1. (25 points) Show the contents of the three caches below for the specified address stream.
Three 4-entry, LRU caches are shown below, with the most recently used entry at the top of each set. They are fully-associative (FA), direct-mapped (DM), and 2-way set-associative (SA). Each cache has a block/line size of 1 byte address. The DM and SA caches use the modulo (LSB-based) indexing function discussed in the class. Show how the contents of the caches change with the access pattern 0, 1, 2, 3, 4, 0, 4, 0 ,2 by writing the memory address in each location of the cache at each time step and circling whether the access is a hit or miss.

Cycle 0: Address 0

FA		DM		SA	
0		0		0	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 1: Address 1

FA		DM		SA	
0		0		0	
1		1			
				1	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 2: Address 2

FA		DM		SA	
0		0		0	
1		1		2	
2		2		1	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 3: Address 3

FA		DM		SA	
0		0		0	
1		1		2	
2		2		1	
3		3		3	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 4: Address 4

FA		DM		SA	
1		4		2	
2		1		4	
3		2		1	
4		3		3	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 5: Address 0

FA		DM		SA	
2		0		4	
3		1		0	
4		2		1	
0		3		3	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 6: Address 4

FA		DM		SA	
2		4		0	
3		1		4	
0		2		1	
4		3		3	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 7: Address 0

FA		DM		SA	
2		0		4	
3		1		0	
4		2		1	
0		3		3	
Hit	Miss	Hit	Miss	Hit	Miss

Cycle 8: Address 2

FA		DM		SA	
3		0		0	
4		1		2	
0		2		1	
2		3		3	
Hit	Miss	Hit	Miss	Hit	Miss

Extra

FA		DM		SA	
Hit	Miss	Hit	Miss	Hit	Miss

2. (25 points)

A. (10 pts) Suppose you had a computer that, on average, exhibited the following properties on the programs that you run:

Instruction miss rate: 1.5%
Data miss rate: 4.0%
Percentage of memory load/store instructions: 30%
Miss penalty: 70 cycles

There is no penalty for a cache hit (i.e, the cache can supply the data as fast as the processor can consume it). Assume also that a cache-block is one word.
You want to upgrade the computer, and your budget will allow one of the following options:

- Get a new processor that is twice as fast as your current computer. The new processor's cache is twice as fast too, so it can keep up with the processor.
- Get a new memory that is twice as fast.

Which is a better choice? Explain fully, showing a quantitative comparison between the two options.

① $\frac{1}{2}(1 + 1.5\% \times 140 + 30\% \times 4\% \times 140)$

$= \frac{1}{2}(1 + 3.78)$

$= 2.39.$

② $1 + 1.5\% \times 35 + 30\% \times 4\% \times 35$

$= 1.945.$

So, new memory is better.

B. (15 pts) Calculate the effective CPI (taking into account both instruction execution and memory access) for the following unified instruction and data cache.

Instruction execution CPI = 1.0
Percent of instructions that are loads or stores = 33%
Cycles to access main memory = 100
Cycles to access cache = 2
Miss rate for cache 2.0%

(a). What is the total memory access rate? (memory accesses per cycle)

1 / 100

(b). How many cycles per instruction are spent handling misses?

2% × 100 + 0.33 × 2% × (100) / 2+100 / 33% × 2% × 100

(c). How many cycles per instruction are spent handling hits?

2 × 98% × 33% / 2 / 2 × 98% × 33% + 2 × 98%

(d). What is the effective CPI for the system with the cache?

1 + 2 + 2% × 100 + 30% × (2 + 2% × 100)
= 1 + 2 + 2 + 0.3 × (2 + 2)
= 5 + 30% × 4
= 6.2

2 + 2% × 100 + 30% × 2% × 100
= 2 + 2 + 0.6
= 4.6

3. (20 points)
Processors can exploit instruction-level parallelism using static compiler analysis, dynamic detection, and hybrids of the two. Very Large Instruction Word (VLIW) machines emphasize static analysis, while out-of-order superscalar machines emphasize dynamic detection. Many recent systems use both static analysis and dynamic detection to maximize instruction-level parallelism.

A. (5 pts) Briefly describe how a VLIW machine works and the role of static analysis.

VLIW machine bound multi instructions and execute them together.
static analysis can use compiler to figure out which instruction can execute in parallel with out data dependent.

B. (5 pts) Briefly describe how an out-of-order superscalar works and the role of dynamic detection.

OOO can execute multi instructions together and change the execution order.
dynamic detection detect which instruction is ready to issue dynamicly during the execution time

C. (5 pts) How can dynamic detection help improve the performance of a VLIW machine?

dynamic detection can tell the VLIW machine which instruction is ready to put into the slot during the execution time.

D. (5 pts) How can static analysis help improve the performance of an out-of-order superscalar machine?

static analysis can detect which instruction has data dependent during the compiler time. That information is needed by the OOO superscalar.

4. (30 points) Achieving parallelism in the execution of instructions, especially long-latency instructions, is a key to achieving higher performance in modern single-core and multi-core machines. Since arithmetic operations, including common floating-point operations, can be implemented with small latencies, memory operations are the dominant form of long-latency operations that occur in most common applications.

A. (10 pts) Techniques to achieve the parallel execution of arithmetic instructions have some limitations when they are applied to achieving memory-level parallelism — the parallel (overlapped) execution of memory instructions. What are these limitations? Discuss ways of enhancing these instruction-level parallelism techniques to make them more amenable achieving at achieving memory-level parallelism. What other techniques can you propose to improve memory level parallelism? Describe them, explain how they improve memory-level parallelism, and discuss their pros and cons.

If they access to the same location, the memory instructions cannot run in parallel.

We can do compiler optimize to re-assign the memory instruction, and put the memory instruction which access to different location together. So that we can increase the memory level parallelism.

B. (10 pts) Can there be some negative interactions between optimizing memory-level parallelism and optimizing memory-level locality (e.g., row-buffer hit rates).

memory level parallelism will prefer the program access different Banks, but that will cost low memory locality.
If we put all the memory access close, we cannot access the memory in parallel.

C. (10 pts) How can compiler and operating system help in improving memory-level parallelism? Give specific examples.

Compiler and OS can manage the instructions and reorder them, So that memory instructions access to different location will be put together, to execute at the same time period.