	[None, None]
11111111110011000	1	['000111111111111', 200153]	0	[None, None]	0	[None, None]	0	[None, None]
11111111110100000	1	['000111111111111', 4]	0	[None, None]	0	[None, None]	0	[None, None]
11111111110101000	1	['000111111111111', 145]	0	[None, None]	0	[None, None]	0	[None, None]
11111111110110000	1	['000111111111111', 1486]	0	[None, None]	0	[None, None]	0	[None, None]
11111111110111000	1	['000111111111111', 7]	1	['111111111111111', 207922]	0	[None, None]	0	[None, None]
11111111111000000	1	['000111111111111', 8]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111001000	1	['000111111111111', 9]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111010000	1	['000111111111111', 10]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111011000	1	['000111111111111', 11]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111100000	1	['000111111111111', 12]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111101000	1	['000111111111111', 13]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111110000	1	['000111111111111', 14]	0	[None, None]	0	[None, None]	0	[None, None]
11111111111111000	1	['000111111111111', 15]	1	['111111111111111', 469239]	0	[None, None]	0	[None, None]
00000000000000000	1	['001000000000000', 16]	1	['000000000000000', 478740]	1	['111111111100000', 115817]	0	[None, None]
00000000000001000	1	['001000000000000', 28633]	1	['001100000000000', 1355]	1	['000000000000000', 478741]	0	[None, None]
00000000000010000	1	['001000000000000', 18]	1	['000000000000000', 478753]	0	[None, None]	0	[None, None]
00000000000011000	1	['001000000000000', 19]	1	['000000000000000', 225075]	1	['001100000000001', 208127]	0	[None, None]
00000000000100000	1	['001000000000000', 20]	1	['000000000000000', 478744]	1	['001100000000001', 208030]	0	[None, None]
00000000000101000	1	['001000000000000', 21]	1	['000000000000000', 478748]	1	['001100000000001', 208577]	0	[None, None]
00000000000110000	1	['001000000000000', 22]	1	['000000000000000', 478750]	1	['001100000000001', 210439]	1	['000000000010000', 208315]
00000000000111000	1	['001000000000000', 23]	1	['000000000000000', 478752]	1	['001100000000001', 211669]	0	[None, None]
00000000001000000	1	['001000000000000', 24]	1	['000000000000000', 199979]	1	['001100000000001', 213866]	0	[None, None]
00000000001001000	1	['001000000000000', 25]	1	['000000000000000', 199980]	1	['001100000000001', 215763]	0	[None, None]
00000000001010000	1	['001000000000000', 26]	1	['001100000000001', 217309]	1	['000000000000000', 478747]	0	[None, None]
00000000001011000	1	['001000000000000', 27]	1	['000000000000000', 478746]	1	['001100000000001', 219804]	0	[None, None]
00000000001100000	1	['001000000000000', 28]	1	['000000000000000', 114753]	1	['001100000000001', 221233]	0	[None, None]
00000000001101000	1	['001000000000000', 29]	1	['000000000000000', 114754]	1	['001100000000001', 222982]	0	[None, None]
00000000001110000	1	['001000000000000', 30]	1	['001100000000001', 224241]	0	[None, None]	0	[None, None]
00000000001111000	1	['001000000000000', 31]	1	['000000000000000', 2768]	1	['001100000000001', 226575]	0	[None, None]
00000000010000000	1	['001000000000000', 32]	1	['000000000000000', 2755]	1	['001100000000001', 227711]	0	[None, None]
00000000010001000	1	['001000000000000', 33]	1	['000000000000000', 2756]	1	['001100000000001', 229404]	0	[None, None]
00000000010010000	1	['001000000000000', 34]	1	['001100000000001', 231044]	1	['000000000000000', 482708]	0	[None, None]
00000000010011000	1	['001000000000000', 35]	1	['001100000000001', 233073]	1	['000000000000000', 112866]	0	[None, None]
00000000010100000	1	['001000000000000', 36]	1	['001100000000001', 234888]	1	['000000000000000', 112867]	0	[None, None]
00000000010101000	1	['001000000000000', 37]	1	['001100000000001', 236003]	1	['000000000000000', 112868]	0	[None, None]
00000000010110000	1	['001000000000000', 38]	1	['001100000000001', 237995]	0	[None, None]	0	[None, None]
00000000010111000	1	['001000000000000', 39]	1	['000000000000000', 3910]	1	['001100000000001', 239252]	0	[None, None]
00000000011000000	1	['001000000000000', 40]	1	['000000000000000', 3911]	1	['001100000000001', 240633]	0	[None, None]
00000000011001000	1	['001000000000000', 41]	1	['000000000000000', 3912]	1	['001100000000001', 242524]	0	[None, None]
00000000011010000	1	['001000000000000', 42]	1	['001100000000001', 244724]	0	[None, None]	0	[None, None]
00000000011011000	1	['001000000000000', 43]	1	['001100000000001', 246031]	1	['000000000000000', 212675]	0	[None, None]
00000000011100000	1	['001000000000000', 44]	1	['001100000000001', 247709]	0	[None, None]	0	[None, None]
00000000011101000	1	['001000000000000', 45]	1	['001100000000001', 249757]	0	[None, None]	0	[None, None]
00000000011110000	1	['001000000000000', 46]	1	['001100000000001', 251031]	1	['000000000000000', 396937]	0	[None, None]
00000000011111000	1	['001000000000000', 47]	1	['000000000000000', 2764]	1	['001100000000001', 252184]	0	[None, None]
00000000100000000	1	['001000000000000', 48]	1	['000000000000000', 2765]	1	['001100000000001', 253996]	0	[None, None]

This is the implementation of 8-way cache of 512kB and 4-byte block size.



This is the implementation of 16-way cache of 512kB and 4-byte block size.