3. You purchased an old IBM System whose model number is unknown so you do not have access to any manuals or spec sheets of the computer – except those listed below by a previous user:

1. 95% of all memory accesses are found in the cache.
2. Each cache block is two words, and the whole block is read on any miss.
3. The processor sends references to its cache at the rate of 10^9 words per second.
4. 25% of those references are writes.
5. Assume that the memory system can support 10^9 words per second, reads or writes.
6. The bus reads or writes a single word at a time (the memory system cannot read or write two words at once).
7. Assume at any one time, 30% of the blocks in the cache have been modified.
8. The cache uses write allocate on a write miss.

You are considering adding an IBM compatible peripheral to the system, and you want to know *how much of the memory system bandwidth is already used*. Calculate the percentage of memory system bandwidth used on the average in the two cases below. Be sure to state your assumptions.

3.1 The cache is Write-Through.

3.2 The cache is Write-Back.

We know:
* Miss rate = 0.05

* Block size = 2 words (8 bytes)

* Frequency of memory operations from processor = 10^9

* Frequency of writes from processor = $0.25 * 10^9$

* Bus can only transfer one word at a time to/from processor/memory

* On average 30% of blocks in the cache have been modified (must be written back in the case of the write back cache)

* Cache is write allocate

So:

Fraction of read hits = $0.75 * 0.95 = 0.7125$

Fraction of read misses = $0.75 * 0.05 = 0.0375$

Fraction of write hits = $0.25 * 0.95 = 0.2375$

Fraction of write misses = $0.25 * 0.05 = 0.0125$

6.1 Write through cache

- On a read hit there is no memory access
- On a read miss memory must send two words to the cache
- On a write hit the cache must send a word to memory

- On a write miss memory must send two words to the cache, and then the cache must send a word to memory

Thus:

Average words transferred = $0.7125 * 0 + 0.0375 * 2 + 0.2375 * 1 + 0.0125 * 3 = 0.35$

Average bandwidth used = $0.35 * 10^9$

Fraction of bandwidth used =

$[0.35 \times 10^9] / 10^9$

= 0.35 (1)

6.2 Write back cache

On a read hit there is no memory access

On a read miss:

1. If replaced line is modified then cache must send two words to memory, and then

memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

On a write hit there is no memory access

On a write miss:

1. If replaced line is modified then cache must send two words to memory, and then memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

Thus:

Average words transferred = $0.7125 * 0 + 0.0375 * (0.7 * 2 + 0.3 * 4) + 0.2375 * 0 + 0.0125 *$

$(0.7 * 2 + 0.3 * 4) = 0.13$

Average bandwidth used = $0.13 * 10^9$

Fraction of bandwidth used =

$0.13 \times 10^9/10^9$

= 0.13 (2)

Comparing 1 and 2 we notice that the write through cache uses more than twice the cache-memory bandwidth of the write back cache.

4A. Given a 4-way set associative cache of total size 2MB that has a 26-bit cache address and blocks of size 8KB each, answer the following questions.

4.1 How many bits in the "index" field of the cache address? Explain your answer

If the cache is 4-way set associative, then the number of sets is not 4. Rather, the number of sets is the number of rows in the cache divided by 4. Since this cache has 256 rows (2MB / 8KB), there are 64 sets. The answer is then: log_2(64) = 6 index bits

4.2 How many bits in the "offset" field of the cache address? Explain your answer

Observe: 8KB/block → 2^{13} bytes per block → $\log_2(2^{13})$ = 13 bits if byte-aligned. Assuming that the blocks are word-aligned, then we have Offset_bits = 13bits-2 bits = 11bits

4.3 How many bits in the "Tag" field of the cache address?

26 - 6 - 11 = 9 bits

4B. Mark whether the following modifications to cache parameters will cause each of the categories to increase, decrease, or whether the modification will have no effect. You can assume the baseline cache is set associative. Explain your reasoning. Assume that in each case the other cache parameters (number of sets, number of ways, number of bytes/line) and the rest of the machine design remain the same.

	compulsory misses	conflict misses	capacity misses
increasing number of sets	no effect block size is constant	decreases more sets are available for data, so there is less of a chance for two ops to collide and evict one another	decreases capacity increases
increasing number of ways	no effect block size is constant	decreases there are more ways available for data to be placed into	decreases capacity increases
increasing number of bytes per line	decreases more data is brought in on a given cache miss	no effect associativity and number of sets is constant	decreases capacity increases

5. Indicate if the following modifications (A,B,C) will cause each of the three metrics (three rightmost columns) to *increase*, *decrease*, or have *no effect*. Explain your reasoning

Assume the initial machine is pipelined. Also assume that any modification is done in a way that preserves correctness and maintains efficiency, but that the rest of the machine remains unchanged.

		Instructions/Program	CPI (Cycles/Instruction)	Circuit complexity
A	Replace the 2 operand ALU with a 3 operand one and add 3 operand register-register instructions to the ISA (for example, `ADD rs1,rs2,rs3,rd`)	**Decrease** The new instructions will replace any two-instruction sequence that accomplished the same, such as `ADD rs1, rs2, rd` `ADD rs3, rd, rd`	**The same** The ALU will perform the three-way addition in one cycle. Also acceptable increase because more RAW	**Increase** Three-operand ALU is more complex than a two-operand one
B	Use the same ALU for instructions and for incrementing the PC by 4	**The same** No difference to instructions	**Increase** All instructions now use the same ALU ALU operations now have to stall	**Decrease** Now there is one less adder/ALU cycle
C	Increase the number of user registers from 32 to 64	**The same or decrease** If the more registers enable the Compiler to avoid loads and stores, Decrease. Otherwise the same.	**The same** Does not affect the actions of each instructions	**Increase** More registers complicate the register file

ECE 6913, Computing Systems Architecture, Fall 2019

Quiz 2, November 20th 2019

Maximum time: 2.5 hours : **9:50 AM – 12:20 PM**

Closed Book, Closed Notes, Calculators allowed. RISC V Card provided

Devices with internet connectivity not allowed

This Test has 5 problems each with several parts. Please attempt all of them. Please show all work. Please write legibly

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.

`0xa2, 0xf4, 0xf5, 0xef, 0xaf`

1.1 For each of these references, *identify* (i) the binary word address, (ii) the tag, and (iii) the index given a **direct-mapped cache** with **32 one-word blocks**. Also, list (iv) whether each reference is a hit or a miss, assuming the cache is initially empty.

Hex Memory Reference	Binary Reference	Tag	Index	Hit / miss
0xa2	1010 0010	101	0 0010	m
0xf4	1111 0100	111	1 0100	m
0xf5	1111 0101	111	1 0101	m
0xef	1110 1111	111	0 1111	m
0xaf	1010 1111	101	0 1111	m

1.2 For each of these references, identify (i) the binary word address, (ii) the tag, (iii) the index, and (iv) the offset given a **direct-mapped cache** with **two-word blocks** and a **total size of 16 blocks**. Also, list (iv) if each reference is a hit or a miss, assuming the cache is initially empty.

Hex Memory Reference	Binary Reference	Tag	Cache Index	Block Offset	Hit / miss
0xa2	1010 0010	101	0001	0	m
0xf4	1111 0100	111	1010	0	m
0xf5	1111 0101	111	1010	1	h
0xef	1110 1111	111	0111	1	m
0xaf	1010 1111	101	0111	1	m

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 128 words of data*: C1 has 4-word blocks, C2 has 8-word blocks, and C3 has 16-word blocks.

Word Address	Binary Address	Tag bits in hex	Cache 1 block size = 4 word			Cache 2 block size = 8 word			Cache 3 block size = 16 word		
			index	offset	Hit/ Miss	index	offset	Hit/ Miss	index	offset	Hit/ Miss
0xa2	1010 0010	1	0 1000	10	m	0100	010	m	010	0010	m
0xf4	1111 0100	1	1 1101	00	m	1110	100	m	111	0100	m
0xf5	1111 0101	1	1 1101	01	h	1110	101	h	111	0101	h
0xef	1110 1111	1	1 1011	11	m	1101	111	m	110	1111	m
0xaf	1010 1111	1	0 1011	11	m	0101	111	m	010	1111	h

Cache 1: 32 blocks; **Cache 2:** 16 blocks; **Cache 3:** 8 blocks

1.4 How does the Miss rate depend on the size of the Block? Why? Can you list 2 opportunities to improve latency that accompanies increase in block size?

Miss rates lower as block size gets larger in number of words – due to spatial locality of data.

we may be able to hide some of the transfer time so that the miss penalty is effectively smaller. The easiest method for doing this, called **early restart**, is simply to resume execution as soon as the requested word of the block is returned, rather than wait for the entire block. Many processors use this technique for instruction access, where it works best. Instruction accesses are largely sequential, so if the memory system can deliver a word every clock cycle, the processor may be able to restart operation when the requested word is returned, with the memory system delivering new instruction words just in time. This technique is usually less effective for data caches because it is likely that the words will be requested from the block in a less predictable way, and the probability that the processor will need another word from a different cache block before the transfer completes is high. If the processor cannot access the data cache because a transfer is ongoing, then it must stall.

An even more sophisticated scheme is to organize the memory so that the requested word is transferred from the memory to the cache first. The remainder of the block is then transferred, starting with the address after the requested word and wrapping around to the beginning of the block. This technique, called **requested word first** or **critical word first**, can be slightly faster than early restart, but it is limited by the same properties that restrain early restart

2. Consider the following program and cache behaviors.

Data Reads per 1K instructions	Data Writes per 1K instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (Bytes)
300	150	0.5%	5%	128

2.1 Suppose a CPU with a write-through, write allocate 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.). *For a write-allocate policy, a write miss also makes a read request to RAM – please be sure to consider its impact on Read Bandwidth*

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
0.5% of these *instruction* accesses cause a cache ***Read*** miss (and subsequent memory request).
[0.5 instr/cycle] x [0.005 misses/instruction] = missed instructions/cycle
Assuming each miss requests one block and each block is 128 bytes [16 words with 8 bytes (64 bits) per word] , instruction accesses generate an average of
[0.5 instr/cycle] x [0.005 misses/instruction] x[128 bytes/miss] =
= 0.32 bytes/cycle of *read traffic*

Read Data bandwidth:
30% of instructions generate a ***read*** request from data memory.
[0.5 instr/cycle] x [0.3 Read Data Accesses/instruction] = [0.15 Read Data Accesses / cycle]
5% of these generate a cache miss;
[0.15 Read Data Accesses / cycle] x [0.05 misses / Read Data Access] = 0.0075 Read Misses/cycle
Assuming each miss requests one block and each block is 128 bytes [16 words with 8 bytes (64 bits) per word] ,
[0.0075 Read Misses/cycle] x [128 Bytes/block] x [1 block/miss] = 0.0075 x 128 Bytes/cycle
= 0.96 Bytes/cycle
Write Data bandwidth:
15% of instructions generate a ***write*** request into data memory.
[0.5 instr/cycle] x [0.15 Write Data Accesses/instruction] = [0.075 Write Data Accesses / cycle]
All of the words written to the cache must be written into Memory:
[0.075 Write Data Accesses / cycle] x [8 bytes/word] x [1 word/write-through] = 0.6 Bytes/cycle

For a *Write-allocate policy*, a Write miss also makes *a **read** request* to RAM
[0.5 inst/cycle] x [0.15 Write Data Accesses/instruction] x [0.05 misses/Write Data Access] x [128 Bytes/miss]
= 0.48 Bytes/cycle
Assuming each miss requests one Word (8 bytes) since this is a write-through cache *with only 1 word written per miss into memory*,
[0.00375 Write Misses/cycle] x [8 Bytes/word] x [1 word/miss] = 0.03 Bytes/cycle

Total Read Bandwidth
0.32 (Instruction memory) + **0.96** (data memory) + **0.48** (Write-miss in Write-through cache with Write Allocate) **Bytes/cyle = 1.76 Bytes/cycle**
Total Write Bandwidth:
0.6 Bytes/cycle + 0.03 Bytes/cycle = 0.63 Bytes/cycle

2.2 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?

The instruction and data ***read*** bandwidth requirement is the same: **1.76 Bytes/cycle**

the data ***write*** bandwidth requirement becomes

[0.5 inst/cycle] x [0.15 Write Data Accesses/instruction + 0.3 Read Data Accesses/instruction]
= 0.225 Accesses/cycle
[0.225 Accesses/cycle] x [0.05 misses /Access]
= 0.01125 misses/cycle
[0.01125 misses/cycle] x [0.3 blocks/miss]
= 0.003375 blocks/cycle
[0.003375 blocks/cycle] x [128 bytes/block] =
=0.432 bytes/cycle

Problem 1.

Your favorite Computer is described with the following features:
• 95% of all memory accesses are found in the cache.
• Each cache block is two words, and the whole block is read on any miss.
• The processor sends references to its cache at the rate of 10^9 words per second.
• 25% of those references are writes.
• Assume that the memory system can support 10^9 words per second, reads or writes.
• The bus reads or writes a single word at a time (the memory system cannot read or write two words at once).
• Assume at any one time, 30% of the blocks in the cache have been modified.
• The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and *you want to know how much of the memory system bandwidth is already used*.

(a) Calculate the percentage of memory system bandwidth used assuming the cache is Write Back.
(b) Calculate the percentage of memory system bandwidth used assuming the cache is Write Through.
Be sure to state your assumptions.

We know:

* Miss rate = 0.05

* Block size = 2 words (8 bytes)

* Frequency of memory operations from processor = 109

* Frequency of writes from processor = $0.25 * 109$

* Bus can only transfer one word at a time to/from processor/memory

* On average 30% of blocks in the cache have been modified (must be written back in the case of the write back cache)

* Cache is write allocate

So:

Fraction of read hits = $0.75 * 0.95 = 0.7125$

Fraction of read misses = $0.75 * 0.05 = 0.0375$

Fraction of write hits = $0.25 * 0.95 = 0.2375$

Fraction of write misses = $0.25 * 0.05 = 0.0125$

2 (b) Write through cache

- On a read hit there is no memory access
- On a read miss memory must send two words to the cache
- On a write hit the cache must send a word to memory

- On a write miss memory must send two words to the cache, and then the cache must send a word to memory

Thus:

Average words transferred = $0.7125 * 0 + 0.0375 * 2 + 0.2375 * 1 + 0.0125 * 3 = 0.35$

Average bandwidth used = $0.35 * 10^9$

Fraction of bandwidth used =

$[0.35 \times 10^9] / 10^9$

$= 0.35$ (1)

2(a) Write back cache

On a read hit there is no memory access

On a read miss:

1. If replaced line is modified then cache must send two words to memory, and then

memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

On a write hit there is no memory access

On a write miss:

1. If replaced line is modified then cache must send two words to memory, and then memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

Thus:

Average words transferred = $0.7125 * 0 + 0.0375 * (0.7 * 2 + 0.3 * 4) + 0.2375 * 0 + 0.0125 *$

$(0.7 * 2 + 0.3 * 4) = 0.13$

Average bandwidth used = $0.13 * 10^9$

Fraction of bandwidth used =

$0.13 \times 10^9/10^9$

$= 0.13$ (2)

Comparing 1 and 2 we notice that the write through cache uses more than twice the cache-memory bandwidth of the write back cache.

Problem 2.

One difference between a write-through cache and a write-back cache can be in the time it takes to write. During the first cycle, we detect whether a hit will occur, and during the second (assuming a hit) we actually write the data.

Let's assume that 50% of the blocks are dirty for a write-back cache. For this question, assume that the write buffer for the write through will never stall the CPU (no penalty). Assume a cache read hit takes 1 clock cycle, the cache miss penalty is 50 clock cycles, and a block write from the cache to main memory takes 50 clock cycles. Finally, assume the instruction cache miss rate is 0.5% and the data cache miss rate is 1%. Assuming that on average 26% and 9% of instructions in the workload are loads and stores, respectively, **estimate the performance of a write-through cache with a two-cycle write versus a write-back cache with a two-cycle write.**

CPU performance equation: CPUTime = IC * CPI * ClockTime

CPI = CPIexecution + StallCyclesPerInstruction

We know:

Instruction miss penalty is 50 cycles

Data read hit takes 1 cycle

Data write hit takes 2 cycles

Data miss penalty is 50 cycles for write through cache

Data miss penalty is 50 cycles or 100 cycles for write back cache

Miss rate is 1% for data cache (MRD) and 0.5% for instruction cache (MRI)

50% of cache blocks are dirty in the write back cache

26% of all instructions are loads

9% of all instructions are stores

Then: CPIexecution = 0.26 * 1 + 0.09 * 2 + 0.65 * 1 = 1.09

Write through

StallCyclesPerInstruction = MRI * 50 + MRD * (0.26 * 50 + 0.09 * 50) = 0.425

so: CPI = 1.09 + 0.425 = 1.515 (1)

Write back

StallCyclesPerInstruction = MRI * 50 + MRD * (0.26 * (0.5 * 50 + 0.5 * 100) + 0.09 * (0.5 * 50 + 0.5 * 100)) = 0.5125

so: CP I = 1.09 + 0.5125 = 1.6025 (2)

Comparing 1 and 2 we notice that the system with the write back cache is 6% slower.

Problem 3.

. Consider the following program and cache behaviors.

Data Reads per 1K instructions	Data Writes per 1K instructions	Instruction Cache Miss Rate	Data Cache Miss Rate	Block Size (Bytes)
300	150	0.5%	5%	128

Suppose a CPU with a write-through, write allocate 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.). *For a write-allocate policy, a write miss also makes a read request to RAM – please be sure to consider its impact on Read Bandwidth*

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
0.5% of these *instruction* accesses cause a cache ***Read*** miss (and subsequent memory request).
[0.5 instr/cycle] x [0.005 misses/instruction] = missed instructions/cycle
Assuming each miss requests one block and each block is 128 bytes [16 words with 8 bytes (64 bits) per word] , instruction accesses generate an average of
[0.5 instr/cycle] x [0.005 misses/instruction] x[128 bytes/miss] =
= 0.32 bytes/cycle of *read traffic*

Read Data bandwidth:
30% of instructions generate a ***read*** request from data memory.
[0.5 instr/cycle] x [0.3 Read Data Accesses/instruction] = [0.15 Read Data Accesses / cycle]
5% of these generate a cache miss;
[0.15 Read Data Accesses / cycle] x [0.05 misses / Read Data Access] = 0.0075 Read Misses/cycle
Assuming each miss requests one block and each block is 128 bytes [16 words with 8 bytes (64 bits) per word] ,
[0.0075 Read Misses/cycle] x [128 Bytes/block] x [1 block/miss] = 0.0075 x 128 Bytes/cycle
= 0.96 Bytes/cycle
Write Data bandwidth:
15% of instructions generate a ***write*** request into data memory.
[0.5 instr/cycle] x [0.15 Write Data Accesses/instruction] = [0.075 Write Data Accesses / cycle]
All of the words written to the cache must be written into Memory:
[0.075 Write Data Accesses / cycle] x [8 bytes/word] x [1 word/write-through] = 0.6 Bytes/cycle

For a *Write-allocate policy*, a Write miss also makes *a **read** request* to RAM
[0.5 inst/cycle] x [0.15 Write Data Accesses/instruction] x [0.05 misses/Write Data Access] x [128 Bytes/miss]
= 0.48 Bytes/cycle
Assuming each miss requests one Word (8 bytes) since this is a write-through cache *with only 1 word written per miss into memory*,
[0.00375 Write Misses/cycle] x [8 Bytes/word] x [1 word/miss] = 0.03 Bytes/cycle

Total Read Bandwidth
0.32 (Instruction memory) + **0.96** (data memory) + **0.48** (Write-miss in Write-through cache with Write Allocate) **Bytes/cyle = 1.76 Bytes/cycle**
Total Write Bandwidth:
0.6 Bytes/cycle + 0.03 Bytes/cycle = 0.63 Bytes/cycle

1. Students M, O & P are building custom hardware and compilers for their project

The following observations are known. CPI = 1

• Student O has built his design and it has a known execution time for a new program of 600 seconds.

• Student M proposes a single cycle data path, and plans to use a compiler that will generate 300 Billion dynamic instructions per execution.

• Student P proposes a pipelined data path, with a clock period of 1.2 ns. His compiler will generate 400 Billion instructions. Of these 400 Billion instructions, 25% are loads, and 20% are branches. 40% of all loads cause a 2-cycle load-use stall. The penalty for a mis-predicted branch is 5 cycles. The processor has full bypassing, and does not suffer from any other stalls.

How fast does M need to make his clock cycle in order to make the program run faster than O's?

$Tcycle * 300B \text{ instructions} * CPI = Tcycle * 300B * 1 < 600 \text{ secs}$

$Tcycle < 600/300B = 2ns$

2. Assume that you have a computer with 1 clock cycle per instruction (CPI=1) when all accesses to memory are in cache. The only accesses to data come from load and store instructions. Those accesses account for 25 % of the total number of instructions. Miss penalty is 50 clock cycles and miss rate is 5 %. Determine the speedup obtained when there is no cache miss compared to the case when there are cache misses

Given: **cp**: Processor cycles. **cw**: Wait cycles. **T**: Clock period. **IC**: Number of instructions. **CPI**: Cycles per instruction. $\mathbf{a_i}$: Amount of accesses produced by instructions. $\mathbf{a_d}$: Amount of accesses produced by data. **mr**: Miss rate. **mp**: Miss penalty.

In case of no misses:

$Tcpu = (cp + cw) \cdot T = (IC \cdot CPI + 0) \cdot T = IC \cdot 1 \cdot T = IC \cdot T$

Including misses:

$cw = IC \cdot (a_i + a_d) \cdot mr \cdot mp$

$= IC(1 + 1 \cdot 0.25) \cdot 0.05 \cdot 50 = IC \cdot 1.25 \cdot 0.05 \cdot 50 = 3.125$

So,

$Tcpu = (IC \cdot 1 + IC \cdot 3.125) \cdot T = 4.125 \cdot IC \cdot T$

$Speedup = S = 4.125 \cdot IC \cdot T / [IC \cdot T] = 4.125$

3. When Program X is run, the user CPU time is 3 seconds, the total wall clock time is 4 seconds, and the system performance is 10 MFLOP/sec. Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time. We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wall clock time, and the MFLOP/sec performance be now?

$CPU\ performance_B / CPU\ performance_A = CPU\ time_A / CPU\ time_B$

$6 = 3/CPU\ time_B$

User CPU Time = .5 seconds

Since the I/O time is unaffected by the performance increase, it still takes 1 second to do I/O. Therefore, it takes 1 + .5 = 1.5 seconds to run Program A on the faster CPU

Wall clock Time = 1.5 seconds

System Performance in MFLOPS =

Number of Floating-Point Operations * 10^6 / Wall clock Time

Old System Performance (10 MFLOP/sec) = #FLOP * $10^6/4$

#FLOP = 40 * 10^6

New System Performance = 40 * 10^6 /1.5

= 26.67 MFLOP/sec

5. Your favorite Computer is described with the following features:
• 95% of all memory accesses are found in the cache.
• Each cache block is two words, and the whole block is read on any miss.
• The processor sends references to its cache at the rate of 10^9 words per second.
• 25% of those references are writes.
• Assume that the memory system can support 10^9 words per second, reads or writes.
• The bus reads or writes a single word at a time (the memory system cannot read or write two words at once).
• Assume at any one time, 30% of the blocks in the cache have been modified.
• The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and *you want to know how much of the memory system bandwidth is already used*.

(a) Calculate the percentage of memory system bandwidth used assuming the cache is Write Back.
(b) Calculate the percentage of memory system bandwidth used assuming the cache is Write Through.
Be sure to state your assumptions.

We know:

* Miss rate = 0.05

* Block size = 2 words (8 bytes)

* Frequency of memory operations from processor = 109

* Frequency of writes from processor = 0.25 * 109

* Bus can only transfer one word at a time to/from processor/memory

* On average 30% of blocks in the cache have been modified (must be written back in the case of the write back cache)

* Cache is write allocate

So:

Fraction of read hits = 0.75 * 0.95 = 0.7125

Fraction of read misses = 0.75 * 0.05 = 0.0375

Fraction of write hits = 0.25 * 0.95 = 0.2375

Fraction of write misses = 0.25 * 0.05 = 0.0125

(a) Write through cache

- On a read hit there is no memory access
- On a read miss memory must send two words to the cache
- On a write hit the cache must send a word to memory
- On a write miss memory must send two words to the cache, and then the cache must send a word to memory

Thus:

Average words transferred = 0.7125 * 0 + 0.0375 * 2 + 0.2375 * 1 + 0.0125 * 3 = 0.35

Average bandwidth used = $0.35 * 10^9$

Fraction of bandwidth used =

$[0.35 \times 10^9] / 10^9$

= 0.35 (1)

(b) Write back cache

On a read hit there is no memory access

On a read miss:

1. If replaced line is modified then cache must send two words to memory, and then

memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

On a write hit there is no memory access

On a write miss:

1. If replaced line is modified then cache must send two words to memory, and then memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

Thus:

Average words transferred = 0.7125 * 0 + 0.0375 * (0.7 * 2 + 0.3 * 4) + 0.2375 * 0 + 0.0125 *

(0.7 * 2 + 0.3 * 4) = 0.13

Average bandwidth used = $0.13 * 10^9$

Fraction of bandwidth used =

$0.13 \times 10^9/10^9$

= 0.13 (2)

Comparing 1 and 2 we notice that the write through cache uses more than twice the cache-memory bandwidth of the write back cache.

2. Your favorite Computer is described with the following features:
• 95% of all memory accesses are found in the cache.
• Each cache block is two words, and the whole block is read on any miss.
• The processor sends references to its cache at the rate of 10^9 words per second.
• 25% of those references are writes.
• Assume that the memory system can support 10^9 words per second, reads or writes.
• The bus reads or writes a single word at a time (the memory system cannot read or write two words at once).
• Assume at any one time, 30% of the blocks in the cache have been modified.
• The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and *you want to know how much of the memory system bandwidth is already used*.

(a) Calculate the percentage of memory system bandwidth used assuming the cache is Write Back.
(b) Calculate the percentage of memory system bandwidth used assuming the cache is Write Through.
Be sure to state your assumptions.

We know:

* Miss rate = 0.05

* Block size = 2 words (8 bytes)

* Frequency of memory operations from processor = 109

* Frequency of writes from processor = 0.25 ∗ 109

* Bus can only transfer one word at a time to/from processor/memory

* On average 30% of blocks in the cache have been modified (must be written back in the case of the write back cache)

* Cache is write allocate

So:

Fraction of read hits = 0.75 ∗ 0.95 = 0.7125

Fraction of read misses = 0.75 ∗ 0.05 = 0.0375

Fraction of write hits = 0.25 ∗ 0.95 = 0.2375

Fraction of write misses = 0.25 ∗ 0.05 = 0.0125

2 (b) Write through cache

- On a read hit there is no memory access
- On a read miss memory must send two words to the cache
- On a write hit the cache must send a word to memory
- On a write miss memory must send two words to the cache, and then the cache must send a word to memory

Thus:

Average words transferred = $0.7125 * 0 + 0.0375 * 2 + 0.2375 * 1 + 0.0125 * 3 = 0.35$

Average bandwidth used = $0.35 * 10^9$

Fraction of bandwidth used =

$[0.35 \times 10^9] / 10^9$

$= 0.35$ (1)

2(a) Write back cache

On a read hit there is no memory access

On a read miss:

1. If replaced line is modified then cache must send two words to memory, and then

memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

On a write hit there is no memory access

On a write miss:

1. If replaced line is modified then cache must send two words to memory, and then memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

Thus:

Average words transferred = $0.7125 * 0 + 0.0375 * (0.7 * 2 + 0.3 * 4) + 0.2375 * 0 + 0.0125 *$

$(0.7 * 2 + 0.3 * 4) = 0.13$

Average bandwidth used = $0.13 * 10^9$

Fraction of bandwidth used =

$0.13 \times 10^9/10^9$

$= 0.13$ (2)

Comparing 1 and 2 we notice that the write through cache uses more than twice the cache-memory bandwidth of the write back cache.

3. Assume we have a computer where the CPI is 1.0 when all memory accesses (including data and instruction accesses) hit in the cache. The cache is a unified (data + instruction) cache of size 256 KB, 4-way set associative, with a block size of 64 bytes. The data accesses (loads and stores) constitute 50% of the instructions. The unified cache has a miss penalty of 25 clock cycles and a miss rate of 2%. Assume 32-bit instruction and data addresses.

3.1 What are the Number of bits used for block offset?
Block size = 64 bytes, so 6 bits decode one of 64 bytes in a Block. Number of bits used for Block offset = 6

3.2 What are the Number of sets in the cache?
256KB /(64B x 4way) = 1K sets

3.3 What are the Number of bits for the cache index?
bits for the cache index pick one of all available sets ➔10 bits

3.4 What are the Number of bits for the tag?
Address has a total of 32 bits of which 6 bits used for Block offset and 10 bits used for the cache index. So, 32 – 16 = 16 bits for the Tag

3.5 Calculate the number of Stall Cycles per instruction
CPI for a computer that always hits = 1
Stall_Cycles_per_instruction = [Memory accesses per instr] x MR x MP
Memory accesses per instr. = 1 (for Instruction) + 0.5 (for Data) = 1.5

SCPI= 1.5 mem access/instruction x 0.02 misses/access x 25 cycles = 0.75 cycles/instruction

3.6 How much faster would the computer be if all memory accesses were cache hits?

1.75/1 = 1.75 faster

7. In general, cache access time is proportional to capacity. Assume that main memory accesses take 100 ns and that 40% 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	4KiB	10%	0.75ns
P2	8KiB	7%	1.0ns

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

P1: 1.33GHz

P2: 1GHz

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

P1: AMAT = 0.75ns + 100ns x 10% = 10.75ns

P2: AMAT = 1ns + 100ns x 7% = 8ns

7.3 Assuming a base CPI of 1.5 without any memory stalls, *what is the total CPI for P1 and P2? Which processor is faster?* (When we say a "base CPI of 1.5", we mean that 2 instructions complete in three cycles, unless either the instruction access or the data access causes a cache miss.)

Total CPI for P1= 1.5 + [(100x0.1/0.75ns)] x 0.4 = 6.83

Total CPI or P2 = 1.5 + [(100 x 0.07)/1.0] x 0.4 = 4.3

7.4 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
4MiB	90%	10ns

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

P1 AMAT: 0.75ns + 0.1 x (10ns + 0.9 x 100ns) = 10.75 ns

AMAT is the same

7.5 Assuming a base CPI of 1.5 without any memory stalls, *what is the total CPI for P1 with the addition of an L2 cache?*

CPI = 1.5 + 0.4 x [0.1(10 + 0.9 x 100)/0.75] = 6.83

8. Mark whether the following modifications to cache parameters will cause each of the categories to increase, decrease, or whether the modification will have no effect. You can assume the baseline cache is set associative. Explain your reasoning. Assume that in each case the other cache parameters (number of sets, number of ways, number of bytes/line) and the rest of the machine design remain the same

	compulsory misses	conflict misses	capacity misses
increasing number of sets	no effect block size is constant	decreases more sets are available for data, so there is less of a chance for two ops to collide and evict one another	decreases capacity increases
increasing number of ways	no effect block size is constant	decreases there are more ways available for data to be placed into	decreases capacity increases
increasing number of bytes per line	decreases more data is brought in on a given cache miss	no effect associativity and number of sets is constant	decreases capacity increases