CS 533 **Name:** _____
Spring 2024
Homework 1 **NetID:** _____
Assigned: 1/25/2024
Due: 2/13/2024 **UIN:** _____

This homework contains 9 pages (including this cover page) and 8 questions.
Please clearly print or typeset your answers.
Note: up to 10 points (out of 100) may be deducted for illegible submissions

Problem Breakdown

Question	Points	Score
1	10	
2	12	
3	16	
4	12	
5	8	
6	12	
7	20	
8	10	
Total:	100	

1. (10 points) **Parallel Performance**

   Amdahl's law assumes that the sequential portion of execution cannot be sped up and that the parallel portion can be indefinitely parallelized. In more general models, both assumptions can be relaxed: some parallel segments might have limited parallelism and there are techniques to speed up the execution of sequential sections (as we will see later in the course).

   Assume you have a multi-core processor with n Cores. You can configure the system before running a program so that r cores work together as a single larger r-Core that can run sequential code $\sqrt{r}$ times as fast as a primitive core; the resulting system therefore will have one r-Core and $n - r$ primitive cores.

   You have an application that runs in three phases. Phase 1 (20% of computation) is sequential; Phase 2 (50% of computation) is linearly and indefinitely parallelizable while Phase 3 (30% of computation) can be linearly parallelized up to 3 ways. Assume that parallelization adds no overhead.

   (a) For the application on a system with one r-Core and $n - r$ primitive cores ($n \geq 1, n \geq r$), give the general formula for the speedup over sequential execution on a single primitive core.

   (b) Find the optimal r that offers the highest possible speedup for the following systems: $n = 2, n = 4$ and $n = 16$.

2. (12 points) **Cache Coherence vs. Memory Consistency**

   (a) Define the concept of coherence.

   (b) Define the concept of memory consistency model.

   (c) What are the differences between the two?

   (d) Are the ideas of coherency and consistency relevant to a uniprocessor system? Why?

   (e) Can a multithreaded program tell if it is running on a cache coherent multiprocessor machine or a uniprocessor machine with preemptive scheduling? The program should only decide based on the values returned by its loads.

   (f) Can a multithreaded program tell if the hardware is not SC by observing the values returned by the loads? Is it possible to decide whether the hardware is TSO in this way?

3. (16 points) **Coherence Example**

Show the sharing state of the cache line in different processors after each access in the following trace under Firefly and Illinois (MESI) protocols. Also show the number of cycles spent on coherence action for each access. How many total bus cycles are spent performing coherence?

	Illinois (MESI)				Firefly			
	P1	P2	P3	Cycles	P1	P2	P3	Cycles
	-	-	-		-	-	-	
1: Processor 1 reads								
2: Processor 1 writes								
3: Processor 2 writes								
4: Processor 1 reads								
5: Processor 3 reads								
6: Processor 3 writes								
7: Processor 1 reads								
8: Processor 3 reads								
9: Processor 1 writes								
10: Processor 2 writes								
11: Processor 3 reads								

Initially, the line is not present in any cache, indicated by a "-". Assume that coherence operations in a snoopy system have the following costs in bus cycles:

- A read request satisfied by main memory takes 80 cycles.
- A read request satisfied by another cache takes 30 cycles. If a writeback to main memory and/or invalidations must be performed in conjunction with the read request, the combined operation still takes 30 cycles.
- An invalidation broadcast takes 8 cycles.
- An update broadcast takes 34 cycles. It updates both the caches and main memory.

4. (12 points) **Directory Coherence Protocols**

(a) Give examples of at least two commercial processors that use directory based coherence schemes.

(b) The Compute Express Link (CXL) [1] technology expands the cache coherency to memories of devices connected in PCIE. Does it use directory-based or snoopy coherence protocol? Explain.

(c) Consider a 128-processor shared memory system using Illinois coherence with 32GB of memory per node. The cache block size is 64B.

(i) What is the directory memory overhead for Dir_N (full bit vector scheme)?

(ii) What is the directory memory overhead for Dir_4CV?

(iii) At how many processors does the Dir_N overhead become 100%?

(iv) At how many processors does the Dir_4CV overhead become 100%?

(d) Coherence Protocol implementations in real systems are notoriously hard to verify. This is because the many transient states that need to be included in actual implementations exponentially increase the number of transitions which must be checked.

(i) Why are there are extra transient states in real coherence protocols?

(ii) Give an example of a coherence protocol verifier. Describe at a high level how the verifier works.

[1] https://arxiv.org/abs/2306.11227

5. (8 points) **Sharing Patterns**

   (a) Which coherence protocol would you prefer in each of the following situations: Illinois (MESI) or Firefly? Why?

   (i) Multiprogramming workload with no sharing

   (ii) Producer-consumer through a FIFO buffer

   (iii) Producer-consumer where producer writes the consumed location multiple times before a consumer reads

   (iv) Physics simulation with N particles

   (b) What is the purpose of the "exclusive" state in the Firefly and Illinois (MESI) protocols? What sharing patterns benefit from this state?

6. (12 points) **Understanding Memory Consistency**

   (a) The memory consistency model of a system is fully determined by the processor hardware architecture. Do you agree or disagree with this statement? Explain.

   (b) Which of the following does the memory model determine? Choose all that apply.

   - ◯ Whether the compiler can hoist a load above an earlier store that writes to a different address
   - ◯ The number of cycles required to execute each memory instruction
   - ◯ Whether the memory hierarchy can include write-back caches
   - ◯ Whether the memory hierarchy can have multiple levels of caches
   - ◯ Whether a processor can read its own writes before any other processor sees them
   - ◯ Whether the interconnect may reorder messages

   (c) Of the four memory order relaxations listed in the Adve-Gharachorloo tutorial (ST $\rightarrow$ LD, ST $\rightarrow$ ST , LD $\rightarrow$ ST and LD $\rightarrow$ LD), which single relaxation enables the most performance improvement over a naive implementation of SC? Why?

7. (20 points) **Memory Consistency Example**

Dekker's algorithm is a synchronization scheme to ensure mutual exclusion between two threads. The conventional algorithm (below) requires SC semantics in order to work.

(a) Show that the algorithm (as shown below) is not enough to ensure mutual exclusion on a TSO machine. Give an example interleaving that allows both processors to enter the critical section.

(b) Assuming a TSO machine, give the minimum set of code locations where memory fences should be inserted to make the algorithm work. Explain why your fences are enough. Note: On a TSO machine, only W $\rightarrow$ R fences are needed.

```
flag[0] = false;
flag[1] = false;
turn    = 0; // or 1
```

```
P0:
flag[0] = true;
while (flag[1] == true) {
    if (turn != 0) {
        flag[0] = false;
        while (turn != 0) {
            // busy wait
        }
        flag[0] = true;
    }
}

// critical section
...

// exit critical section
turn    = 1;
flag[0] = false;
```

```
P1:
flag[1] = true;
while (flag[0] == true) {
    if (turn != 1) {
        flag[1] = false;
        while (turn != 1) {
            // busy wait
        }
        flag[1] = true;
    }
}

// critical section
...

// exit critical section
turn    = 0;
flag[1] = false;
```

8. (10 points) **Interaction of Consistency and Coherence**

Consider a directory-based machine with Weak Ordering (i.e., reordering of data reads and writes is allowed) using invalidation-based coherence over a network that does not preserve message ordering. In a correct implementation of this system, the directory serializes accesses to each line l as follows: Upon receiving a write from some node in the system, the directory sends invalidations to all sharers of l and waits for all sharers to acknowledge the invalidation before servicing any subsequent read requests for l.

Now consider a faulty implementation of the above system where the directory does not wait for all invalidations to be received before providing the new value to subsequent readers. The following code demonstrates the coherence/consistency bug. The `fence` instructions in the code are full (RW $\rightarrow$ RW) memory fences. When run on the faulty system, the resulting vector {r1, r2, r3, r4} can take on a final value that is illegal under WO. What is the illegal value and why is it illegal?

$$a = b = 0$$

(a and b are initialized to zeros in all caches before code fragments are executed.)

```
P0:
a = 1;
b = 1;
```

```
P1:
r1 = load b;
fence;
r2 = load a;
```

```
P2:
r3 = load a;
fence;
r4 = load b;
```