Problem 1

We know:

* Miss rate = 0.1
* Block size = 4 words (16 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 50% of blocks in the cache have been modified (must be written back in the case of the write back cache)
* Cache is write allocate

Problem 1.

Your favorite Computer is described with the following features:
• 90% of all memory accesses are found in the cache.
• Each cache block is 4 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, 40% 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.

So:

Fraction of read hits = 0.75 ∗ 0.9 = 0.675
Fraction of read misses = 0.75 ∗ 0.1 = 0.075
Fraction of write hits = 0.25 ∗ 0.9 = 0.225
Fraction of write misses = 0.25 ∗ 0.1 = 0.025

2 (b) Write through cache

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

Thus:

Average words transferred = 0.675 ∗ 0 + 0.075 ∗ 4 + 0.225 ∗ 1 + 0.025 ∗ 5 = 0.65
Average bandwidth used = 0.65 ∗ 10^9
Fraction of bandwidth used = [0.65 x 10^9] / 10^9 = 0.65

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 four words to memory, and then memory must send four words to the cache
2. If replaced line is clean then memory must send four 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 four words to memory, and then memory must send four words to the cache
2. If replaced line is clean then memory must send four words to the cache

Thus:

Average words transferred = 0.675 ∗ 0 + 0.075 ∗ (0.4 ∗ 8 + 0.6 ∗ 4) + 0.225 ∗ 0 + 0.025 ∗ (0.4 ∗ 8 + 0.6 ∗ 4) = 0.56
Average bandwidth used = 0.56 ∗ 10^9
Fraction of bandwidth used =0.56 x $10^9/10^9$ = 0.56

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

Problem 2

CPU performance equation: CPU Time = IC ∗ CPI ∗ Clock Time
CPI = CPI execution + Stall Cycles Per Instruction

We know:
Instruction miss penalty is 100 cycles
Data read hit takes 1 cycle
Data write hit takes 2 cycles
Data miss penalty is 100 cycles for write through cache
Data miss penalty is 150 cycles or 100 cycles for write back cache
Miss rate is 2% for data cache (MRD) and 0.5% for instruction cache (MRI)
30% of cache blocks are dirty in the write back cache
25% of all instructions are loads
10% of all instructions are stores

Then: CPI execution = $0.25 * 1 + 0.1 * 2 + 0.65 * 1 = 1.1$

Write through
Stall Cycles Per Instruction = $MRI * 100 + MRD * (0.25 * 100 + 0.1 * 100) = 1.2$
so: $CPI = 1.1 + 1.2 = 2.3$

Write back
Stall Cycles Per Instruction = $MRI * 100 + MRD * (0.25 * (0.3 * 150 + 0.7 * 100) +$
$0.1 * (0.3 * 150 + 0.7 * 100)) = 1.405$
so: CP I = 1.1 + 1.405 = 2.405

Comparing 1 and 2 we notice that the system with the write back cache is 4.6% slower.

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 30% 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 100 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 2%. Assuming that on average 25% and 10% 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.**

.

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)
400	200	0.5%	2%	64

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*

Problem 3

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 [64 bytes/miss] = 0.16 bytes/cycle of *read traffic*

Read Data bandwidth:

40% of instructions generate a ***read*** request from data memory.
[0.5 instr/cycle] x [0.4 Read Data Accesses/instruction] = [0.2 Read Data Accesses / cycle]
2% of these generate a cache miss;
[0.2 Read Data Accesses / cycle] x [0.02 misses / Read Data Access] = 0.004 Read Misses/cycle
Assuming each miss requests one block and each block is 128 bytes [16 words with 8 bytes (64 bits) per word] ,
[0.004 Read Misses/cycle] x [64 Bytes/block] x [1 block/miss] = 0.004 x 64 Bytes/cycle
= 0.256 Bytes/cycle

Write Data bandwidth:

20% of instructions generate a ***write*** request into data memory.
[0.5 instr/cycle] x [0.2 Write Data Accesses/instruction] = [0.01 Write Data Accesses / cycle]
All of the words written to the cache must be written into Memory:
[0.01 Write Data Accesses / cycle] x [8 bytes/word] x [1 word/write-through] = 0.8 Bytes/cycle

For a *Write-allocate policy*, a Write miss also makes *a **read** request* to RAM
[0.5 inst/cycle] x [0.2 Write Data Accesses/instruction] x [0.02 misses/Write Data Access] x [64 Bytes/miss]
= 0.128 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*,
[0.002 Write Misses/cycle] x [8 Bytes/word] x [1 word/miss] = 0.016 Bytes/cycle

Total Read Bandwidth
0.16 (Instruction memory) + **0.256** (data memory) + **0.128** (Write-miss in Write-through cache with Write Allocate) **Bytes/cyle = 0.544 Bytes/cycle**

Total Write Bandwidth:
0.8 Bytes/cycle + 0.016 Bytes/cycle = 0.816 Bytes/cycle

Problem 4.

Consider the following RISC V Instruction sequence executing in a 5-stage pipeline:

```
and   x13, x12, x11
ld    x10, 0(x13)
ld    x11, 8(x10)
add   x12, x10, x11
subi  x13, x12, 16
```

4.1 Identify all of the data hazards and their resolution with NOPs assuming no forwarding or hazard detection hardware is being used

4.2 If there is forwarding, for the first seven cycles during the execution of this code, *specify which signals are asserted in each cycle by hazard detection and forwarding units* in Figure below.

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

4.1 Hazards identified:

and X_{13}, X_{12}, X_{11}

ld $X_{10}, 0(X_{13})$ — EX 1st

ld $X_{11}, 8(X_{10})$ — MEM 1st

add X_{12}, X_{10}, X_{11} — MEM 1st and 2nd.

subi $X_{13}, X_{12}, 16$ — EX 1st

↓

and X_{13}, X_{12}, X_{11}

NOPs

NOPs

ld $X_{10}, 0(X_{13})$.

NOPs

NOPs

ld $X_{11}, 8(X_{10})$

NOPs

NOPs

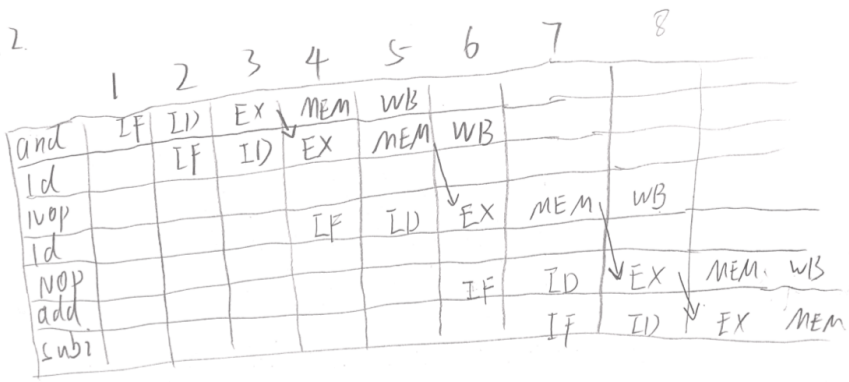
add X_{12}, X_{10}, X_{11}

NOPs

NOPs

subi $X_{13}, X_{12}, 16$

4.2



#	A	B
1	X	X
2	X	X
3	0	0
4	2	X
5	X	0
6	1	X
7	X	