1. You have a 2-way set associative L1 cache that is 8KB, with 4-word cache lines. Writing data to L2 takes 10 cycles. You get the following sequence of writes to the cache -- each is a 32-bit address in hexadecimal:

```
0x1000 4096
0x1004 4100
0x1010 4112
0x11c0 4554
0x2000 8192
0x21c0 8640
0x3400 13312
0x3404 13316
0x3f00 16128
0x2004 8196
0x1004 4100
```

a. How many cache misses occur with an LRU policy?
b. How many cache misses occur with a most-recently used policy?
c. Would the miss-rate increase or decrease if the cache was the same size, but direct-mapped? Explain.
d. How long does a read-miss eviction take if the cache is write-back, write-allocate? What about a write-miss? (Assume cache line is dirty and Assume that writing and reading data to/from L2 takes 10 cycles.)
e. How long does a read-miss eviction take if the cache is write-through, write-allocate? What about a write-miss?

Answer:

Offset: 4 bits (for 16B lines) Index : 8 bits (for 256 lines) Tag : 20 bits

Access Pattern:

0x1000: Miss (index=0)
0x1004: Hit (same cache line)
0x1010: Miss (index=1)
0x11c0: Miss
0x2000: Miss (index=0, but second way)
0x21c0: Miss (second way)
0x3400: Miss
0x3404: Hit
0x3f00: Miss
0x2004: Hit
0x1004: Hit

a. There are 7 cache misses, all compulsory. No evictions.
b. Since there are no evictions, the eviction policy doesn't affect the number of cache misses.
c. If the cache was direct-mapped, you would have double the cache lines, so 1 extra index bit. This would cause 0x1000 and 0x2000 to still go to different cache lines, so it would not affect the miss-rate.
d. Given that, on a read miss, you need to read data from L2: 10 cycles. You need to write that somewhere in the cache. Assuming that line was dirty, you have to write that data back first: 10 cycles. You could potentially do both at the same time though. For a write miss, you write the old data back first (10 cycles), then write data to the cache line, and it stays dirty.
e. The read miss works the same way, except the cache line being evicted is clean, so it doesn't need to be written to L2. On a write miss, you are still evicting a clean cache line, so it doesn't need to be written to L2. However, since it is write-through, the new data needs to be written to L2 (10 cycles). Also, since it is write-allocate, the

new data needs to be written to L1 as well (which is fast, so it doesn't matter in terms of time - it does however cause an eviction, and potentially future conflict misses).

2. You have 3 cache designs for a 16-bit address machine.
C1:

```
Direct-mapped cache.
Each cache line is 1 byte.
10-bit index, 6-bit tag.
1 cycle hit time.
```

C2:

```
2-way set associative cache.
Each cache line is 1 word (4 bytes).
7-bit index, 7-bit tag.
2 cycle hit time.
```

C3:

```
fully associative cache with 256 cache lines.
Each cache line is 1 word.
14-bit tag.
5 cycle hit time.
```

a. What is the size of each cache?
b. How much space does each cache need to store tags?
c. Which cache design has the most conflict misses? Which has the least?
d. If someone told you the hit rate for the 3 caches is 50%, 70% and 90% but did not tell you which hit rate corresponds to which cache, which cache would you guess corresponded to which hit rate? Why?
e. Assuming the miss time for each is 20 cycles, what is the average service time for each? (`Service Time = (hit rate)*(hit time) + (miss rate)*(miss time)`).

Answer: 1. C1: 1024 1B lines = 1KB.
C2: 128 4B lines x 2 ways = 1KB
C3: 256 4B lines = 1KB

2. C1: 1024 x 6-bit tags = 6Kbit
C2: 256 x 7-bit tags = 1792 bit
C3: 256 x 14-bit tags = 3584 bit
3. C1 probably has the most conflict misses, since it is direct mapped. C3, because it is fully associative, can never have conflict misses.
4. Since the caches are the same size and the reason in the previous answer, C1 is 50%, C2 is 70%, and C3 is 90%.
5. Based on the previous answer, average service time: C1: 0.5*1 + 0.5*20 = 10.5 cycles
C2: 0.7*2 + 0.3*20 = 7.4 cycles
C3: 0.9*5 + 0.1*20 = 6.5 cycles

3. Consider an 8-line one-word-block direct-mapped cache initialized to all zeroes where the following sequence of word addresses are accessed: 1, 4, 5, 20, 9, 19, 4, 5, 6, and 9. Which of the following tables reflect the final tag bits of the cache?

	tag
0	1
1	4
2	5
3	20
4	9
5	19
6	6
7	0

A

	tag
0	0
1	9
2	3
3	19
4	4
5	5
6	6
7	8

B

	tag
0	1
1	1
2	1
3	2
4	0
5	0
6	0
7	0

C

	tag
0	0
1	0
2	0
3	19
4	4
5	5
6	6
7	0

D

	tag
0	0
1	1
2	0
3	2
4	0
5	0
6	0
7	0

E

Answer. First map the addresses to cache line numbers and tags where

line number = address mod 8

tag = floor(address / 8)

address:	1	4	5	20	9	19	4	5	6	9
line #:	1	4	5	4	1	3	4	5	6	1
tag:	0	0	0	2	1	2	0	0	0	1

So, figure (E) represents the final tag bits of the cache.

4. Consider a processor with a 16 Kbyte 4-way set associative cache, with 8-byte lines. Show how you will translate 32-bit address to find the data in cache. Show the cache set to which the following (hex) address falls: ABCED8F8. How many bits are needed for tags in this cache?

Answer. A 16K Byte cache with 8-byte line will contain 2048 lines and with 4-way set associativity, we will have 512 sets (each set with 4 lines). Thus we need 9-bits to locate a set, and 3-bits to locate a byte within a line (since each line has 8bytes). So we will divide a 32 bit address as follows:

20-bit Tag; 9-bit set ; 3-bit offset

Address ABCED8F8 = 1010 1011 1100 1110 1101 1000 1111 1000 yields 1 000 1111 1 = 287 as the set number (and byte zero in the line).

This leave the tag as ABCED = 1010 1011 1100 1110 1101

There are a total of 2048 lines and each line will have 20 bit tags. So the total number of bits for tags = 20*2048 = 40,960 bits.

5. Consider a cache with a line size of 64 bytes. Assume that on average 30% of the lines in the cache are dirty. A word consists of 8 bytes.

a. Assume there is a 3% miss rate (0.97 hit ratio). Compute the amount of main memory traffic, in terms of bytes per instruction for both write-through and write-back policies. Memory is read into cache one line at a time. However, for write-back, a single word can be written from cache to main memory.

b. Repeat part a for a 5% rate.

c. Repeat part a for a 7% rate.

d. What conclusion can you draw from these results?

Answer. a. Consider the execution of 100 instructions. Under **write-through**, this creates 200 cache references (168 read references and 32 write references). On average, the read references result in (0.03)*168 = 5.04 read misses.

For each read miss, a line of memory must be read in, generating 5.04*8 = 40.32 physical words of traffic. For write misses, a single word is written back, generating 32 words of traffic. **Total traffic**: 72.32 words.

For **write back**, 100 instructions create 200 cache references and thus 6 cache misses. Assuming 30% of lines are dirty, on average 1.8 of these misses require a line write before a line read. Thus, **total traffic** is (6 + 1.8) * 8 = 62.4 words. The traffic rate:
Write through = 0.7232 byte/instruction
Write back = 0.624 bytes/instruction

b. For write-through: [(0.05) * 168 * 8] + 32 = 99.2 => 0.992 bytes/instruction
For write-back: (10 + 3) * 8 = 104 # 0.104 bytes/instruction

c. For write-through: [(0.07) * 168 * 8] + 32 = 126.08 => 0.12608 bytes/instruction
For write-back: (14 + 4.2) * 8 = 145.6 => 0.1456 bytes/instruction

d. A 5% miss rate is roughly a crossover point. At that rate, the memory traffic is about equal for the two strategies. **For a lower miss rate, write-back is superior. For a higher miss rate, write-through is superior.**

6. Consider adding a new index addressing mode to the machine A such that the following code sequence

ADD R1, R1, R2 // R1 = R1 + R2
LW Rd, 0(R1) // load can be combined as
LW Rd, 0(R1 + R2)

(a) Assume that the load and store instructions are 10% and 20% of all the benchmarks considered, respectively, and this new addressing mode can be used for 5% of both of them. Determine the ratio of instruction count on the machine A before adding the new addressing mode and after adding the new addressing mode.

(b) Assume that adding this new addressing mode also increases the clock cycle by 3%, which machine (either machine A *before* adding the new addressing mode or machine A *after* adding the new addressing mode) will be faster and by how much?

Answer:

$$\text{(a)}\ \frac{IC_{old}}{IC_{new}} = \frac{IC_{old}}{IC_{old} \times (1 - 0.05 \times 0.3)} = 1.01523$$

$$\text{(b)}\ \text{speedup} = \frac{(1 - 0.5 \times 0.3) \times 1.03 \times 1}{1} = 1.01455$$

Machine A before adding the new addressing mode is faster by 1.01455 times.

7. (a) What are the two characteristics of program memory accesses that caches exploit?
(b) What are three types of cache misses?

Answer. (a) **Temporal locality:** if an item is referenced, it will tend to be referenced again soon.
Spatial locality: if an item is referenced, items whose addresses are close by will tend to be eferenced soon.
(b) **Compulsory misses:** a cache miss caused by the first access to a block that has never been in the cache.
Capacity misses: a cache miss that occurs because the cache even with fully associativity, can not contain all the block needed to satisfy the request.
Conflict misses: a cache miss that occurs in a set-associative or direct-mapped cache when multiple blocks compete for the same set.

8. Increasing associativity requires more comparators, as well as more tag bits per cache block. Assuming a cache of 4K blocks, a four-word block size, and a 32-bit address, find the total number of sets and the total number of tag bits for caches that are direct mapped, two-way and four-way set associative, and fully associative.

Answer:

Cache	The total number of sets	The total number of tag bits
Direct mapped	4K	(32 – 12 – 4) * 4K = 64K bits

Two-way set associative	4K / 2 = 2K	(32 – 11 – 4) * 4K = 68K bits
Four-way set associative	4K / 4 = 2K	(32 – 10 – 4) * 4K = 72K bits
Fully associative	1	(32 – 4) * 4K = 112K bits

9. Assume a five stage pipelined MIPS processor (i.e., instruction fetch, register read, ALU operation, data access, register write) and the following two sequences of instructions.

Instruction sequence		

a	lw $1, 40($6)
	add $6, $2, $2
	sw $6, 50($1)	

b	lw $5, -16($5)
	sw $5, -16($5)
	add $5, $5, $5	

(a) Assume there is no forwarding in this pipelined processor. Add *nop* instructions in above two sequences of instructions to eliminate the hazards.
(b) Assume there is full forwarding. Add *nop* instructions in above two sequences of instructions to eliminate the hazards

Answer: (a)

Instruction sequence		

a.	lw $1, 40($6)
	add $6, $2, $2
	nop
	nop
	sw $6, 50($1)
		Delay I3 to avoid RAW hazard on $1 and $6 from I1 and I2

b.	lw $5, -16($5)
	nop
	nop
	sw $5, -16($5)
	add $5, $5, $5
		Delay I2 to avoid RAW hazard on $5 from I1.

(b).

Instruction sequence		

a.	lw $1, 40($6)
	add $6, $2, $2
	sw $6, 50($1)
		No RAW hazard on $1 from I1 (forwarded)

b.	lw $5, -16($5)
	nop
	sw $5, -16($5)
	add $5, $5, $5
		Delay I2 to avoid RAW hazard on $5 from I1

10. A DMA controller transfers 16-bit words to memory using cycle stealing. The words are assembled from a device that transmits characters at a rate of 2400 characters per second (each character takes up one byte). The CPU is fetching and executing instructions at an average rate of 1 million instructions per second. By how much will the CPU be slowed down because of the DMA transfer? Do not consider any extra overhead.

Ans. The DMA combines (assembles) one word from two consecutive characters (bytes) so we get

2400 chars/s = 2400 bytes/s = 1200 words/s = 1200 W/s

If we assume that one CPU instruction is one word wide then: 1 million instructions/s = 1 million words/s = 10^6 W/s

So we have 1200 words received during one second and (10^6-1200) words processed by the CPU (while DMA is transferring a word, the CPU cannot fetch the instruction so we have to subtract the number of words transferred by DMA) and we get:

(1200 w/s) / (10^6 w/s - 1200 w/s) = 1200 / (10^6 - 1200)

= 1200 / 998800 = 0.0012014417...

= approx. 0.0012 = 0.12%

The CPU will be slowed down by 0.12%.

Some More Questions and Hints

1. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 60% of the time, measured as a percentage of the execution time when the enhanced mode is in use. [Recall that Amdahl's Law depends on the fraction of the original unenhanced execution time that could make use of enhanced mode. Thus, we cannot directly use this 60% measurement to compute speedup with Amdahl's Law.]

a. What is the speedup we have obtained from the enhanced mode (relative to the original)?

Ans. 6.4

b. What percentage of the original execution time has been converted to enhanced mode?

Ans. .937

2. Suppose a program segment consists of a purely sequential part which takes 25 cycles to execute, and an iterated loop which takes 100 cycles per iteration. Assume the loop iterations are independent, and cannot be further parallelized. If the loop is to be executed 100 times, what is the maximum speedup possible using an infinite number of processors (compared to a single processor)?

3. Consider a machine with a byte addressable main memory of 216 bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16-bit memory address divided into tag, line number, and byte number?

b. Into what line would bytes with each of the following addresses be stored?

0001 0001 0001 1011
1100 0011 0011 0100
1101 0000 0001 1101
1010 1010 1010 1010

Ans. [Line 3. Line 6. Line 3. Line 21.]

a. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

Ans. Bytes with addresses 0001 1010 0001 1000 through 0001 1010 0001 1111

b. How many total bytes of memory can be stored in the cache?

Ans. 256 bytes

4. Why is the tag also stored in the cache?

Ans. Because two words with two different memory addresses can be stored in the same place in the cache. The tag is used to distinguish between them.

5. Consider a memory system that uses a 32-bit address to address at the byte level, plus a cache that uses a 64-byte line size.

a. Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

b. Assume an associative cache. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

c. Assume a four-way set-associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in set, number of sets in cache, number of lines in cache, size of tag.

6. Consider a virtual memory system with the following characteristics:

- Page size of 16 KB = **214 bytes**
- Page table and page directory entries of 8 bytes per entry
- Inverted page table entries of 16 bytes per entry
- 31 bit physical address, byte addressed
- Two-level page table structure (containing a page directory and one layer of page tables)

a. What is size (in number of address bits) of the virtual address space supported by the above virtual memory configuration?

Ans. 36 bit virtual address space

b. What is the maximum required size (in KB, MB, or GB) of an inverted page table for the above virtual memory configuration?

Ans. 2MB in size

Do Yourself

Problem 1: (a) A one-level SRAM cache is n times faster than the main memory whose cycle time is T. Show that this cache will provide an average memory access speed up of $n/(1 + n – nh)$, where h is the hit rate of the cache. Also, show that irrespective of how fast the cache hardware is made the speed up cannot exceed $1/(1 – h)$.

(b) Show that the average memory access speed up has an upper bound $1/(m_1 \times m_2)$, where m_1 and m_2 are the miss rates of L1 and L2 caches, respectively.

Problem 2: To meet the data access time requirement of a computer its cache system is to have a 98% hit rate. However, to reduce the cost and to match the speeds of the processor and memory, we are forced to use a smaller size SRAM, which when placed in a 1-level cache system can only provide a 60% hit rate.

(a) If the main memory is 70 times slower than the SRAM cache hardware, find the average data access time for the 1-level cache with 60% hit rate, expressed in terms of T1, the cycle time of the SRAM cache. Show that this data access time is about twelve times longer than that if the hit rate was 98%, i.e., cache was larger but still had a fast cycle time matching the speed of the processor.

(b) We add a level-2 cache to bring the data access time to the required value. The cycle time of the L2 cache is 1.2 times slower than the L1 SRAM. Determine the minimum hit rate for the L2 cache.

Problem 3: For a two-level cache, the cycle times for L1 and L2 caches and main memory are 1, 10 and 200 clock cycles, respectively. Both L1 and L2 caches have the same hit rate h. What should h be so that the average data access time of this cache system is 2 cycles?

Problem 4: Consider a system with 4-way set associative cache of 256 KB. The cache line size is 8 words (32 bits per word). The smallest addressable unit is a byte, and memory addresses are 64 bits long. Show the division of the bits in a memory address and how they are used to access the cache.

a. Draw a diagram showing the organization of the cache and, using your answer from part (a), indicate how physical addresses are related to cache locations.
b. What percentage of the cache memory is used for tag bits?