NYU Tandon School of Engineering

Fall 2023, ECE 6913

Homework Assignment 6

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistant Office Hour Schedule

On Zoom: 9:30AM – 11AM Monday, Tuesday, Wednesday & Thursday(Focus on Project A)

1. Arushi Arora, aa10350@nyu.edu

2. Prashanth Rebala, pr2359@nyu.edu

3. Moin Khan, mk8793@nyu.edu

4. Raj Ghodasara, g4357@nyu.edu

Homework Assignment 6 [released Thursday November 9th 2023] [due Wednesday November 22nd by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.
Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

.

---

1. Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 64-bit memory address references, given as word addresses.

`0x02, 0xb3, 0x2a, 0x01, 0xbe, 0x57, 0xbf, 0x0d, 0xb6, 0x2b, 0xbc, 0xfd`

1.1 For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

Hex Memory Reference	Binary Reference	Tag	Index	Hit / Miss
0x03	0000 0011	0	3	M
0xb4	1011 0100	B	4	M
0x2b	0010 1011	2	b	M
0x02	0000 0010	0	2	M
0xbf	1011 1111	B	f	M
0x58	0101 1000	5	8	M
0xbe	1011 1110	B	e	M
0x0e	0000 1110	0	e	M
0xb5	1011 0101	B	5	M
0x2c	0010 1100	2	c	M
0xba	1011 1010	B	a	M
0xfd	1111 1101	f	d	M

1.2 For each of these references, identify the binary word address, the tag, the index, and the offset given a direct-mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Hex Memory Reference	Binary Reference	Tag	Cache Index	Hit / Miss	Block Offset
0x03	0000 0011	0	1	M	1
0xb4	1011 0100	b	2	M	0
0x2b	0010 1011	2	5	M	1
0x02	0000 0010	0	1	M	0
0xbf	1011 1111	b	7	M	1
0x58	0101 1000	5	4	M	0
0xbe	1011 1110	B	7	M	0
0x0e	0000 1110	0	7	M	0
0xb5	1011 0101	B	2	M	1
0x2c	0010 1100	2	6	M	0
0xba	1011 1010	b	5	M	0
0xfd	1111 1101	f	6	M	1

1.3 You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of eight words of data:
C1 has 1-word blocks,
C2 has 2-word blocks, and
C3 has 4-word blocks.

Word Address	Binary Address	Tag (5 bits in hex)	Cache 1 block size = 1 word		Cache 2 block size = 2 word		Cache 3 block size = 4 word	
			Index (3 bits)	Hit/Miss	Index (2 bits)	Hit/Miss	Index (1 bit)	Hit/Miss
0x03	0000 0011	0x00	3	M		M	0	M
0xb4	1011 0100	0x16	4	M	1	M	1	M
0x2b	0010 1011	0x05	3	M	21	M	0	M
0x02	0000 0010	0x00	2	M	1	M	0	M
0xbf	1011 1111	0x17	7	M	3	M	1	M
0x58	0101 1000	0x0b	0	M	0	M	0	M
0xbe	1011 1110	0x17	6	M	3	M	1	M
0x0e	0000 1110	0x01	6	M	3	M	1	M
0xb5	1011 0101	0x16	5	M	2	M	1	M
0x2c	0010 1100	0x05	4	M	2	M	1	M

0xba	1011 1010	0x17	2	M	1	M	0	M
0xfd	1111 1101	0x1f	5	M	2	M	1	M

2. *Section 5.3* shows the typical method to index a direct-mapped cache, specifically (Block address) modulo (Number of blocks in the cache). Assuming a 64-bit address and 1024 blocks in the cache, consider a different indexing function, specifically (Block address[63:54] XOR Block address[53:44]). Is it possible to use this to index a direct-mapped cache? If so, explain why and discuss any changes that might need to be made to the cache. If it is not possible, explain why.

In a direct-mapped cache with a 10-bit index, it is crucial that the 10-bit cache index remains unique for each block address. This ensures that a given block address cannot map to more than one cache index, as illustrated in the color-coded figure (5.8 from the text). Without this uniqueness, a block address could potentially map to multiple cache locations. Therefore, any function capable of generating a distinct 10-bit output corresponding to the cache index, covering all possible cache blocks, is considered satisfactory.

Furthermore, in Observation B, it is noted that for every unique set of bits in [63:54], there exists only one result of the XOR function for each of the 1024 combinations in [53:44]. This guarantees a unique cache index for any given memory address vector of 64 bits. Combining both observations, it can be concluded that the proposed XOR function for indexing the cache in a direct-mapped cache is deemed sufficient.

3. For a direct-mapped cache design with a 64-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
63–10	9–5	4–0

3.1 What is the cache block size (in words)?

3.2 How many blocks does the cache have?

3.3 What is the ratio between total bits required for such a cache implementation over the data storage bits?

Beginning from power on, the following byte-addressed cache references are recorded.

Address												
Hex	00	04	10	84	E8	A0	400	1E	8C	C1C	B4	884
Dec	0	4	16	132	232	160	1024	30	140	3100	180	2180

3.4 For each reference, list (1) its tag, index, and offset, (2) whether it is a hit or a miss, and (3) which bytes were replaced (if any).

3.5 What is the hit ratio?

Each Block has 32 Bytes (offset is 5 bits wide) or 4 64-bit words or 4 8-Byte words (total of 32 Bytes).

2 bits determine one of 4 (64-bit) Words in the Block, 3 least significant bits determine the byte in each 64-bit word.

5 bits in the index field indicate 32 Blocks or 32 lines in the cache.

The cache stores 32 Blocks x 4 Words/Block x 8 Bytes/word = 1024 Bytes = 8192 bits.

In addition to data, 53 bits for Tag and 1 valid bit.

Total bits required = 8192 + 54x32 + 1 x 32 = 9952 bits
9952 / 8192 = 1.2.

Byte Address	Binary Address	Tag	Index	Offset	Line replaced	Hit/Miss
0x00	000000000000	0x0	0x00	0x00	No	M
0x04	000000000100	0x0	0x00	0x04	No	H
0x10	000000010000	0x0	0x00	0x10	No	H
0x84	000010000100	0x0	0x04	0x04	No	M
0xe8	000011101000	0x0	0x07	0x08	No	M

0xa0	000010100000	0x0	0x05	0x00	No	M
0x400	010000000000	0x1	0x00	0x00	Yes	M
0x1r	000000011110	0x0	0x00	0x1e	Yes	M
0x8c	000010001100	0x0	0x04	0x0c	No	H
0xc1c	110000011100	0x3	0x00	0x1c	Yes	M
0xb4	000010110100	0x0	0x05	0x14	No	H
0x884	100010000100	0x2	0x04	0x04	Yes	M

4. Recall that we have two write policies and two write allocation policies, and their combinations can be implemented either in L1 or L2 cache. Assume the following choices for L1 and L2 caches:

L1	L2
Write through, non-write allocate	Write back, write allocate

4.1 Buffers are employed between different levels of memory hierarchy to reduce access latency. For this given configuration, list the possible buffers needed between L1 and L2 caches, as well as L2 cache and memory.

The L1 cache exhibits a minimal write miss penalty, whereas the L2 cache incurs a higher write miss penalty due to the increased latency between RAM and L2 compared to that between L1 and L2. Introducing a write buffer between the L1 and L2 caches would efficiently streamline the write process to the L2 cache, reducing it to just one cycle. This buffer, designed with sufficient depth, can store data destined for L2 from L1, preventing delays caused by subsequent write misses in L1.

The utilization of write buffers between L1 and L2 would prove advantageous for the L2 cache, especially during the replacement of a dirty block. In such cases, the new block can be read into the buffer between L1 and L2 and retained there until the dirty block is written to memory. Only after this process can the new block overwrite the old one in the L2 cache.

4.2 Describe the procedure of handling an L1 write-miss, considering the components involved and the possibility of replacing a dirty block.

In the event of an L1 write miss, the word is directly written to L2 without bringing its block into the L1 cache, adhering to a write-through or write-back policy with no write-allocate. If this leads to an L2 miss, the corresponding block needs to be fetched into the L2 cache from memory. This action may involve replacing a dirty block in the L2 cache, necessitating the prior write of that block to memory.

4.3 For a multilevel exclusive cache configuration (a block can only reside in one of the L1 and L2 caches), describe the procedures of handling an L1 write-miss and an L1 read-miss, considering the components involved and the possibility of replacing a dirty block.

Following an L1 write miss, the block will be present in L2 but absent in L1. If a subsequent read miss occurs on the same block, it becomes necessary to write the block back to memory from L2, transfer it to L1, and invalidate its presence in L2.

5. Consider the following program and cache behaviors.

Data Reads per 1000 Instructions	Data Writes per 1000 Instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (bytes)
250	100	0.30%	2%	64

5.1 Suppose a CPU with a write-through, writeallocate cache achieves a CPI of 2. What are the read and write bandwidths (measured by bytes per cycle) between RAM and the cache? (Assume each miss generates a request for one block.)

Instruction bandwidth: When the CPI is 2, there are, on average, 0.5 instruction accesses per cycle. 0.5 instructions read from Instruction memory per cycle 0.3% of these instruction accesses cause a cache Read miss (and subsequent memory request). [0.5 instr/cycle] x [0.003 misses/instruction] = missed instructions/cycle Assuming each miss requests one block and each block is 64 bytes [8 words with 8 bytes (64 bits) per word], instruction accesses generate an average of [0.5 instr/cycle] x [0.003 misses/instruction] x [64 bytes/block] = = 0.096 bytes/cycle of read traffic

Read Data bandwidth: 25% of instructions generate a read request. [0.5 instr/cycle] x [0.25 Read Data Accesses/instruction] = [0.125 Read Data Accesses / cycle] 2% of these generate a cache miss; [0.125 Read Data Accesses / cycle] x [0.02 misses / Read Data Access] = 0.0025 Read Misses/cycle Assuming each miss requests one block and each block is 64 bytes [8 words with 8 bytes (64 bits) per word] , [0.0025 Read Misses/cycle] x [64 Bytes/block] x [1 block/miss] = 0.0025 x 64 Bytes/cycle = 0.16 Bytes/cycle

Write Data bandwidth: 10% of instructions generate a write request. [0.5 instr/cycle] x [0.10 Write Data Accesses/instruction] = [0.05 Write Data Accesses / cycle] All of the words written to the cache must be written into Memory: [0.05 Write Data Accesses / cycle] x [8 bytes/word] x [1 word/write-through] = 0.4 Bytes/cycle For a Write-allocate policy, a Write miss also makes a read request to RAM

5.2 For a write-back, write-allocate cache, assuming 30% of replaced data cache blocks are dirty, what are the read and write bandwidths needed for a CPI of 2?

[0.5 inst/cycle] x [0.10 Write Data Accesses/instruction + 0.25 Read Data Accesses/instruction] = 0.175 Accesses/cycle [0.175 Accesses/cycle] x [0.02 misses /Access] = 0.0035 misses/cycle [0.0035 misses/cycle] x [0.3 blocks/miss] = 0.00105 blocks/cycle [0.00105 blocks/cycle] x [64 bytes/block] = =0.0672 bytes/cycle

6. Media applications that play audio or video files are part of a class of workloads called "streaming" workloads (i.e., they bring in large amounts of data but do not reuse much of it). Consider a video streaming workload that accesses a 512 KiB working set sequentially with the following word address stream:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 …

6.1 Assume a 64 KiB direct-mapped cache with a 32-byte block. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

As the address stream consists of word addresses, and each 32-byte block comprises four words, every fourth access results in a miss, indicating a miss rate of 1/4. All misses are classified as compulsory misses, stemming from the initial access to a block not previously stored in the cache. The miss rate remains unaffected by changes in cache size or the working set size. However, it is responsive to variations in access patterns and block sizes.

6.2 Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

If the block size is reduced by half, the miss rates will increase twofold, as the 16-byte block now accommodates only 2 words, causing every second access to result in a miss (a miss rate of 1/2). Conversely, for a cache block size of 64 bytes (comprising 8 words), the miss rate decreases to 1/8, as every eighth access leads to a miss. Similarly, with a cache block size of 128 bytes (including 16 words), the miss rate becomes 1/16, as every sixteenth access turns out to be a miss.

6.3 "*Prefetching*" is a technique that leverages predictable address patterns to speculatively bring in additional cache blocks when a particular cache block is accessed. One example of prefetching is a stream buffer that prefetches sequentially adjacent cache blocks into a separate buffer when a particular cache block is brought in. If the data are found in the prefetch buffer, it is considered as a hit, moved into the cache, and the next cache block is prefetched. Assume a two-entry stream buffer; and, assume that the cache latency is such that a cache block can be loaded before the computation on the previous cache block is completed. What is the miss rate for the address stream above?

The miss rate is zero because the pre-fetch buffer consistently has the subsequent request prepared and waiting.

7. Cache block size (B) can affect both miss rate and miss latency. Assuming a machine with a base CPI of 1, and an average of 1.35 references (both instruction and data) per instruction, find the block size that minimizes the total miss latency given the following miss rates for various block sizes.

8: 4%	16: 3%	32: 2%	64: 1.5%	128: 1%

7.1 What is the optimal block size for a miss latency of 20 ×B cycles?

AMAT = Miss Rate x Miss Latency

AMAT for B = 8: 0.040 × (20 × 8) = 6.40

AMAT for B = 16: 0.030 × (20 × 16) = 9.60 AMAT for B = 32: 0.020 × (20 × 32) = 12.80

AMAT for B = 64: 0.015 × (20 × 64) = 19.20 AMAT for B = 128: 0.010 × (20 × 128) = 25.60

B = 8 is optimal.

7.2 What is the optimal block size for a miss latency of 24 +B cycles?

AMAT = Miss Rate x (24 + B)

AMAT for B = 8: 0.040 × (24 + 8) = 1.28

AMAT for B = 16: 0.030 × (24 + 16) = 1.20

AMAT for B = 32: 0.020 × (24 + 32) = 1.12

AMAT for B = 64: 0.015 × (24 + 64) = 1.32

AMAT for B = 128: 0.010 × (24 + 128) = 1.52

B = 32 is optimal

7.3 For constant miss latency, what is the optimal block size?

B = 128 is optimal: Minimizing the miss rate minimizes the total miss latency.

8. In this exercise, we will look at the different ways capacity affects overall performance. In general, cache access time is proportional to capacity. Assume that main memory accesses take 70 ns and that 36% of all instructions access data memory. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

	L1 Size	L1 Miss Rate	L1 Hit Time
P1	2 KiB	8.0%	0.66 ns
P2	4 KiB	6.0%	0.90 ns

8.1 Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates?

Cycle times for P1 and P2 when L1 hit time determines cycle time: P1: 1.515 GHz; P2: 1.11 GHz

8.2 What is the Average Memory Access Time for P1 and P2 (in cycles)?

Average Memory Access Time (in cycles) = # of cycles for a hit + Miss Rate x Miss Penalty

Miss penalty for P1 = 70ns/0.66ns = 107 cycles

Miss penalty for P2 = 70ns/0.90ns = 78 cycles

P1: AMAT = 1 + 0.08 x 107 cycles = 9.56 cycles or 6.31 ns

P2: AMAT = 1 + 0.06 x 78 cycles = 5.68 cycles or 5.11 ns

8.3 Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster? (When we say a "base CPI of 1.0", we mean that instructions complete in one cycle, unless either the instruction access or the data access causes a cache miss.)

we will now consider the addition of an L2 cache to P1 (to presumably make up for its limited L1 cache capacity). Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate.

L2 Size	L2 Miss Rate	L2 Hit Time
1 MiB	95%	5.62 ns

For P1: Each instruction requires 1 cycle for a hit

If the instruction fetch from the Instruction memory misses at Miss Rate of 8%, the instruction incurs a 107 cycle delay + 0.08 x 107 cycles = 8.56

36% of the instructions also require access to Data Memory at a Miss rate of 8% which incur an additional delay of + 0.36 x 0.08 x 107 cycles = 3.08

So, 1 + 0.08 x 107 + 0.36 x 0.08 x 107 = 12.64 cycles @ 0.66ns/cycle = 8.34ns

For P2: Each instruction requires 1 cycle for a hit 1 cycle

If the instruction fetch from the Instruction memory misses at Miss Rate of 8%, the instruction incurs a 107 cycle delay + 0.06 x 78 cycles = 4.68 36% of the instructions also require access to Data Memory at a Miss rate of 8% which incur an additional delay of + 0.36 x 0.06 x 78 cycles = 1.68

So, 1 + 0.08 x 78 + 0.36 x 0.08 x 78 = 7.36 cycles @ 0.66ns/cycle = 6.63ns

8.4 What is the AMAT for P1 with the addition of an L2 cache? Is the AMAT better or worse with the L2 cache?

An L2 access requires 9 cycles (5.62ns/0.66ns).

All memory accesses require at least one cycle.

8% of memory accesses miss in the L1 cache and make an L2 access, which takes 9 cycles.

95% of all L2 access are misses and require a 107-cycle memory lookup.

AMAT = 1 + .08[9 + 0.95*107] = 9.85 cycles – worse than without the L2 (9.56 cycles)

8.5 Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 with the addition of an L2 cache?

CPI for P1 with an L2 cache is [AMAT] + [%DataMemory Accesses/cycle] * [AMAT-1]

9.85 * 0.36*8.85 = 13.04 cycles

8.6 What would the L2 miss rate need to be in order for P1 with an L2 cache to be faster than P1 without an L2 cache?

AMAT with L2 = 1 + .08[9 + MR_L2*107] = AMAT without L2 = 1 + 0.08 x 107 cycles = 9.56

So, When the MR_L2 drops to below.

[(9.56 -1)/0.08 -9]/107 = 91.5%, the AMAT with L2 will become faster than without the L2

8.7 What would the L2 miss rate need to be in order for P1 with an L2 cache to be faster than P2 without an L2 cache?

We want P1's average time per instruction to be less than 6.63 ns. This means that we want

(CPI_P1 * 0.66) < 6.63.

Thus, we need

CPI_P1 < 10.05

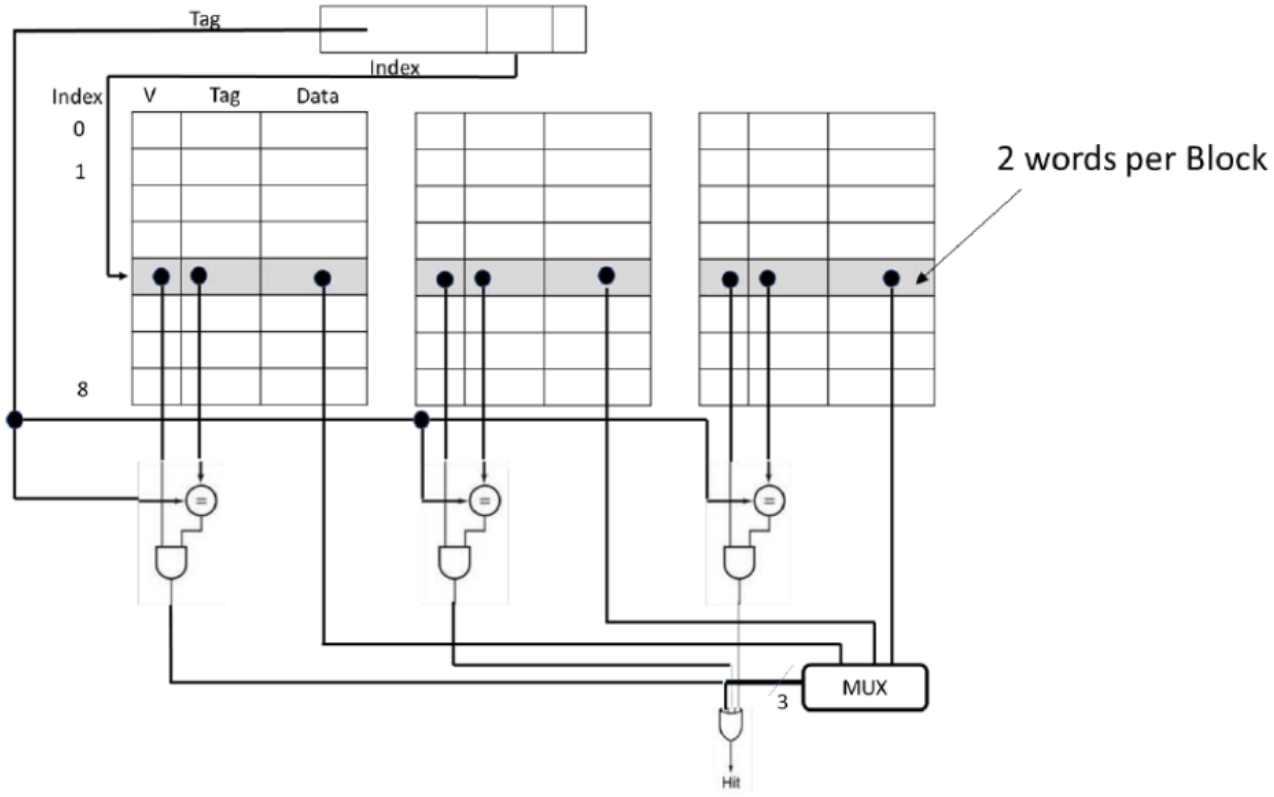
CPI_P1 = AMAT_P1 + 0.36(AMAT_P1 – 1)

AMAT_P1+ 0.36(AMAT_P1-1) < 10.05
AMAT_P1< 7.65.
or 1+ 0.08[9+ MR_L2*107] < 7.65
or MR_L2< 0.693

9. This exercise examines the effect of different cache designs, specifically comparing associative caches to the direct-mapped caches from *Section 5.4*. For these exercises, refer to the sequence of word address shown below.

`0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f, 0xb5, 0xbf, 0xba, 0x2e, 0xce`

9.1 Sketch the organization of a three-way set associative cache with two-word blocks and a total size of 48 words. Your sketch should have a style similar to *Figure 5.18*, but clearly show the width of the tag and data fields.



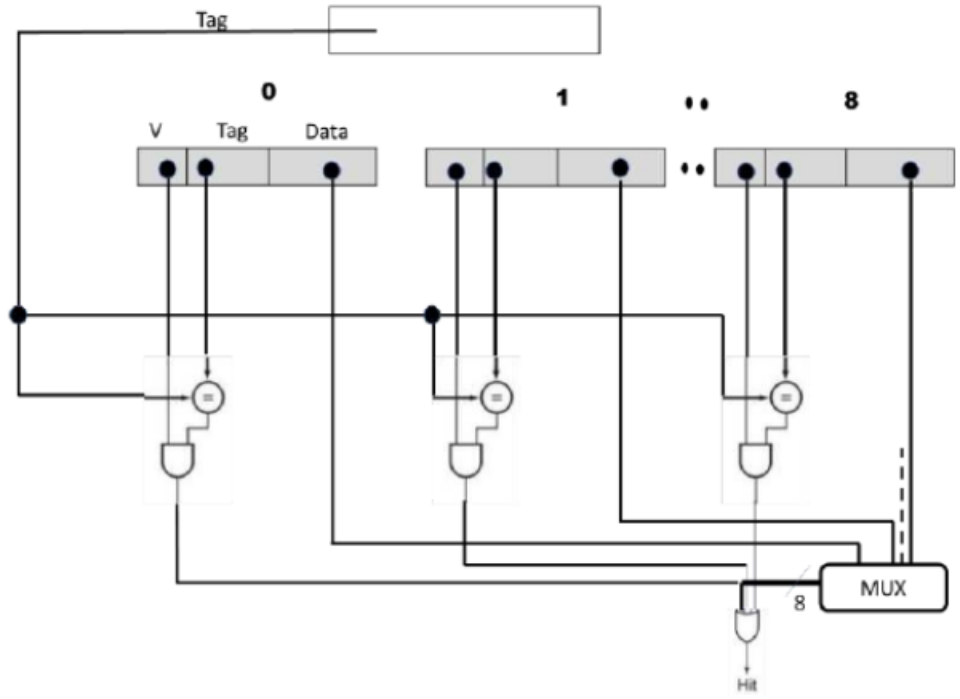
9.2 Trace the behavior of the cache from Exercise 9.1 Assume a true LRU replacement policy. For each reference, identify

- *the binary word address,*
- *the tag,*
- *the index,*
- *the offset*
- *whether the reference is a hit or a miss, and*
- *which tags are in each way of the cache after the reference has been handled.*

Hex	Binary	Tag	Index	Hit / Miss	Offset
0x03	00000011	0x0	1	M	1
0xb4	10110100	0xb	2	M	0
0x2b	00101011	0x2	5	M	1
0x02	0000010	0x0	1	H	0
0xbe	10111110	0xb	7	M	0
0x58	01011000	0x5	4	M	0

0xbf	10111111	0xb	7	H	1
0x0e	00001110	0x0	7	M	0
0x1f	00011111	0x1	7	M	1
0xb5	10110101	0xb	2	H	1
0xbf	10111111	0xb	7	H	1
0xba	10111010	0xb	5	M	0
0x2e	00101110	0x2	7	M	0
0xce	11001110	0xc	7	M	0

9.3 Sketch the organization of a fully associative cache with one-word blocks and a total size of eight words. Your sketch should have a style similar to Figure 5.18, but clearly show the width of the tag and data fields.



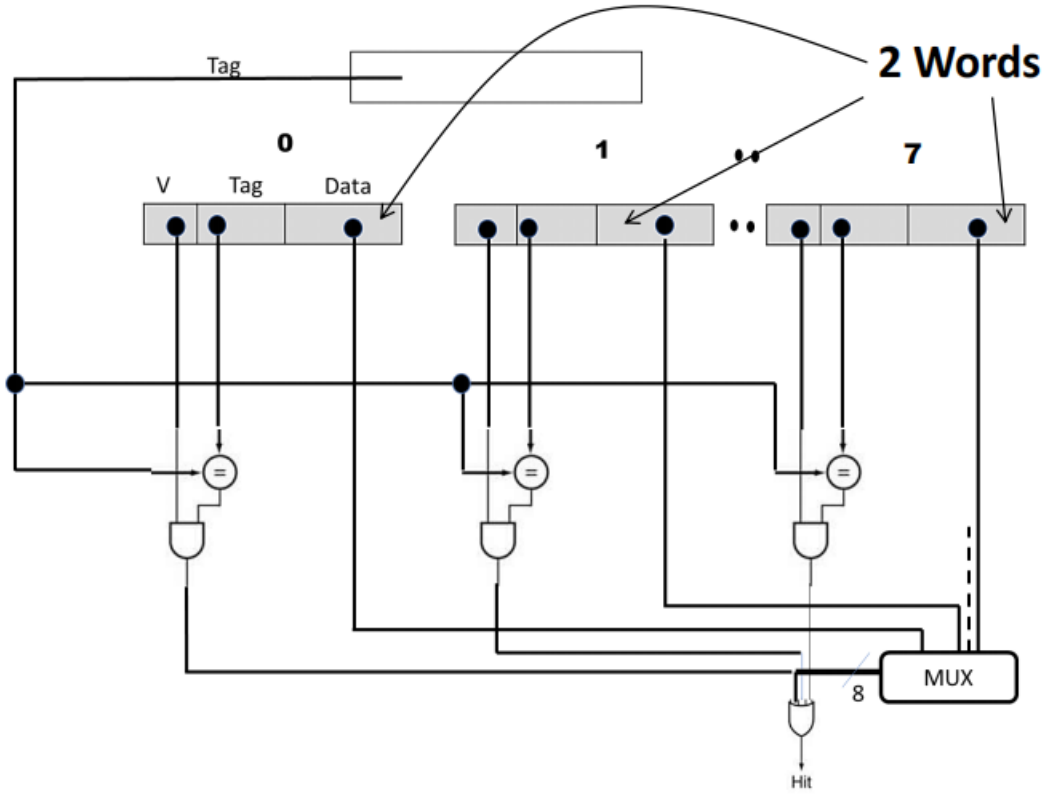
9.4 Trace the behavior of the cache from Exercise 9.3. Assume a true LRU replacement policy. For each reference, identify

- *the binary word address,*
- *the tag,*
- *the index,*
- *the offset,*
- *whether the reference is a hit or a miss*
- *the contents of the cache after each reference has been handled.*

Hex	Binary	Tag	Hit/Miss	Content
0x03	00000011	0x03	M	3
0xb4	10110100	0xb4	M	3, b4
0x2b	00101011	0x2b	M	3, b4, 2b
0x02	00000010	0x02	M	3, b4, 2b, 2
0xbe	10111110	0xbe	M	3, b4, 2b, 2, be

0x58	01011000	0x58	M	3, b4, 2b, 2, be, 58
0xbf	10111111	0xbf	M	3, b4, 2b, 2, be, 58, bf
0x0e	00001110	0x0e	M	3, b4, 2b, 2, be, 58, bf, e
0x1f	00011111	0x1f	M	b4, 2b, 2, be, 58, bf, e, 1f
0xb5	10110101	0xb5	M	2b, 2, be, 58, bf, e, 1f, b5
0xbf	10111111	0xbf	H	2b, 2, be, 58, e, 1f, b5, bf
0xba	10111010	0xba	M	2, be, 58, e, 1f, b5, bf, ba
0x2e	00101110	0x2e	M	be, 58, e, 1f, b5, bf, ba, 2e
0xce	11001110	0xce	M	58, e, 1f, b5, bf, ba, 2e, ce

9.5 Sketch the organization of a fully associative cache with two-word blocks and a total size of eight words. Your sketch should have a style similar to Figure 5.18, but clearly show the width of the tag and data fields.



9.6 Trace the behavior of the cache from Exercise 9.5. Assume an LRU replacement policy. For each reference, identify

- *the binary word address,*

- *the tag,*
- *the index,*
- *the offset,*
- *whether the reference is a hit or a miss,*
- *the contents of the cache after each reference has been handled.*

Hex	Binary	Tag	Offset	Hit/Miss	Content
0x03	00000011	0x01	1	M	[2,3]
0xb4	10110100	0x5a	0	M	[2,3], [b4,b5]
0x2b	00101011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	00000010	0x01	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	10111110	0x5f	0	M	[b4,b5], [2a,2b], [2,3], [be,bf]
0x58	01011000	0x2c	0	M	[2a,2b], [2,3], [be,bf], [58,59]
0xbf	10111111	0x5f	1	H	[2a,2b], [2,3], [58,59], [be,bf]
0x0e	00001110	0x07	0	M	[2,3], [58,59], [be,bf], [e,f]
0x1f	00011111	0x0f	1	M	[58,59], [be,bf], [e,f], [1e,1f]
0xb5	10110101	0x5a	1	M	[be,bf], [e,f], [1e,1f], [b4,b5]
0xbf	10111111	0x5f	1	H	[e,f], [1e,1f], [b4,b5], [be,bf]
0xba	10111010	0x5d	0	M	[1e,1f], [b4,b5], [be,bf], [ba,bb]
0x2e	00101110	0x17	0	M	[b4,b5], [be,bf], [ba,bb], [2e,2f]
0xce	11001110	0x67	0	M	[be,bf], [ba,bb], [2e,2f], [ce,cf]

9.7 Repeat Exercise 9.6 using MRU (most recently used) replacement.

Hex	Binary	Tag	Offset	Hit/Miss	Content
0x03	00000011	0x01	1	M	[2,3]
0xb4	10110100	0x5a	0	M	[2,3], [b4,b5]
0x2b	00101011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	00000010	0x01	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	10111110	0x5f	0	M	[b4,b5], [2a,2b], [2,3], [be,bf]
0x58	01011000	0x2c	0	M	[b4,b5], [2a,2b], [2,3], [58,59]
0xbf	10111111	0x5f	1	M	[b4,b5], [2a,2b], [2,3], [be,bf]
0x0e	00001110	0x07	0	M	[b4,b5], [2a,2b], [2,3], [e,f]
0x1f	00011111	0x0f	1	M	[b4,b5], [2a,2b], [2,3], [1e,1f]
0xb5	10110101	0x5a	1	H	[2a,2b], [2,3], [1e,1f], [b4,b5]
0xbf	10111111	0x5f	1	M	[2a,2b], [2,3], [1e,1f], [be,bf]
0xba	10111010	0x5d	0	M	[2a,2b], [2,3], [1e,1f], [ba,bb]
0x2e	00101110	0x17	0	M	[2a,2b], [2,3], [1e,1f], [2e,2f]
0xce	11001110	0x67	0	M	[2a,2b], [2,3], [1e,1f], [ce,cf]

9.8 Repeat Exercise 9.6 using the optimal replacement policy (i.e., the one that gives the lowest miss rate).

Hex	Binary	Tag	Offset	Hit/Miss	Content
0x03	00000011	0x01	1	M	[2,3]
0xb4	10110100	0x5a	0	M	[2,3], [b4,b5]
0x2b	00101011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	00000010	0x01	0	H	[2,3], [b4,b5], [2a,2b]
0xbe	10111110	0x5f	0	M	[2,3], [b4,b5], [2a,2b], [be,bf]
0x58	01011000	0x2c	0	M	[58,59], [b4,b5], [2a,2b], [be,bf]
0xbf	10111111	0x5f	1	H	[58,59], [b4,b5], [2a,2b], [be,bf]
0x0e	00001110	0x07	0	M	[e,f], [b4,b5], [2a,2b], [be,bf]
0x1f	00011111	0x0f	1	M	[1e,1f], [b4,b5], [2a,2b], [be,bf]

0xb5	10110101	0x5a	1	H	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xbf	10111111	0x5f	1	H	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xba	10111010	0x5d	0	M	[1e,1f], [ba,bb], [2a,2b], [be,bf]
0x2e	00101110	0x17	0	M	[1e,1f], [ba,bb], [2e,2f], [be,bf]
0xce	11001110	0x67	0	M	[1e,1f], [ba,bb], [2e,2f], [ce,cf]

10. Multilevel caching is an important technique to overcome the limited amount of space that a first-level cache can provide while still maintaining its speed. Consider a processor with the following parameters:

Base CPI, No Memory Stalls	Processor Speed	Main Memory Access Time	First-Level Cache Miss Rate per Instruction**	Second-Level Cache, Direct-Mapped Speed	Miss Rate with Second-Level Cache, Direct-Mapped	Second-Level Cache, Eight-Way Set Associative Speed	Miss Rate with Second-Level Cache, Eight-Way Set Associative
1.5	2 GHz	100 ns	7%	12 cycles	3.5%	28 cycles	1.5%

***First Level Cache miss rate is per instruction. Assume the total number of L1 cache misses*
(instruction and data combined) is equal to 7% of the number of instructions.

10.1 Calculate the CPI for the processor in the table using: 1) only a first-level cache, 2) a second-level direct mapped cache, and 3) a second-level eight-way set associative cache. How do these numbers change if main memory access time doubles? (Give each change as both an absolute CPI and a percent change.) Notice the extent to which an L2 cache can hide the effects of a slow memory.

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 (thus, 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).

10.2 It is possible to have an even greater cache hierarchy than two levels? Given the processor above with a second-level, direct-mapped cache, a designer wants to add a third-level cache that takes 50 cycles to access and will have a 13% miss rate. Would this provide better performance? In general, what are the advantages and disadvantages of adding a third-level cache?

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.

10.3 In older processors, such as the Intel Pentium or Alpha 21264, the second level of cache was external (located on a different chip) from the main processor and the first-level cache. While this allowed for large second-level caches, the latency to access the cache was much higher, and the bandwidth was typically lower because the second-level cache ran at a lower frequency. Assume a 512 KiB off-chip second level cache has a miss rate of 4%. If each additional 512 KiB of cache lowered miss rates by 0.7%, and the cache had a total access time of 50 cycles, how big would the cache have to be to match the performance of the second-level direct-mapped cache listed above?

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.