5.1

5.1.1 4

5.1.2 I, J

5.1.3 A[I][J]

5.1.4 3596 = 8 × 800/4 × 2−8×8/4 + 8000/4

5.1.5 I, J

5.1.6 A(J, I)

5.2

5.2.1

Word Address	Binary Address	Tag	Index	Hit/Miss
3	0000 0011	0	3	M
180	1011 0100	11	4	M
43	0010 1011	2	11	M
2	0000 0010	0	2	M
191	1011 1111	11	15	M
88	0101 1000	5	8	M
190	1011 1110	11	14	M
14	0000 1110	0	14	M
181	1011 0101	11	5	M
44	0010 1100	2	12	M
186	1011 1010	11	10	M
253	1111 1101	15	13	M

5.2.2

Word Address	Binary Address	Tag	Index	Hit/Miss
3	0000 0011	0	1	M
180	1011 0100	11	2	M
43	0010 1011	2	5	M
2	0000 0010	0	1	H
191	1011 1111	11	7	M
88	0101 1000	5	4	M
190	1011 1110	11	7	H
14	0000 1110	0	7	M
181	1011 0101	11	2	H
44	0010 1100	2	6	M
186	1011 1010	11	5	M
253	1111 1101	15	6	M

5.2.3

Word Address	Binary Address	Tag	Cache 1		Cache 2		Cache 3	
			index	hit/miss	index	hit/miss	index	hit/miss
3	0000 0011	0	3	M	1	M	0	M
180	1011 0100	22	4	M	2	M	1	M
43	0010 1011	5	3	M	1	M	0	M
2	0000 0010	0	2	M	1	M	0	M
191	1011 1111	23	7	M	3	M	1	M
88	0101 1000	11	0	M	0	M	0	M
190	1011 1110	23	6	M	3	H	1	H
14	0000 1110	1	6	M	3	M	1	M
181	1011 0101	22	5	M	2	H	1	M
44	0010 1100	5	4	M	2	M	1	M
186	1011 1010	23	2	M	1	M	0	M
253	1111 1101	31	5	M	2	M	1	M

Cache 1 miss rate = 100%

Cache 1 total cycles = $12 \times 25 + 12 \times 2 = 324$

Cache 2 miss rate = 10/12 = 83%

Cache 2 total cycles = $10 \times 25 + 12 \times 3 = 286$

Cache 3 miss rate = 11/12 = 92%

Cache 3 total cycles = $11 \times 25 + 12 \times 5 = 335$

Cache 2 provides the best performance.

5.2.4 First we must compute the number of cache blocks in the initial cache configuration. For this, we divide 32 KiB by 4 (for the number of bytes per word) and again by 2 (for the number of words per block). This gives us 4096 blocks and a resulting index field width of 12 bits. We also have a word offset size of 1 bit and a byte offset size of 2 bits. This gives us a tag field size of 32 − 15 = 17 bits. These tag bits, along with one valid bit per block, will require $18 \times 4096 = 73728$ bits or 9216 bytes. The total cache size is thus 9216 + 32768 = 41984 bytes.

The total cache size can be generalized to

totalsize = datasize + (validbitsize + tagsize) × blocks

totalsize = 41984

datasize = blocks × blocksize × wordsize

wordsize = 4

tagsize = 32 − log2(blocks) − log2(blocksize) − log2(wordsize)

validbitsize = 1

Increasing from 2-word blocks to 16-word blocks will reduce the tag size from 17 bits to 14 bits.

In order to determine the number of blocks, we solve the inequality:

41984 <= 64 × blocks + 15 × blocks

Solving this inequality gives us 531 blocks, and rounding to the next power of two gives us a 1024-block cache.

The larger block size may require an increased hit time and an increased miss penalty than the original cache. The fewer number of blocks may cause a higher conflict miss rate than the original cache.

5.2.5 Associative caches are designed to reduce the rate of conflict misses. As such, a sequence of read requests with the same 12-bit index field but a different tag field will generate many misses. For the cache described above, the sequence 0, 32768, 0, 32768, 0, 32768, …, would miss on every access, while a 2-way set associate cache with LRU replacement, even one with a significantly smaller overall capacity, would hit on every access after the first two.

5.2.6 Yes, it is possible to use this function to index the cache. However, information about the five bits is lost because the bits are XOR'd, so you must include more tag bits to identify the address in the cache.

5.3

5.3.1 8

5.3.2 32

5.3.3 1+ (22/8/32) = 1.086

5.3.4 3

5.3.5 0.25

5.3.6 <Index, tag, data>

<000001_2, 0001_2, mem[1024]>

<000001_2, 0011_2, mem[16]>

<001011_2, 0000_2, mem[176]>

<001000_2, 0010_2, mem[2176]>

<001110_2, 0000_2, mem[224]>

<001010_2, 0000_2, mem[160]>

5.4

5.4.1 The L1 cache has a low write miss penalty while the L2 cache has a high write miss penalty. A write buffer between the L1 and L2 cache would hide the write miss latency of the L2 cache. The L2 cache would benefit from write buffers when replacing a dirty block, since the new block would be read in before the dirty block is physically written to memory.

5.4.2 On an L1 write miss, the word is written directly to L2 without bringing its block into the L1 cache. If this results in an L2 miss, its block must be brought into the L2 cache, possibly replacing a dirty block which must first be written to memory.

5.4.3 After an L1 write miss, the block will reside in L2 but not in L1. A subsequent read miss on the same block will require that the block in L2 be written back to memory, transferred to L1, and invalidated in L2.

5.4.4 One in four instructions is a data read, one in ten instructions is a data write. For a CPI of 2, there are 0.5 instruction accesses per cycle, 12.5% of cycles will require a data read, and 5% of cycles will require a data write.

The instruction bandwidth is thus $(0.0030 \times 64) \times 0.5 = 0.096$ bytes/cycle. The data read bandwidth is thus $0.02 \times (0.13+0.050) \times 64 = 0.23$ bytes/cycle. The total read bandwidth requirement is 0.33 bytes/cycle. The data write bandwidth requirement is $0.05 \times 4 = 0.2$ bytes/cycle.

5.4.5 The instruction and data read bandwidth requirement is the same as in 5.4.4. The data write bandwidth requirement becomes $0.02 \times 0.30 \times (0.13+0.050) \times 64 = 0.069$ bytes/cycle.

5.4.6 For CPI=1.5 the instruction throughput becomes $1/1.5 = 0.67$ instructions per cycle. The data read frequency becomes $0.25 / 1.5 = 0.17$ and the write frequency becomes $0.10 / 1.5 = 0.067$.

The instruction bandwidth is $(0.0030 \times 64) \times 0.67 = 0.13$ bytes/cycle.

For the write-through cache, the data read bandwidth is $0.02 \times (0.17 +0.067) \times 64 = 0.22$ bytes/cycle. The total read bandwidth is 0.35 bytes/cycle. The data write bandwidth is $0.067 \times 4 = 0.27$ bytes/cycle.

For the write-back cache, the data write bandwidth becomes $0.02 \times 0.30 \times (0.17+0.067) \times 64 = 0.091$ bytes/cycle.

Address	0	4	16	132	232	160	1024	30	140	3100	180	2180
Line ID	0	0	1	8	14	10	0	1	9	1	11	8
Hit/miss	M	H	M	M	M	M	M	H	H	M	M	M
Replace	N	N	N	N	N	N	Y	N	N	Y	N	Y

5.5

5.5.1 Assuming the addresses given as byte addresses, each group of 16 accesses will map to the same 32-byte block so the cache will have a miss rate of 1/16. All misses are compulsory misses. The miss rate is not sensitive to the size of the cache or the size of the working set. It is, however, sensitive to the access pattern and block size.

5.5.2 The miss rates are 1/8, 1/32, and 1/64, respectively. The workload is exploiting temporal locality.

5.5.3 In this case the miss rate is 0.

5.5.4 AMAT for B = 8: $0.040 \times (20 \times 8) = 6.40$

AMAT for B = 16: $0.030 \times (20 \times 16) = 9.60$

AMAT for B = 32: $0.020 \times (20 \times 32) = 12.80$

AMAT for B = 64: $0.015 \times (20 \times 64) = 19.20$

AMAT for B = 128: $0.010 \times (20 \times 128) = 25.60$

B = 8 is optimal.

5.5.5 AMAT for B = 8: $0.040 \times (24 + 8) = 1.28$

AMAT for B = 16: $0.030 \times (24 + 16) = 1.20$

AMAT for B = 32: $0.020 \times (24 + 32) = 1.12$

AMAT for B = 64: $0.015 \times (24 + 64) = 1.32$

AMAT for B = 128: $0.010 \times (24 + 128) = 1.52$

B = 32 is optimal.

5.5.6 B=128

5.6

5.6.1

P1	1.52 GHz
P2	1.11 GHz

5.6.2

P1	6.31 ns	9.56 cycles
P2	5.11 ns	5.68 cycles

5.6.3

P1	12.64 CPI	8.34 ns per inst
P2	7.36 CPI	6.63 ns per inst

5.6.4

6.50 ns	9.85 cycles	Worse

5.6.5 13.04

5.6.6 P1 AMAT = 0.66 ns + 0.08 × 70 ns = 6.26 ns

P2 AMAT = 0.90 ns + 0.06 × (5.62 ns + 0.95 × 70 ns) = 5.23 ns

For P1 to match P2's performance:

5.23 = 0.66 ns + MR × 70 ns

MR = 6.5%

5.7

5.7.1 The cache would have 24 / 3 = 8 blocks per way and thus an index field of 3 bits.

Word Address	Binary Address	Tag	Index	Hit/Miss	Way 0	Way 1	Way 2
3	0000 0011	0	1	M	T(1)=0		
180	1011 0100	11	2	M	T(1)=0 T(2)=11		
43	0010 1011	2	5	M	T(1)=0 T(2)=11 T(5)=2		
2	0000 0010	0	1	M	T(1)=0 T(2)=11 T(5)=2	T(1)=0	
191	1011 1111	11	7	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11	T(1)=0	
88	0101 1000	5	4	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0	
190	1011 1110	11	7	H	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0	
14	0000 1110	0	7	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0 T(7)=0	
181	1011 0101	11	2	H	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5	T(1)=0 T(7)=0	

44	0010 1100	2	6	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5 T(6)=2	T(1)=0 T(7)=0	
186	1011 1010	11	5	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5 T(6)=2	T(1)=0 T(7)=0 T(5)=11	
253	1111 1101	15	6	M	T(1)=0 T(2)=11 T(5)=2 T(7)=11 T(4)=5 T(6)=2	T(1)=0 T(7)=0 T(5)=11 T(6)=15	

5.7.2 Since this cache is fully associative and has one-word blocks, the word address is equivalent to the tag. The only possible way for there to be a hit is a repeated reference to the same word, which doesn't occur for this sequence.

Tag	Hit/Miss	Contents
3	M	3
180	M	3, 180
43	M	3, 180, 43
2	M	3, 180, 43, 2
191	M	3, 180, 43, 2, 191
88	M	3, 180, 43, 2, 191, 88
190	M	3, 180, 43, 2, 191, 88, 190
14	M	3, 180, 43, 2, 191, 88, 190, 14
181	M	181, 180, 43, 2, 191, 88, 190, 14
44	M	181, 44, 43, 2, 191, 88, 190, 14
186	M	181, 44, 186, 2, 191, 88, 190, 14
253	M	181, 44, 186, 253, 191, 88, 190, 14

5.7.3

Address	Tag	Hit/ Miss	Contents
3	1	M	1
180	90	M	1, 90
43	21	M	1, 90, 21
2	1	H	1, 90, 21
191	95	M	1, 90, 21, 95
88	44	M	1, 90, 21, 95, 44
190	95	H	1, 90, 21, 95, 44
14	7	M	1, 90, 21, 95, 44, 7
181	90	H	1, 90, 21, 95, 44, 7
44	22	M	1, 90, 21, 95, 44, 7, 22
186	143	M	1, 90, 21, 95, 44, 7, 22, 143
253	126	M	1, 90, 126, 95, 44, 7, 22, 143

The final reference replaces tag 21 in the cache, since tags 1 and 90 had been reused at time=3 and time=8 while 21 hadn't been used since time=2.

Miss rate = 9/12 = 75%

This is the best possible miss rate, since there were no misses on any block that had been previously evicted from the cache. In fact, the only eviction was for tag 21, which is only referenced once.

5.7.4 L1 only:

$.07 \times 100 = 7$ ns

CPI = 7 ns / .5 ns = 14

Direct mapped L2:

$.07 \times (12 + 0.035 \times 100) = 1.1$ ns

CPI = ceiling(1.1 ns/.5 ns) = 3

8-way set associated L2:

$.07 \times (28 + 0.015 \times 100) = 2.1$ ns

CPI = ceiling(2.1 ns / .5 ns) = 5

Doubled memory access time, L1 only:

$.07 \times 200 = 14$ ns

CPI = 14 ns / .5 ns = 28

Doubled memory access time, direct mapped L2:

$.07 \times (12 + 0.035 \times 200) = 1.3$ ns

CPI = ceiling(1.3 ns/.5 ns) = 3

Doubled memory access time, 8-way set associated L2:

$.07 \times (28 + 0.015 \times 200) = 2.2$ ns

CPI = ceiling(2.2 ns / .5 ns) = 5

Halved memory access time, L1 only:

$.07 \times 50 = 3.5$ ns

CPI = 3.5 ns / .5 ns = 7

Halved memory access time, direct mapped L2:

$.07 \times (12 + 0.035 \times 50) = 1.0$ ns

CPI = ceiling(1.1 ns/.5 ns) = 2

Halved memory access time, 8-way set associated L2:

$.07 \times (28 + 0.015 \times 50) = 2.1\,\text{ns}$

CPI = ceiling(2.1 ns / .5 ns) = 5

5.7.5 $.07 \times (12 + 0.035 \times (50 + 0.013 \times 100)) = 1.0\,\text{ns}$

Adding the L3 cache does reduce the overall memory access time, which is the main advantage of having a L3 cache. The disadvantage is that the L3 cache takes real estate away from having other types of resources, such as functional units.

5.7.6 Even if the miss rate of the L2 cache was 0, a 50 ns access time gives

AMAT = $.07 \times 50 = 3.5$ ns, which is greater than the 1.1 ns and 2.1 ns given by the on-chip L2 caches. As such, no size will achieve the performance goal.