Karan Vora (kv2154)
Computer Systems Architecture Assignment 7

Problem 1):

Solution 1.1):

Tag: 28 bits
Index: 4 bits

Address: 0x03
Tag: 0 (0000)
Index: 3 (0011)
Hit/Miss: M

Address: 0xb4
Tag: b (1011)
Index: 4 (0100)
Hit/Miss: M

Address: 0x2b
Tag: 2 (0010)
Index: b (1011)
Hit/Miss: M

Address: 0x02
Tag: 0 (0000)
Index: 2 (0010)
Hit/Miss: M

Address: 0xbf
Tag: b (1011)
Index: f (1111)
Hit/Miss: M

Address: 0x58
Tag: 5 (0101)
Index: 8 (1000)
Hit/Miss: M

Address: 0xbe
Tag: b (1011)
Index: e (1110)
Hit/Miss: M

Address: 0x0e
Tag: 0 (0000)
Index: e (1110)
Hit/Miss: M

Address: 0xb5
Tag: b (1011)
Index: 5 (0101)
Hit/Miss: M

Address: 0x2c
Tag: 2 (0010)
Index: c (1100)
Hit/Miss: M

Address: 0xba
Tag: b (1011)
Index: a (1010)
Hit/Miss: M

Address: 0xfd
Tag: f (1111)
Index: d (1101)
Hit/Miss: M

Solution 1.2:

Block Size: 2 word
Block Offset: 1 bit
Cache Size: 8 block
Index bit: 3 bit

Address: 0x03
Tag: 0
Index: 1
Offset: 1
Hit/Miss: M

Address: 0xb4
Tag: 11
Index: 2
Offset: 0
Hit/Miss: M

Address: 0x2b
Tag: 2
Index: 5
Offset: 1
Hit/Miss: M

Address: 0x02
Tag: 0
Index: 1
Offset: 0

Hit/Miss: H

Address: 0xbf
Tag: 11
Index: 7
Offset: 1
Hit/Miss: M

Address: 0x58
Tag: 5
Index: 4
Offset: 0
Hit/Miss: M

Address: 0xbe
Tag: 11
Index: 7
Offset: 0
Hit/Miss: H

Address: 0x0e
Tag: 0
Index: 7
Offset: 0
Hit/Miss: M

Address: 0xb5
Tag: 11
Index: 2
Offset: 1
Hit/Miss: H

Address: 0x2c
Tag: 2
Index: 6
Offset: 0
Hit/Miss: M

Address: 0xba
Tag: 11
Index: 5
Offset: 0
Hit/Miss: M

Address: 0xfd
Tag: 15
Index: 6
Offset: 1
Hit/Miss: M

Solution 1.3:

C1:

Tag: 5 bits
Index: 3 bits

Address: 0x03
Tag in Hex: 0x00
Index: 3
Hit/Miss: M

Address: 0xb4
Tag in Hex: 0x16
Index: 4
Hit/Miss: M

Address: 0x2b
Tag in Hex: 0x05
Index: 3
Hit/Miss: M

Address: 0x02
Tag in Hex: 0x00
Index: 2
Hit/Miss: M

Address: 0xbf
Tag in Hex: 0x17
Index: 7
Hit/Miss: M

Address: 0x58
Tag in Hex: 0x0b
Index: 0
Hit/Miss: M

Address: 0xbe
Tag in Hex: 0x17
Index: 6
Hit/Miss: M

Address: 0x0e
Tag in Hex: 0x01
Index: 6
Hit/Miss: M

Address: 0xb5
Tag in Hex: 0x16

Index: 5
Hit/Miss: M

Address: 0x2c
Tag in Hex: 0x05
Index: 4
Hit/Miss: M

Address: 0xba
Tag in Hex: 0x17
Index: 2
Hit/Miss: M

Address: 0xfd
Tag in Hex: 0x1f
Index: 5
Hit/Miss: M

C2:

Tag: 5 bits
Index: 2 bits

Address: 0x03
Tag in Hex: 0x00
Index: 1
Hit/Miss: M

Address: 0xb4
Tag in Hex: 0x16
Index: 2
Hit/Miss: M

Address: 0x2b
Tag in Hex: 0x05
Index: 1
Hit/Miss: M

Address: 0x02
Tag in Hex: 0x00
Index: 1
Hit/Miss: M

Address: 0xbf
Tag in Hex: 0x17
Index: 3
Hit/Miss: M

Address: 0x58

Tag in Hex: 0x0b
Index: 0
Hit/Miss: M

Address: 0xbe
Tag in Hex: 0x17
Index: 3
Hit/Miss: H

Address: 0x0e
Tag in Hex: 0x01
Index: 3
Hit/Miss: M

Address: 0xb5
Tag in Hex: 0x16
Index: 2
Hit/Miss: H

Address: 0x2c
Tag in Hex: 0x05
Index: 2
Hit/Miss: M

Address: 0xba
Tag in Hex: 0x17
Index: 1
Hit/Miss: M

Address: 0xfd
Tag in Hex: 0x1f
Index: 2
Hit/Miss: M

C2:

Tag: 5 bits
Index: 1 bits

Address: 0x03
Tag in Hex: 0x00
Index: 0
Hit/Miss: M

Address: 0xb4
Tag in Hex: 0x16
Index: 1
Hit/Miss: M

Address: 0x2b
Tag in Hex: 0x05
Index: 0
Hit/Miss: M

Address: 0x02
Tag in Hex: 0x00
Index: 0
Hit/Miss: M

Address: 0xbf
Tag in Hex: 0x17
Index: 1
Hit/Miss: M

Address: 0x58
Tag in Hex: 0x0b
Index: 0
Hit/Miss: M

Address: 0xbe
Tag in Hex: 0x17
Index: 1
Hit/Miss: H

Address: 0x0e
Tag in Hex: 0x01
Index: 1
Hit/Miss: M

Address: 0xb5
Tag in Hex: 0x16
Index: 1
Hit/Miss: M

Address: 0x2c
Tag in Hex: 0x05
Index: 1
Hit/Miss: M

Address: 0xba
Tag in Hex: 0x17
Index: 0
Hit/Miss: M

Address: 0xfd
Tag in Hex: 0x1f
Index: 1
Hit/Miss: M

C1 Miss rate: 100%
C2 Miss rate: 83%
C3 Miss rate: 92%

==========

Problem 2):

For a larger direct-mapped cache to have a lower or equal miss rate than a smaller 2-way set associative cache, it would need to have at least double the cache block size. The advantage of such a solution is less misses for nearby addresses, but with the disadvantage of suffering longer access time. For the second part, it is possible to use the function to index the cache as per asked. However, Information about this six bits is lost because the bits are XORed. Therefore you should include more lag bits to identify the address in the cache.

==========

Problem 3):

Solution 3.1):

Offset = byte select + word select
8 Bytes / Word (64 bit word) = 3 byte select bits
Words select bits = 5 offset bits - 3 byte select bits = 2 bits
Words /block = 2^2 = 4 words/block

Solution 3.2):

Since the cache size is determined by the index bits, then with 5 index bits we have 2^5 or 32 blocks.

Solution 3.3):

Cache size = 32 blocks * 4 words/block * 64 bits/word = 8192 bits
Determine tag information storage = 54 * 2^5= 1728 [bits].
Total tag bits = 32 blocks * 54 tags bits/block = 1728 bits
Valid bits = 32 blocks * 1 bit/block = 32 bits
Determine total bits required for cache = 8192 + 1728 + 32 = 9952 bits.
Total bits:data bits = 9952 total / 8192 data = 1.215

Solution 3.4):

00 -> 0000 0000 0000 0000 -> (1) Tag=0000 00, Index=00 000, Offset=0 0000, (2) Miss, (3) None

04 -> 0000 0000 0000 0100 -> (1) Tag=0000 00, Index=00 000, Offset=0 0100, (2) Hit, (3) None

10 -> 0000 0000 0001 0000 -> (1) Tag=0000 00, Index=00 000, Offset=1 0000, (2) Hit, (3) None

84 -> 0000 0000 1000 0100 -> (1) Tag=0000 00, Index=00 100, Offset=0 0100, (2) Miss, (3) None

E8 -> 0000 0000 1110 1000 -> (1) Tag=0000 00, Index=00 111, Offset=0 1000, (2) Miss, (3) None

A0 -> 0000 0000 1010 0000 -> (1) Tag=0000 00, Index=00 101, Offset=0 0000, (2) Miss, (3) None

400 -> 0000 0100 0000 0000 -> (1) Tag=0000 01, Index=00 000, Offset=0 0000, (2) Miss, (3) Replace 0th Index

1E -> 0000 0000 0001 1110 -> (1) Tag=0000 00, Index=00 000, Offset=1 1110, (2) Miss, (3) Replace 0th Index

8C -> 0000 0000 1000 1100 -> (1) Tag=0000 00, Index=00 100, Offset=0 1100, (2) Hit, (3) None

C1C -> 0000 1100 0001 1100 -> (1) Tag=0000 11, Index=00 000, Offset=1 1100, (2) Miss, (3) Replace 0th Index

B4 -> 0000 0000 1011 0100 -> (1) Tag=0000 00, Index=00 101, Offset=1 0100, (2) Hit, (3) None

884 -> 0000 1000 1000 0100 -> (1) Tag=0000 10, Index=00 100, Offset=0 0100, (2) Miss, (3) Replace 4th Index

Solution 3.5):

Given total access as 12, total hit as 4, and total miss as 8.
Determine hit ratio=4/12=0.333.

Solution 3.6):

<0, 3, Mem[0xC00]-Mem[0xC1F]>
<4, 2, Mem[0x880]-Mem[0x89F]>
<5, 0, Mem[0xA0]-Mem[0xBF]>
<7, 0, Mem[0xE0]-Mem[0xFF]>

==========

Solution 4):

Solution 4.1):

Buffer needed between L1 and L2 caches. The possible buffer needed between L1 cache and L2 cache is write buffer or it can be called write back buffer. It is used to reduce the access latency. If any miss occurs on L1 cache, then fetch the data from L2 cache. Now, by using the write back L2 cache load the data the data into L1 cache. Hence, in between L1 cache and L2 cache, the needed buffer list is write buffer or back buffer.

Buffer needed between L2 and memory. The possible buffer needed between L2 cache and memory is called write buffer and store buffer which is used to reduce the access latency. If any miss occurs on L2 cache then the write back buffer takes or loads the data from the main memory to L2 cache memory. The loaded data is stored in the Store buffer to store or hold the data before the processor place the new data into store buffer. Hence, in between L2 cache and memory, the needed buffer list is write buffer or

write back buffer and store buffer.

Solution 4.2):

According to the given condition such as non-write allocate and write through cache for L1 cache, there is no need to check dirty block on L1 cache. Thus, we check L2 cache directly. If there is any miss occurrence on L2 cache memory, check if the block is dirty. If block is dirty then a block must be allocated to L2 cache memory and then the evicted block is written back to main memory, else, If there is any hit on L2 cache memory then L2 and the dirty block set is simply updated.

Solution 4.3):

When we are going to write data in L1 cache, the following of two condition occurs,
→ If the written data is present, then there is no need to update the L1 cache block.
→ If the data is not available, then the write-miss has occurred on L1. It means data is fetched from L2 memory.

Now, when the data is fetched from L2 cache memory then also two conditions occurs:

Fetch data is available or not.
→ If data is present on L2 cache then it loads into cache L1 and the dirty block on cache L1 is modified.
→ If the data is not present then the data is fetched from the main memory. It means write-miss occurs on L2 cache memory and it is loaded from the main memory and then the dirty block is replaced.

Hence, in the same way, the read-miss occurs when the readable data is not present on L1 and L2 cache memory and it is loaded from the main memory and then the dirty block is replaced. So, when any miss occurs, fetch the data for next level of memory and replace the dirty block after every miss-read/write occurrence.

==========

Problem 5):

Solution 5.1):

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 (miss/instruction) * 64 (bytes/miss)) * 0.5 (instruction/cycle) = 0.096 bytes/cycle.

The data read bandwidth is thus 0.02 (miss/access) (0.13 (reads/cycle)+0.050 (writes/cycle)) * 64 (bytes/miss) = 0.23 bytes/cycle.

The total read bandwidth requirement is 0.33 bytes/cycle. (instruction bw + data bw)

The data write bandwidth requirement is 0.05 (writes/cycle) * 4 (bytes/write) = 0.2

bytes/cycle.

Solution 5.2):

The instruction and data read bandwidth requirement is the same as in 5.7.1. The data write bandwidth requirement becomes 0.02 (miss/access) * 0.30 (writes/miss) * (0.13+0.050) (access/cycle) * 64 (bytes/write)= 0.069 bytes/cycle.

==========

Problem 6):

Solution 6.1):

Given: video streaming workload access time = 512 KiB, sequence of address stream = 0, 2, 4, 6, 8, 10, 12, 14, 16, ..., difference between the address streams = 2, direct mapped cache size = 64 KiB, and size of cache line = 32 bytes

Determine block size:
block size = (video streaming workload access time)/(size of cache line) = (512 KiB)/(32 bytes) = 16 K

Determine occurrence of miss:
occurrence of miss = ((direct mapped cache size)/(block size))/(difference between the address streams) = ((64 KiB)/(16 K)/(2) = (2 B)*((8 bits)/(1 B)) = 16 bits

Determine miss rate:
miss rate = (1/occurence of miss)*100% = 1/(16 bits)*100% = 6.25%

The miss rate is independent of the working set as well as the size of the cache. So, the miss rate does not change with the working set or cache size.

In a 3C Model, the cache model consists of cache misses classified into one of the following 3 categories:
1) compulsory misses/cold-start misses
2) capacity misses
3) conflict misses/collision misses
As seen, since the cache misses are caused based on the first access to a block and the remaining accesses are not present in the cache, the cache misses fall under the category cold-start misses.

Solution 6.2):

→ Block Size: 16 bytes

Given: video streaming workload access time = 512 KiB, sequence of address stream = 0, 2, 4, 6, 8, 10, 12, 14, 16, ..., difference between the address streams = 2, direct mapped cache size = 64 KiB, and size of cache line = 16 bytes

Determine block size:

block size = (video streaming workload access time)/(size of cache line) = (512 KiB)/(16 bytes) = 32 K
Determine occurrence of miss:
occurrence of miss = ((direct mapped cache size)/(block size))/(difference between the address streams) = ((64 KiB)/(32 K)/(2) = (1 B)*((8 bits)/(1 B)) = 8 bits

Determine miss rate:
miss rate = (1/occurence of miss)*100% = 1/(8 bits)*100% = 12.5%
From the calculations, the miss rate for 16 bytes cache line is 12.5%.

→ Block size: 64 bytes

Given: video streaming workload access time = 512 KiB, sequence of address stream = 0, 2, 4, 6, 8, 10, 12, 14, 16, ..., difference between the address streams = 2, direct mapped cache size = 64 KiB, and size of cache line = 64 bytes

Determine block size:
block size = (video streaming workload access time)/(size of cache line) = (512 KiB)/(64 bytes) = 8 K

Determine occurrence of miss:
occurrence of miss = ((direct mapped cache size)/(block size))/(difference between the address streams) = ((64 KiB)/(8 K)/(2) = (4 B)*((8 bits)/(1 B)) = 32 bits

Determine miss rate:
miss rate = (1/occurence of miss)*100% = 1/(32 bits)*100% = 3.125%
From the calculations, the miss rate for 64 bytes cache line is 3.125%

→ Block size: 128 bytes

Given: video streaming workload access time = 512 KiB, sequence of address stream = 0, 2, 4, 6, 8, 10, 12, 14, 16, ..., difference between the address streams = 2, direct mapped cache size = 64 KiB, and size of cache line = 128 bytes

Determine block size:
block size = (video streaming workload access time)/(size of cache line) = (512 KiB)/(128 bytes) = 4 K

Determine occurrence of miss:
occurrence of miss = ((direct mapped cache size)/(block size))/(difference between the address streams) = ((64 KiB)/(4 K)/(2) = (8 B)*((8 bits)/(1 B)) = 64 bits

Determine miss rate:
miss rate = (1/occurence of miss)*100% = 1/(64 bits)*100% = 1.5625%
From the calculations, the miss rate for 128 bytes cache line is 1.5625%.

The kind of locality this workload is exploiting is spatial locality because the next access is a nearby one for every access.

Solution 6.3):

Given: video streaming workload access time = 512 [KiB], sequence of address stream = 0, 2, 4, 6, 8, 10, 12, 14, 16, ..., difference between the address streams = 2

Assumptions:

→ A two-entry stream buffer.
→ Cache latency is such that one cache block can be preloaded before the completion of the previous cache block.
→ A cache is divided into cache blocks also called cache lines.
→ Prefetching technique is effectively used in streaming applications.
→ In the prefetching approach, future accesses are predicted at an early stage into the cache memory and the cache blocks are obtained on speculation.
→ During the execution of the current block a new speculated block is preloaded, through which the miss rate reduces to zero effectively.
→ When byte 0 is accessed, it results in a miss. Fetch the entire cache block and cache it. Using the two-entry stream buffer prefetching, the sequentially adjacent cache block is prefetched into separate buffers 1 and 2.
→ When the byte 2 is accessed, and if it is already present in prefetch buffer it is considered as a hit.
→ Similarly, prefetch the next 2 bytes 3 and 4. The process for the rest of 512 KiB.
→ Observing the sequence of addresses, there occurs only 1 miss and rests of accesses are hits.

Determine miss rate:
miss rate = ((occurence of misses)/(total accesses))*100% = (1/((512*1024)/2))*100% = 0.000382%

==========

Problem 7):

Solution 7.1):

B = 8, Therefore 0.04*20*8 = 6.4
B = 16, Therefore 0.03*20*16 = 9.6
B = 32, Therefore 0.02*20*32 = 12.8
B = 64, Therefore 0.015*20*64 = 19.2
B = 128, Therefore 0.01*20*128 = 25.6

Thus, B = 8 is optimal.

Solution 7.2):

B = 8, Therefore 0.04*(24+8) = 1.28
B = 16, Therefore 0.03*(24+16) = 1.2
B = 32, Therefore 0.02*(24+32) = 1.12
B = 64, Therefore 0.015*(24+64) = 1.32
B = 128, Therefore 0.01*(24+128) = 1.52

Thus, B = 32 is optimal.

Solution 7.3):

B = 128

==========

Problem 8):

Solution 8.1):

P1 Cycle Time = 0.66 ns. $f_1 = 1/T = 1/(0.66 \times 10^{-9}) = 1.52$ GHz
P2 Cycle Time = 0.90 ns. $f_2 = 1/T = 1/(0.90 \times 10^{-9}) = 1.11$ GHz

Solution 8.2):

AMAT = Hit Time + (Miss Rate) x (Miss Latency)

For P1: AMAT = 0.66 + (0.08)(70) = 6.26 ns
For P2: AMAT = 0.90 + (0.06)(70) = 5.10 ns

Solution 8.3):

$$CPI = Base\,CPI + Miss\,Rate * \frac{Memory\,Access}{Instruction} * Miss\,Penalty$$

For P1: CPI = 1 + (0.08)(1.36)(70/0.66) = 12.54
For P2: CPI = 1 + (0.06)(1.36)(70/0.90) = 7.34

Solution 8.4):

Find AMAT for L2 cache: AMAT = Hit Time + (Miss Rate) x (Miss Latency)
AMAT = 5.62 + (0.95)(70) = 72.12 ns

Find AMAT for processor core: AMAT = Hit Time + (Miss Rate) x (Miss Latency)
AMAT = 0.66 + (0.08)(72.12) = 6.43 ns

Solution 8.5):

$$CPI = Base\,CPI + L1\,Miss\,Rate * \frac{Memory\,Access}{Instruction} * L1\,Miss\,Penalty$$
$$+ L2\,Miss\,Rate * \frac{Memory\,Access}{Instruction} * L2\,Miss\,Penalty$$

CPI = 1 + (0.08)(1.36)(5.62/0.66) + (0.95*0.08)(1.36)(70/0.66) = 12.89

Solution 8.6):

P1's CPI is 12.89, from 5.6.5. P2's CPI is 7.34. In of AMAT, P1's AMAT is 6.43 ns with an L2 cache. P2's AMAT is 5.10 ns. P2 is faster.

Since P2 is faster, set P1's AMAT to equal P2's and solve for the miss rate:

AMAT = Hit Time + (Miss Rate) x (Miss Latency) = 5.10
Miss Rate = (AMAT - Hit Time) / Miss Latency = (5.10 - 0.66)/70 = 6.3%.

Solution 8.7):

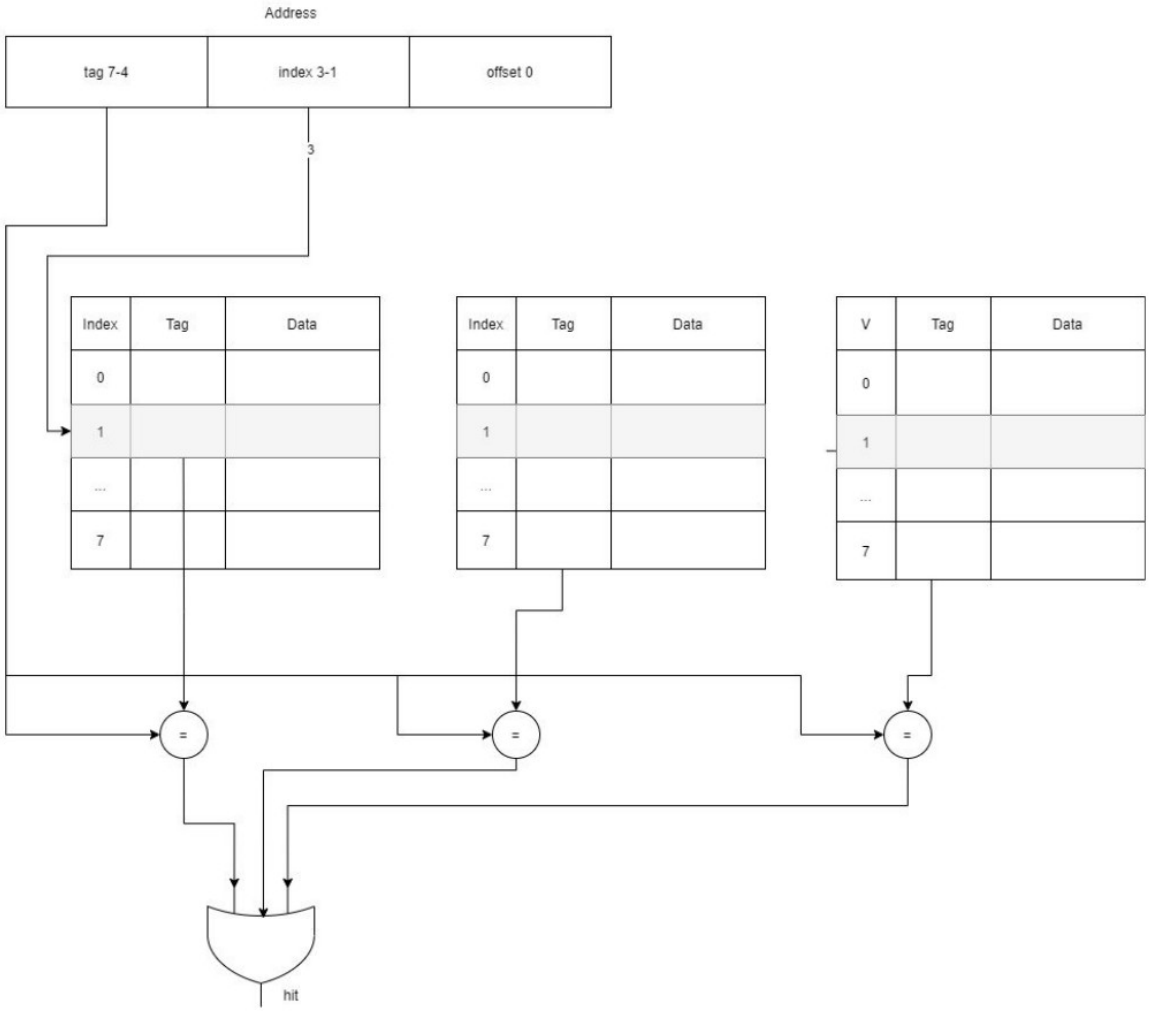
P1's CPI is 12.89, from 5.6.5. P2's CPI is 7.34. In of AMAT, P1's AMAT is 6.43 ns with an L2 cache. P2's AMAT is 5.10 ns. P2 is faster.
Since P2 is faster, set P1's AMAT to equal P2's and solve for the miss rate:

AMAT = Hit Time + (Miss Rate) x (Miss Latency) = 5.10
Miss Rate = (AMAT - Hit Time) / Miss Latency = (5.10 - 0.66)/70 = 6.3%.

==========

Problem 9):

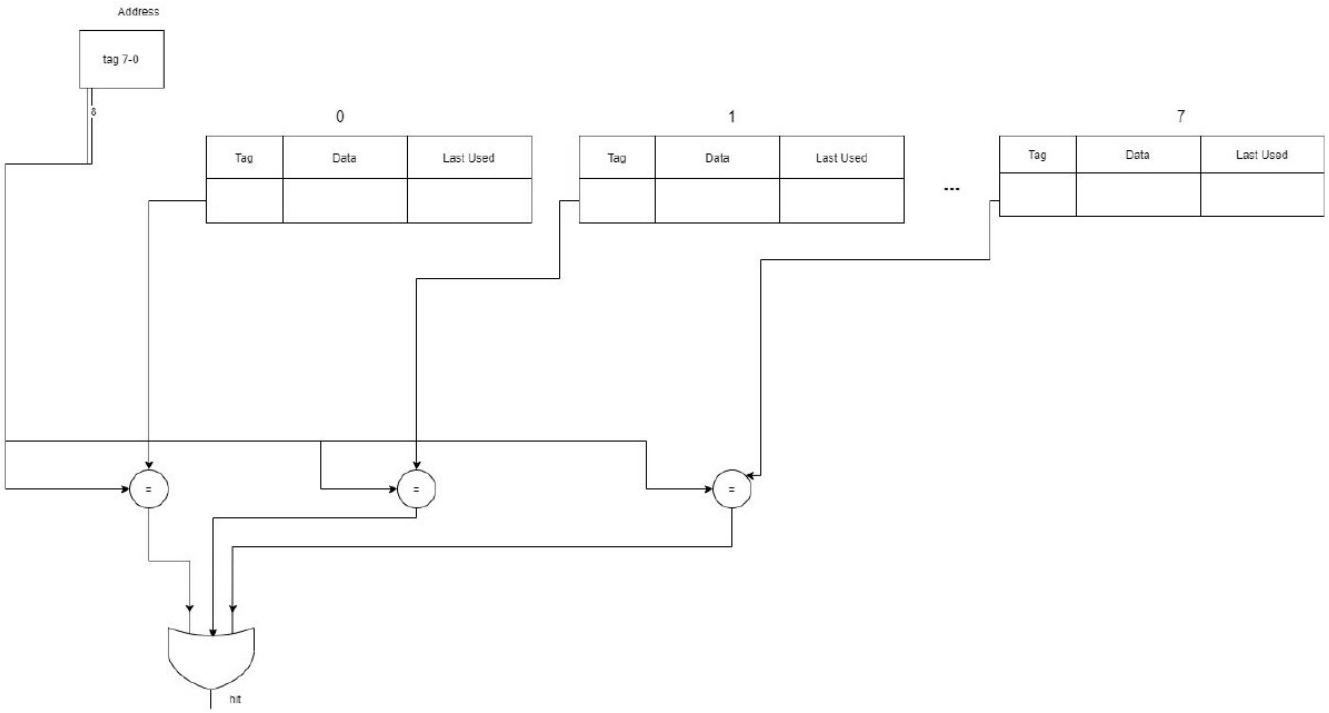
Solution 9.1):



Solution 9.2):

Hex	Binary	Tag	Index	Offset	H/M
0x03	0000 0011	0x0	001	1	M
0xb4	1011 0101	0xb	010	0	M
0x2b	0010 1011	0x2	101	1	M
0x02	0000 0010	0x0	001	0	H
0xbe	1011 1110	0xb	111	0	M
0x58	0101 1000	0x5	100	0	M
0xbf	1011 1111	0xb	111	1	H
0x0e	0000 1110	0x0	111	0	M
0x1f	0001 1111	0x1	111	1	M
0xb5	1011 0101	0xb	010	1	H
0xbf	1011 1111	0xb	111	1	H
0xba	1011 1010	0xb	101	0	M
0x2e	0010 1110	0x2	111	0	M
0xce	1100 1110	0xc	111	0	M

Solution 9.3):



Solution 9.4):

Hex	Binary	Tag	Index	Offset	H/M	Cache
0x03	0000 0011	0x03	N/A	N/A	M	3
0xb4	1011 0101	0xb4	N/A	N/A	M	3, b4
0x2b	0010 1011	0x2b	N/A	N/A	M	3, b4, 2b
0x02	0000 0010	0x02	N/A	N/A	M	3, b4, 2b, 2
0xbe	1011 1110	0xbe	N/A	N/A	M	3, b4, 2b, 2, be
0x58	0101 1000	0x58	N/A	N/A	M	3, b4, 2b, 2, be, 58
0xbf	1011 1111	0xbf	N/A	N/A	M	3, b4, 2b, 2, be, 58, bf
0x0e	0000 1110	0x0e	N/A	N/A	M	3, b4, 2b, 2, be, 58, bf, e
0x1f	0001 1111	0x1f	N/A	N/A	M	b4, 2b, 2, be, 58, bf, e, 1f
0xb5	1011 0101	0xb5	N/A	N/A	M	2b, 2, be, 58, bf, e, 1f, b5
0xbf	1011 1111	0xbf	N/A	N/A	H	2b, 2, be, 58, e, 1f, b5, bf
0xba	1011 1010	0xba	N/A	N/A	M	2, be, 58, e, 1f, b5, bf, ba
0x2e	0010 1110	0x2e	N/A	N/A	M	be, 58, e, 1f, b5, bf, ba, 2e
0xce	1100 1110	0xce	N/A	N/A	M	58, e, 1f, b5, bf, ba, 2e, ce

Solution 9.5):

Similar to 9.3 except the data field is twice as large holding 2 words

Solution 9.6):

Hex	Binary	Tag	Index	Offset	H/M	Cache
0x03	0000 0011	0x01	N/A	1	M	[2, 3]
0xb4	1011 0101	0x5a	N/A	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	N/A	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	N/A	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	1011 1110	0x5f	N/A	0	M	[b4,b5], [2a,2b] [2,3], [be,bf]
0x58	0101 1000	0x2c	N/A	0	M	[2a,2b] [2,3], [be,bf], [58,59]
0xbf	1011 1111	0x5f	N/A	1	H	[2a,2b] [2,3], [58,59], [be,bf]
0x0e	0000 1110	0x07	N/A	0	M	[2,3], [58,59], [be,bf], [e,f]
0x1f	0001 1111	0x0f	N/A	1	M	[58,59], [be,bf], [e,f], [1e,1f]
0xb5	1011 0101	0xb5	N/A	1	M	[be,bf], [e,f], [1e,1f], [b4,b5]
0xbf	1011 1111	0xbf	N/A	1	H	[e,f], [1e,1f], [b4,b5], [be,bf]

0xba	1011 1010	0xba	N/A	0	M	[1e,1f], [b4,b5], [be,bf], [ba,bb]
0x2e	0010 1110	0x2e	N/A	0	M	[b4,b5], [be,bf], [ba,bb], [2e,2f]
0xce	1100 1110	0xce	N/A	0	M	[be,bf], [ba,bb], [2e,2f], [ce,cf]

Solution 9.7):

Hex	Binary	Tag	Index	Offset	H/M	Cache
0x03	0000 0011	0x01	N/A	1	M	[2, 3]
0xb4	1011 0101	0x5a	N/A	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	N/A	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	N/A	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	1011 1110	0x5f	N/A	0	M	[b4,b5], [2a,2b] [2,3], [be,bf]
0x58	0101 1000	0x2c	N/A	0	M	[b4,b5], [2a,2b] [2,3], [58,59]
0xbf	1011 1111	0x5f	N/A	1	M	[b4,b5], [2a,2b] [2,3], [be,bf]
0x0e	0000 1110	0x07	N/A	0	M	[b4,b5], [2a,2b] [2,3], [e,f]
0x1f	0001 1111	0x0f	N/A	1	M	[b4,b5], [2a,2b] [2,3], [1e,1f]
0xb5	1011 0101	0x5a	N/A	1	H	[2a,2b] [2,3], [1e,1f], [b4,b5]
0xbf	1011 1111	0x5f	N/A	1	M	[2a,2b] [2,3], [1e,1f], [be,bf]
0xba	1011 1010	0x5d	N/A	0	M	[2a,2b] [2,3], [1e,1f], [ba,bb]
0x2e	0010 1110	0x17	N/A	0	M	[2a,2b] [2,3], [1e,1f], [2e,2f]
0xce	1100 1110	0x67	N/A	0	M	[2a,2b] [2,3], [1e,1f], [ce,cf]

For the first memory reference in Hex: 0x03, if we invert the last bit in this address string (to get the address of the second word in the Block) we would have the memory reference in Hex: 0x02. Thus these 2 memory references are identified in the Content column – for this pair and for following pairs. In the Content column, the pair of memory references in italics correspond to the candidate words that are replaced in the next cycle with an MRU replacement policy. Content column entries in bold correspond to entries that are responsive to a hit – that were loaded into the cache from previous memory requests and are requested again.

Solution 9.8):

Hex	Binary	Tag	Index	Offset	H/M	Cache
0x03	0000 0011	0x01	N/A	1	M	[2, 3]
0xb4	1011 0101	0x5a	N/A	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	N/A	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	N/A	0	H	[2,3], [b4,b5], [2a,2b]

0xbe	1011 1110	0x5f	N/A	0	M	[2,3], [b4,b5], [2a,2b] [be,bf]
0x58	0101 1000	0x2c	N/A	0	M	[58,59], [b4,b5], [2a,2b], [be,bf]
0xbf	1011 1111	0x5f	N/A	1	M	[58,59], [b4,b5], [2a,2b], [be,bf]
0x0e	0000 1110	0x07	N/A	0	M	[e,f], [b4,b5], [2a,2b], [be,bf]
0x1f	0001 1111	0x0f	N/A	1	M	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xb5	1011 0101	0x5a	N/A	1	H	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xbf	1011 1111	0x5f	N/A	1	M	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xba	1011 1010	0x5d	N/A	0	M	[1e,1f], [ba,bb], [2a,2b], [be,bf]
0x2e	0010 1110	0x17	N/A	0	M	[1e,1f], [ba,bb], [2e,2f], [be,bf]
0xce	1100 1110	0x67	N/A	0	M	[1e,1f], [ba,bb], [2e,2f], [ce,cf]

For the 6th memory reference, 0x58, the LRU policy used
For the 8th memory reference, 0x0e, the MRU policy used
For the 9th memory reference, 0x1f, the MRU policy used
For the 12-14th memory reference, 0xbf-0xce, the LRU policy used

==========

Problem 10):

Solution 10.1):

Standard memory time: Each cycle on a 2 GHz machine takes 0.5 ps. Thus, a main memory access requires 100/0.5 = 200 cycles

L1 only: 1.5 + 0.07*200 = 15.5
Direct mapped L2: 1.5 + .07 × (12 + 0.035 × 200) = 2.83
8-way set associated L2: 1.5 + .07 × (28 + 0.015 × 200) = 3.67.

Doubled memory access time. A main memory access requires 400 cycles

L1 only: 1.5 + 0.07*400 = 29.5 (90% increase)
Direct mapped L2: 1.5 + .07 × (12 + 0.035 × 400) = 3.32 (17% increase)
8-way set associated L2: 1.5 + .07 × (28 + 0.015 × 400) = 3.88 (5% increase).

Solution 10.2):

1.5 + 0.07 × (12 + 0.035 × (50 + 0.013 × 100)) = 2.47

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

Solution 10.3):

We want the CPI of the CPU with an external L2 cache to be at most 2.83.
Let x be the necessary miss rate.

$1.5 + 0.07*(50 + x*200) < 2.83$

Solving for x gives that $x < -0.155$. This means that even if the miss rate of the L2 cache was 0, a 50-ns access time gives a CPI of $1.5 + 0.07*(50 + 0*200) = 5$, which is greater than the 2.83 given by the on-chip L2 caches. As such, no size will achieve the performance goal.