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

3. Assume that there are no pipeline stalls and that the breakdown of executed instructions is as follows for the processor shown in the Figure below:

add	addi	not	beq	lw	sw
17%	18%	5%	25%	25%	10%

3.1 In what fraction of all cycles is the shift-left 2 unit used?

The shift left-2 unit is used only for branch instructions: 25%

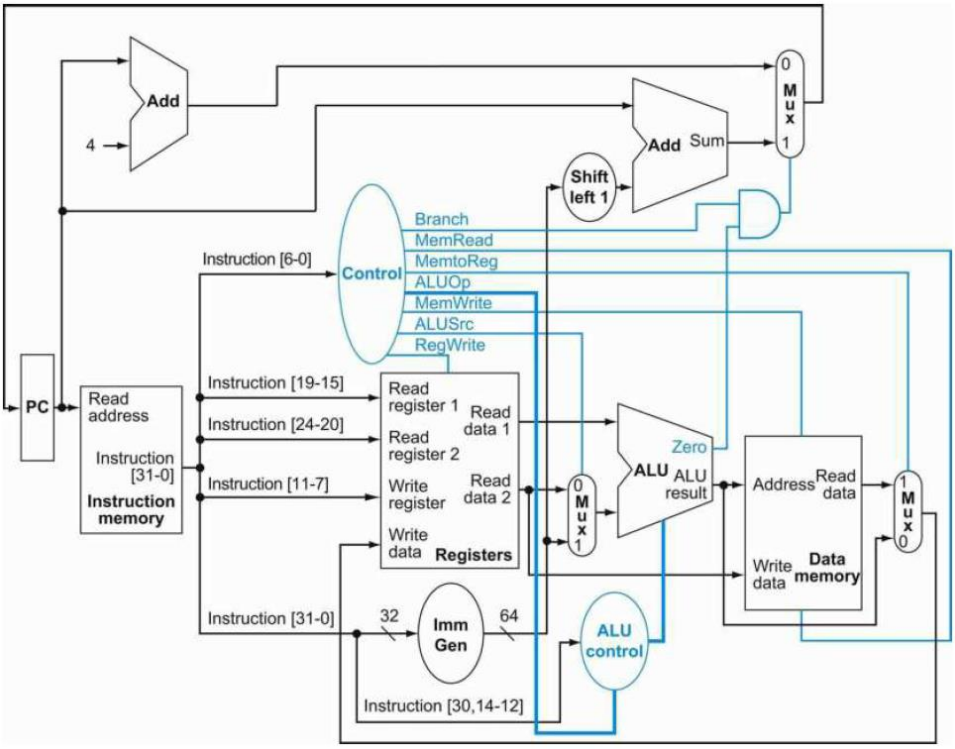
3.2 In what fraction of all cycles is the input of the sign-extend circuit needed?

addi, beq, lw, sw: 18 + 25 + 25 + 10 = 78%

3.3 What is the utilization of the *second* input data port of the ALU?

The second input port of the ALU is used in all instructions except the unconditional branch instruction (jal for example) where the ALU does not need to determine if a branch is to be taken.

add, addi, not, beq, lw, sw = 100%



4. This problem explores energy efficiency and its relationship with performance. The parts of this problem assume the following energy consumption for activity in Instruction memory, Registers, and Data memory. You can assume that the other components of the datapath consume a negligible amount of energy. ("Register Read" and "Register Write" refer to the register file only.)

I-Mem	1 Register Read	Register Write	D-Mem Read	D-Mem Write
140pJ	70pJ	60pJ	140pJ	120pJ

Assume that components in the datapath have the following latencies. You can assume that the other components of the datapath have negligible latencies.

I-Mem	Control	Register Read or Write	ALU	D-Mem Read or Write
200 ps	150 ps	90 ps	90 ps	250 ps

4.1 How much energy is spent to execute an **addi** instruction in a single-cycle design and in the five-stage pipelined design

I-Mem is read, two registers are read, and a register is written
We have: 140pJ + 2*70pJ + 60j = 340pJ

4.2 How much energy is spent to execute a **lw** instruction in a single-cycle design

140pJ + 2*70pJ + 60pJ + 140pJ = 480pJ

4.3 How much energy is spent to execute a **beq** instruction in a single-cycle design

I-Mem + 2 registers = 140pJ + 2 * 70pJ = 280pJ