1. （20 分）（基于监听的缓存一致性 Snoop-based Cache Coherence (CC)）考虑一个对称共享内存多处理器系统（symmetric shared memory multiprocessor model）（3 个处理器共享一个总线），实现了一种监听式缓存一致性协议（Snoop-based CC with write invalidation）。对于下面的每个事件，解释一致性协议的步骤（缓存是否标记命中/未命中，总线上放置了什么请求，谁响应，是否需要写回等），并提及每个处理器缓存中数据块的最终状态。假设在序列开始时，X 和 Y 不在任何缓存中，缓存是直接映射(Direct-mapped Cache)的，并且块 X 和 Y 在每个缓存中映射到同一组（set）（在任何时候，缓存中不能同时存在 X 和 Y）。请完成以下表内容。

P1：写入 X
P2：写入 X
P3：读取 X
P1：读取 X
P3：写入 X
P3：读取 Y
P2：写入 Y

Request	**Cache Hit/Miss**	**Request on the bus**	**Who responds**	**State in Cache 1**	**State in Cache 2**	**State in Cache 3**
				Inv	Inv	Inv
P1: Wr X	Miss	Wr req X	Memory	M(X)	Inv	Inv
P2: Wr X	Miss	Wr req X	P1	Inv	M(X)	Inv
P3: Rd X	Miss	Rd req X	P2 responds (Memory Writeback)	Inv	S(X)	S(X)
P1: Rd X	Miss	Rd req X	Memory	S(X)	S(X)	S(X)
P3: Wr X	Permission miss (or Hit)	Upgrade X	No response. Other caches invalidate	Inv	Inv	M(X)
P3: Rd Y	Miss	Rd req Y	Memory	Inv	Inv	S(Y)
P2: Wr Y	Miss	Wr req Y	Memory	Inv	M(Y)	Inv

2. （20 分）（基于目录的缓存一致性（Directory-based Cache Coherence (CC)））**考虑与问题 1 相同的内存访问序列。**假设 4 个处理器（P1, P2, P3 和 P4）通过点对点互联网络连接，并实现带有目录的分布式共享内存(Distributed shared memory multiprocessor model)和基于目录的缓存一致性协议(Directory-based CC with write invalidation）。对于上述指令序列，基于写无效协议，互联消息传输的总数是多少?

对于每条指令，列出必须在网络上发送的消息以及缓存和目录中缓存块或者行的状态（参考课件例子）。假设一个消息可以包括一些控制信息以及地址和缓存行。还假设内存位置 X 和 Y 的主节点都与处理器 P4 关联。假设在序列开始时，X 和 Y 不在任何缓存中，缓存是直接映射的，并且块 X 和 Y 在每个缓存中映射到同一组（在任何时候，缓存中不能同时存在 X 和 Y）。

Request	**Cache Hit/Miss**	**Messages**	**Directory State**	**State in Cache 1**	**State in Cache 2**	**State in Cache 3**
			X: S: Y: S:	Inv	Inv	Inv
P1: Wr X	Miss	Wr-req to Dir in P4. Dir responds with the copy	X: M:1 Y: S:	M(X)	Inv	Inv
P2: Wr X	Miss	Wr-req to Dir in P4. Inv to P1. P1 sends data to Dir. Dir sends data to P2	X: M:2 Y: S:	Inv	M(X)	Inv
P3: Rd X	Miss	Rd-req to Dir. Dir forwards req to P2. P2 sends to data to Dir. Memory writeback. Dir sends data to P3	X: S:2,3 Y: S	Inv	S(X)	S(X)
P1: Rd X	Miss	Rd-req to Dir. Dir responds P1 with data	X:S:1,2,3 Y: S	S(X)	S(X)	S(X)
P3: Wr X	Permission Miss	Upgrade-req to Dir. Dir sends Inv to P1 and P2, and grant to P3	X:M:3	Inv	Inv	M(X)
P3: Rd Y	Miss	Rd-req sends to Dir. Dir responds P3 with the copy	X:S: Y:S:3	Inv	Inv	S(Y)
P2: Wr Y	Miss	Wr-req sends to Dir. Dir sends Inv to P3, and forwards data to P2.	X:S: Y:M:2	Inv	M(Y)	Inv

3. （10 分）（Memory System）一台新的超级计算机配备了基于 DRAM 的内存系统，具体配置如下：

总容量为 1EB（Exabyte），DRAM 行（Row）大小为 8 千字节（KB）。系统中所有 DRAM 行的最小保留时间(The minimum retention time)为 64 毫秒。为确保数据不丢失，每行 DRAM 每 64 毫秒刷新一次(refresh)。

（注意：对于本问题中的每次计算，你可以将答案以 2 的幂的简化形式给出。）

（a）内存系统中有多少 DRAM 行（Row)?

（b）在 64 毫秒内发生了多少次 DRAM 刷新（DRAM refreshes）？

Capacity = 2^60 and row size = 2^13, so there are 2^47 rows.

Because each row needs to be refreshed every 64 ms, all rows need to be refreshed within 64ms.

Thus, the number of refreshes = number of rows = 2^47.

4. （20 分）（Memory System）一台服务器拥有 4GB 的 DRAM 主内存系统。每行 DRAM 每 64 毫秒刷新一次（提示：可以从 Rowbuffer 的功能和功耗方面考虑）。

（a）设计者该服务器上独立运行两个应用程序 A 和 B。尽管应用程序 A 和 B 的内存请求数量相似，但应用程序 A 在内存延迟上花费的周期比应用程序 B 多得多。这可能是什么原因?

Application A experiences a high rate of row-buffer conflicts for its memory requests, leading to longer service times and increased stalling for memory. In contrast, Application B enjoys a high rate of row-buffer hits, which results in more efficient memory request servicing.

（b）应用程序 A 消耗的内存能耗（Energy）也比应用程序 B 多得多。这可能是什么原因?

A row-buffer conflict consumes more energy than a row-buffer hit. A row-buffer conflict requires a precharge, an activate and a read/write, whereas a row-buffer hit only requires a read/write. Hence, application A consumes more memory energy.

（c）当应用程序 A 和 B 一起在机器上运行时，应用程序 A 的性能显著下降，而应用程序 B 的性能下降得不多。这可能是什么原因?

When applications are executed concurrently, they can interfere with one another, leading to a degradation in performance for both. Hence, both applications' performance degrades when they run together. However, if a memory scheduler that favors row-buffer hits over row-buffer conflicts (like FR-FCFS) is used, it would favor application B's requests over application A's requests. As a result, Application A experiences a more significant performance degradation due to the scheduler's bias towards the memory access pattern of Application B.

（d）设计者决定采用更智能的策略来刷新内存。只有在过去 64 毫秒内未被访问的行（Row）才会被刷新。你认为这是一个好的设计吗？为什么？

This can significantly cut down on refresh energy consumption if a large proportion of the memory rows are populated with data that is accessed within the 64 ms refresh interval, as these rows can avoid explicit refresh cycles. However, if only a small number of rows contain data and are accessed, the policy offers minimal savings in refresh energy, since the majority of rows continue to undergo refresh operations at the 64 ms interval.

（e）当应用这个新的刷新策略时，在运行应用程序 B 时刷新能耗显著下降。相比之下，在运行应用程序 A 时，刷新能耗只略有减少。这可能吗？为什么？

This is possible. If application B has a large working set, it could access a large fraction of the memory rows (within the 64 ms refresh window) and hence these rows do not have to be refreshed explicitly. On the other hand, application A could have a much smaller working set and hence a large fraction of rows still have to be refreshed at the 64 ms frequency.

5. （10 分）大型缓存（Large Cache）假设有一个大型共享的 LLC（最后一级缓存），它是 tiled 分布的，并分布在处理器芯片上。假设操作系统的页面大小为 16KB。整个 LLC 的大小为 32MB，使用 64 字节的缓存块，并且是 16 路组相联。那么最大的 tile 数量是多少，以便操作系统可以完全灵活地在其选择的 tile 中放置页面？

Number of cache blocks = 2^25 / 2^6= 2^19

Number of sets = 2^19 / 2^4 = 2^15, thus we have 15 bits for set indexing, and 6 bits for block offset, total 21 bits, while 14 bits for page size.

We can use all 21 – 14 = 7 bits for tile mapping, so we have 2^7 =128 tiles at most.

6. （20 分）(Memory scheduling）以下两种不同的内存调度策略：

FR-FCFS：最先就绪，先来先服务（First-Ready, First-Come-First-Served），

STFM：延迟时间公平内存调度（Stall-Time Fair Memory scheduling），

(a) FR-FCFS 策略是一种常用的内存调度策略，旨在最大化 DRAM 吞吐量。请解释为什么 FR-FCFS 与 FCFS (先来先服务）策略相比提高 DRAM 吞吐量。请通过一个或多个访存例子解释。

FR-FCFS improves DRAM throughput by maximizing the row-hit ratio (i.e., minimizing the row-miss penalty). In the following memory-access pattern, FR-FCFS first serves memory requests to Row 0 without re-activating the row. On the other hand, FCFS repeats activating the two rows alternately, which significantly increases the latency of each request.

Row access pattern: Row 0 - Row 1 - Row 0 - Row 1 - ...

(b) 尽管 FR-FCFS 策略提供了比 FCFS 策略更高的 DRAM 吞吐量，但使用 FR-FCFS 的 DRAM 内存系统容易受到 Denial-of-Service（DoS）攻击的影响，这种攻击在多个线程共享 DRAM 内存系统时会显著降低某个线程的性能。考虑两个内存密集型线程 A 和 B，它们表现出以下内存访问模式：

A 表现出流式访问模式（Stream），顺序地访问同一行（Row）中的数据。

B 表现出随机访问模式（random），随机访问不同行中的数据。

A 和 B 的内存密集度相似，即在给定的时间框架内，它们生成的内存请求数量大致相同。在 FR-FCFS 策略下，哪个线程（A 或 B）可以被对手用来对另一个线程执行 DoS 攻击？请解释你的答案。

When A and B share a DRAM memory system that adopts the FR-FCFS policy, the memory controller (MC) prefers to serve memory requests from A, as they likely hit the row buffer.