Computer System Architecture
6.5900 Quiz #2
November 17th, 2023

Name: _____

This is a closed book, closed notes exam.
80 Minutes
17 Pages (+2 Scratch)

Notes:

- Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully.
- Please carefully state any assumptions you make.
- Show your work to receive full credit.
- Please write your name on every page in the quiz.
- You must not discuss a quiz's contents with other students who have not yet taken the quiz.
- Pages 18 and 19 are scratch pages. Use them if you need more space to answer one of the questions, or for rough work.

Part A	_____	16 Points
Part B	_____	30 Points
Part C	_____	24 Points
Part D	_____	30 Points
TOTAL	**_____**	**100 Points**

Name _____

Part A: Multithreading (16 points)

Consider the following code which repeatedly operates on array A. Array A consists of 256 4B integers (so the total size is 1KB). Assume N is very large.

```
int A[256];
int sum = 0;

for (int i = 0; i < N; i++) {
    for (int j = 0; j < 256; j++) {
        sum += A[j] + 8;
    }
}
```

The following is the equivalent RISC-V assembly *just for the inner loop*:

```
# Assume the following:
# x1 holds sum
# x2 holds base address of A
# x3 holds j
# x10 holds 1024 (the loop bound)

_iloop:      ADD    x5, x2, x3       # Calculate address A[j]
             LW     x4, 0(x5)        # A[j]
             ADDI   x4, x4, 8        # A[j] + 8
             ADD    x1, x1, x4       # sum += A[j] + 8
             ADDI   x3, x3, 4
             BNE    x3, x10, _iloop
_iloop_done:
```

Ben runs this code on an in-order pipelined processor with the following characteristics:

- The processor has a 2-way set-associative 1KB data cache with 16B blocks.
- Memory operations take 5 cycles if they hit in the cache, and 100 cycles if they miss. This means that if a load is issued on cycle N and it hits in the cache, the value is available for the dependent instruction on cycle N + 5.
- Integer operations take 1 cycle, i.e., their results are immediately usable by the following instruction via bypassing.
- Assume that instruction fetches cause no stalls and all branches are predicted perfectly.

Name _____

Question 1 (4 points)

In steady state, how many cycles does the processor stall waiting for loads per iteration of the inner loop?

In steady state, the whole array is cached, so each load takes 5 cycles. This means there are 4 cycles worth of load stalls each iteration since there's one load.

Question 2 (6 points)

Ben Bitdiddle modifies his processor to support fine-grain round-robin multithreading. The processor now supports 2 threads, and threads are switched every cycle using a fixed round-robin schedule. If a thread cannot be scheduled because it is stalled on an instruction, the processor inserts a pipeline bubble and switches to schedule the other thread for the next cycle.

Each thread now works on its own 1KB-sized array, calculating the sum independently (i.e., there is no data sharing between threads).

Alyssa P. Hacker points out that this multithreaded implementation will reduce the number of summations done per cycle compared to the single-threaded version. Is she correct? Briefly explain why or why not.

Yes, she is correct. Since it's impossible to fit both arrays now, and they are switched round-robin, only half of each array will fit in the cache. Since load misses take 100 cycles to resolve, this will severely degrade the summation throughput.

Name _____

Question 3 (6 points)

Devise a simple modification to the C code to make the multithreaded implementation faster. Write the modified C code below, and briefly explain why it improves performance.

Simply divide the A array into two equal haves, and work on each half to completion before moving on to the next one.

```
for (int i = 0; i < N; i++) {
    for (int j = 0; j < 128; j++) {
        sum += A[j] + 8;
    }
}

for (int i = 0; i < N; i++) {
    for (int j =128; j < 256; j++) {
        sum += A[j] + 8;
    }
}
```

Name _____

Part B: Cache Coherence (30 points)

Ben wants to design a directory-based MSI coherence protocol where the directory uses a bit vector to store each sharer set. Given an N-core system, the directory keeps a N-bit wide bit vector for each line it tracks. If the i-th bit of the bit vector is set, it means that core i's cache holds a copy of the line in either S or M state.

Question 1 (4 points)

Suppose that the processor has N cores, each of which has a 1KB private cache with 32B lines.

a) How many *entries* does the directory need to have to keep track of all cache lines in the private caches?

32 private lines * N cores = 32N

b) In this directory, what is the total size of all the sharer sets in bits?

32N entries * N bits/entry = 32N^2

Question 2 (5 points)

Ben changes the design of the sharer sets by replacing the bit vector with a set of **sharer pointers**. Each pointer now keeps track of one sharer by storing the sharer's core id. For example, if cores 10 and 12 hold the line in S state, the sharer set for this line in the directory will consist of two pointers, "10" and "12". **Each sharer set can store at most 16 sharer pointers.**

Given an N-core system where N is a power of 2, what is the total number of bits needed for all sharer pointers in a sharer set?

16 pointers * logN bits / pointer = 16logN

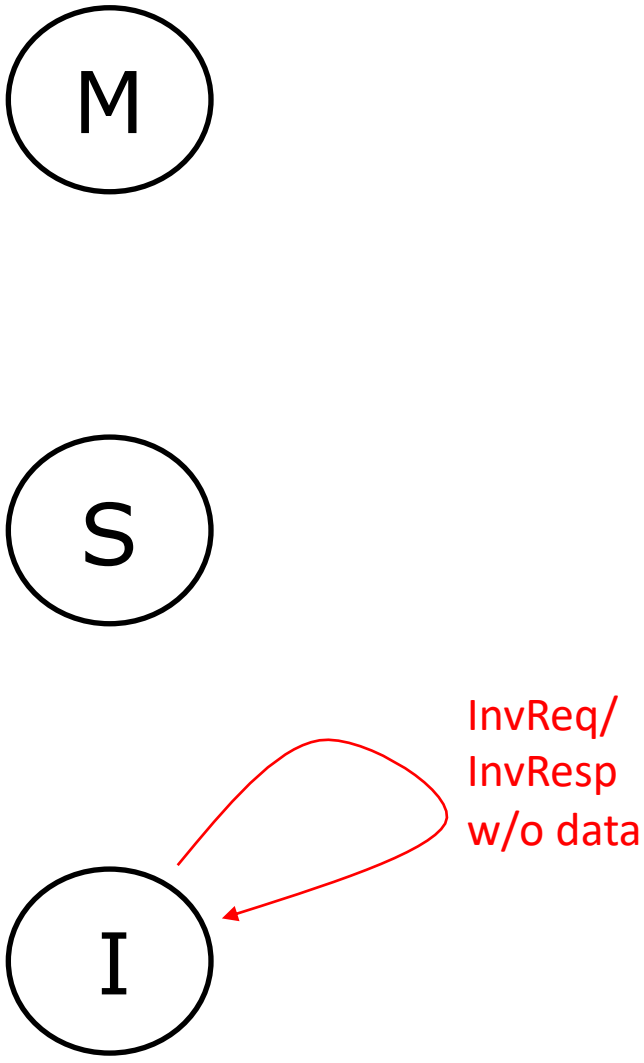
Name _____

Question 3 (8 points)

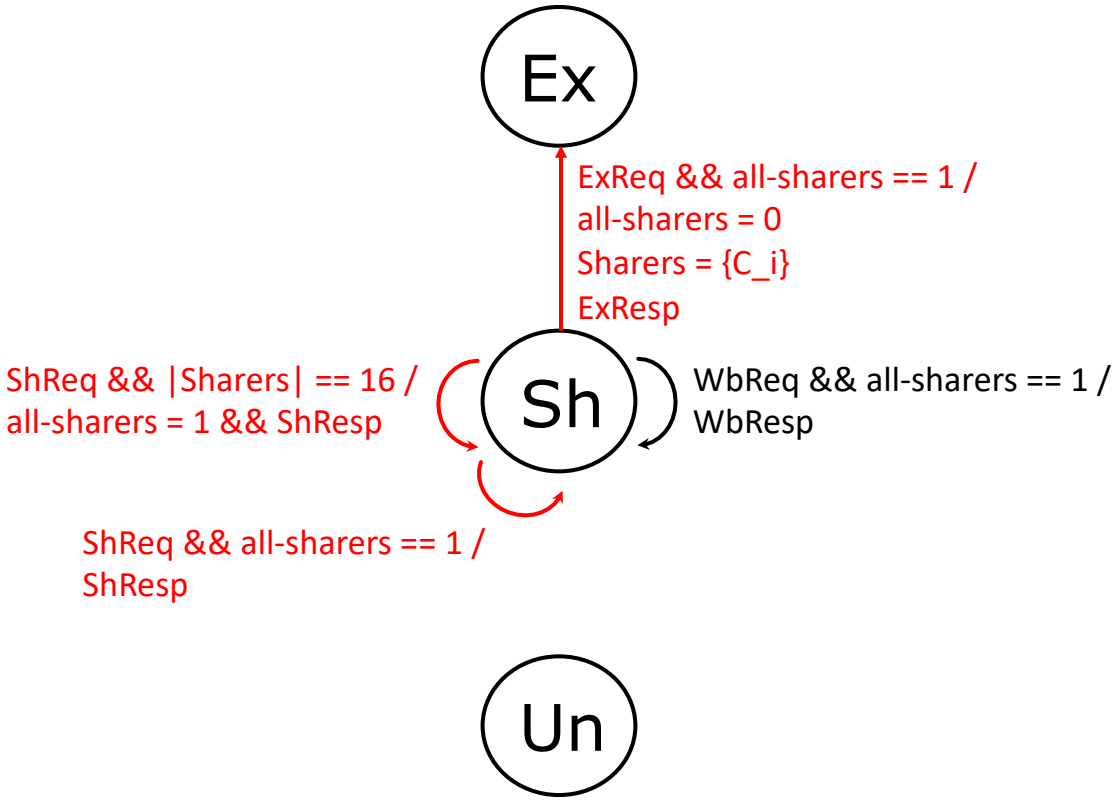
To design the coherence protocol with sharer pointers, Ben starts with the basic MSI protocol described in the quiz handout. Because the number of sharer pointers is less than the number of potential sharers (assume N > 16 cores), Ben introduces an **all-sharers bit** to each line in the directory. When this bit is set, the directory conservatively considers all private caches as sharers of this line.

For the following questions, consider only transitions between stable states.

a) What additional transitions does Ben need for each of the 3 processor states (M, S, and I)? Show the transition(s) below with an arrow between the source and destination states, clearly denoting it with the triggering action and the action taken during the transition. Do not introduce new messages.



b) What additional transitions does Ben need for each of the 3 directory states (Ex, Sh, and Un)? Show the transition(s) below with an arrow between the source and destination states, clearly denoting it with the triggering action and the action taken during the transition. Do not introduce new messages. We've added an additional transition due to a WbReq to get you started.

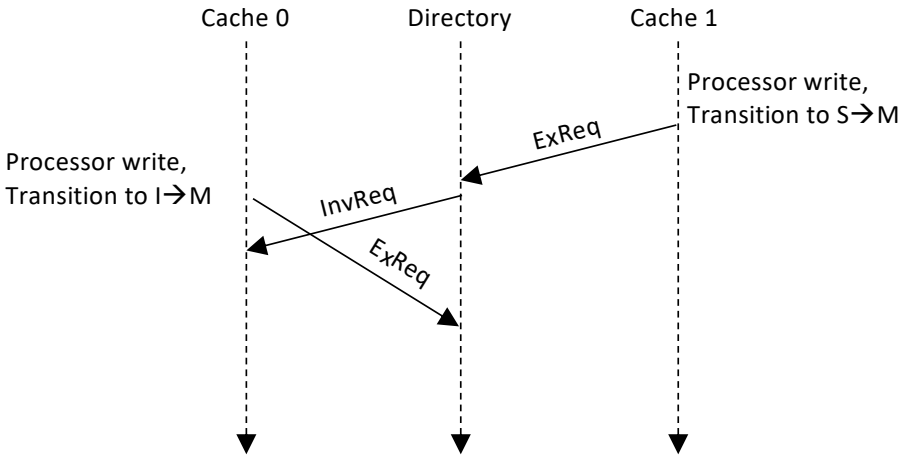


Name _____

Question 4 (6 points)

So far, we assumed that each coherence transaction completes before the next transaction begins. Alyssa P. Hacker points out that Ben's modification to the coherence protocol introduces additional races that he must now deal with when transient states are considered.

To see this, consider the following 2-core scenario where a line is in shared state, with the all-sharers bit set in the directory, but only Core 1 has an actual copy of the line in its cache. Core 1 attempts to write to the line, triggering the directory to send an InvReq to Core 0's cache. However, before the InvReq arrives at Core 0's cache, Core 0 writes to the **same cache line**, sending an ExReq to the directory. Specifically, Core 0 sends an ExReq before it receives an InvReq.



To maintain coherence, what action should Cache 0 take in response to the InvReq while in the I->M transient state? Pick one of the following three answers:

A: Acknowledge the InvReq by sending an InvResp and remain in the I->M transient state to wait for a later ExResp. Meanwhile, the directory will buffer and serve ExReqs in the order of receiving these requests.

B: Buffer or NACK the InvReq, wait for an ExResp from the directory, and then proceed to performing the invalidation. Meanwhile, the directory will buffer and serve ExReqs in the reverse order of receiving these requests i.e., it will first respond to Cache 0's request with an ExResp, then respond to Cache 1 with an ExResp when Cache 0's InvResp arrives.

C: Performing either of A or B will result in correct behavior.

Name _____

Question 5 (7 points)

Alyssa P. Hacker proposes an alternative solution that works as follows. Instead of using an all-sharers bit when the number of sharers exceeds the size of the sharer set, the directory evicts an existing sharer pointer and sends an InvReq to the corresponding cache. After the InvResp comes back, the directory replaces the pointer with the new sharer.

Describe two scenarios, one where you would prefer using the all-sharers bit, and another where you would prefer evicting a sharer pointer entry. You can either describe them in words or write code snippets that describe these scenarios.

Lots of possible answers here. Key point is the number of extra invalidations you have to do. The following are just some examples.

All-sharer preferred: Data frequently read repeatedly between all cores

Eviction preferred: Producer-consumer relation where one core writes to a value that a subset of cores (16 < # cores sharing << N) consume. Keeping a restricted set of sharers will avoid sending N InvReqs.

Name _____

Part C: Memory Consistency (24 points)

The following questions deal with memory accesses from multiple cores in a cache-coherent shared memory machine. For each question, you will consider the possible outcomes for the following memory consistency models:

- Sequential Consistency (**SC**)
- Total Store Order, IBM370-style (**TSO-IBM370**): Stores can be reordered after later loads, but *store-to-load forwarding is disallowed until the value is globally visible to other cores*.
- Total Store Order, x86-style (**TSO-x86**): Stores can be reordered after later loads, and stores from the same core are visible in the same order. *Store-to-load forwarding within the same core is allowed*.
- Relaxed Memory Order (**RMO**): Loads and stores can be reordered after later loads and stores, and store-to-load forwarding is allowed.

Assume that all registers (r1, r2, ...) and memory locations (a, b, ...) initially contain 0.

Question 1 (8 points)

Consider a cache-coherent shared-memory machine that executes the following two threads on two different cores. Assume that memory locations `a`, `b`, and `c` contain initial value `0`.

T1	T2
ST (a) ← 1 LD r1 ← (a) LD r2 ← (b)	ST (b) ← 1 ST (a) ← 2

Circle the consistency models for which the final values r1 = 1, r2 = 0, and (a) = 1 are possible.

SC **TSO-IBM370** **TSO-x86** **Relaxed**

Name _____

Question 2 (8 points)

T1	T2	T3	T4
ST (a) ← 1	ST (a) ← 2	LD r1 ← (a) LD r2 ← (a)	LD r3 ← (a) LD r4 ← (a)

Circle the consistency models for which the final values r1 = 1, r2 = 2, r3 = 2, and r4 = 1 are possible

SC **TSO-IBM370** **TSO-x86** **Relaxed**

Question 3 (8 points)

T1	T2	T3
ST (a) ← 1 LD r1 ← (a) LD r2 ← (b)	ST (b) ← 1	LD r3 ← (b) LD r4 ← (a)

Circle the consistency models for which the final values r1 = 1, r2 = 0, r3 = 1, r4 = 0 are possible.

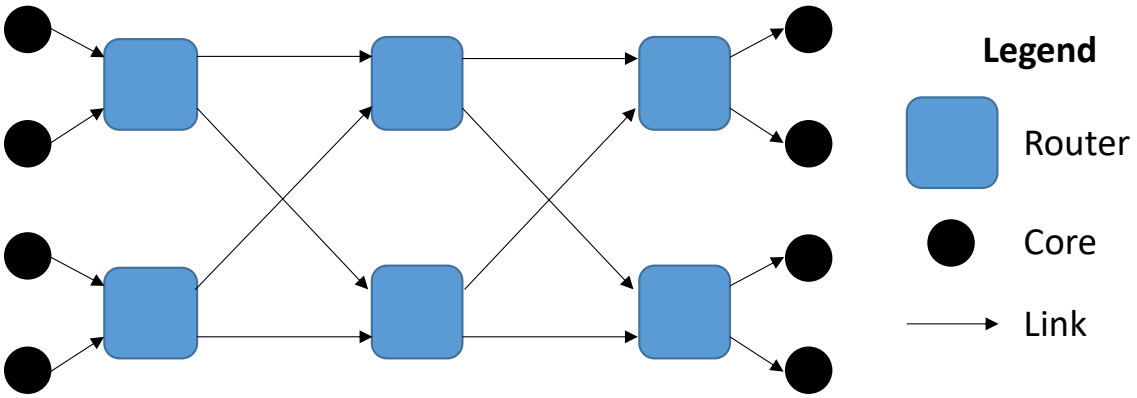
SC **TSO-IBM370** **TSO-x86** **Relaxed**

Name _____

Part D: On-Chip Networks (30 points)

Question 1 (6 points)

Consider the following **Benes network topology** for 4 processor cores. The set of source cores are shown on the left column and the destination cores are on the right column. The routers all have two input and output ports. All the links connecting the cores and routers are **unidirectional**.



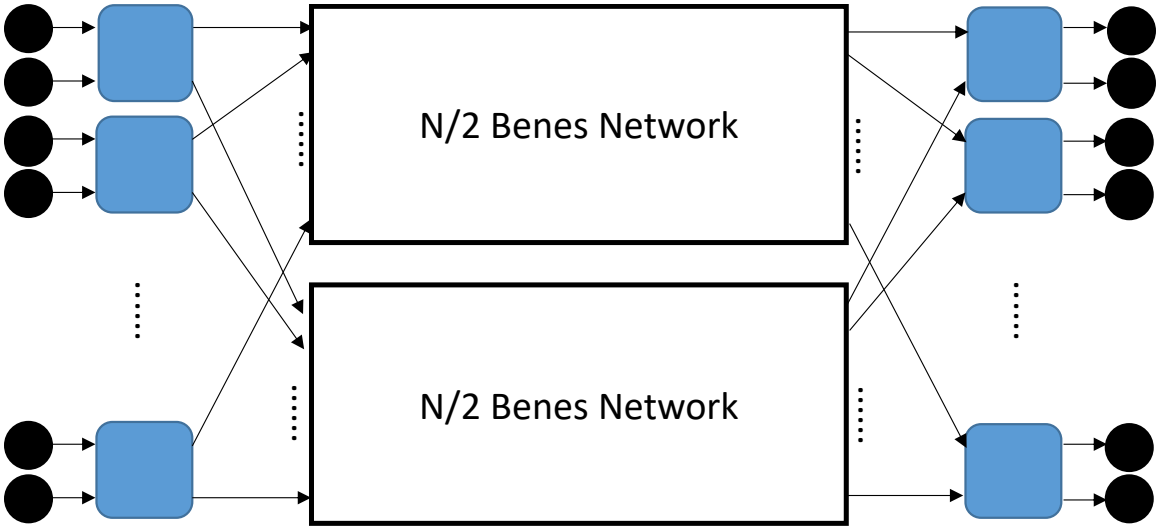
Fill in the following table of topology metrics for this network.

	Benes Network
Diameter	4
Average distance	4
Bisection bandwidth	4

Name _____

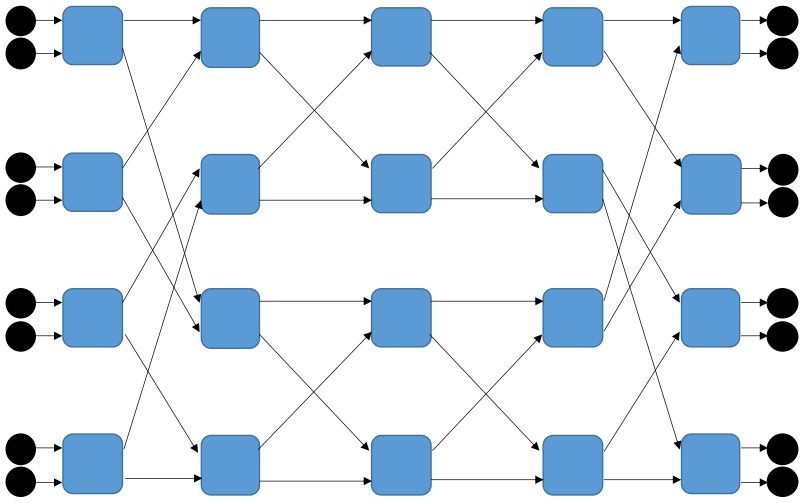
Question 2 (9 points)

A Benes network for N cores can be defined recursively as a combination of two smaller N/2 Benes networks, as shown below:



Notice that, to combine the two N/2 Benes networks, we add N/2 routers on the input side and another N/2 routers on the output side of the network. For the two outputs of each router on the input side, one is connected to the input of the top Benes network and the other is connected to the bottom one. The connections are such that the top router is connected to the topmost input of each network, the second from top to the second input of each network, and so forth.

For example, the following is a Benes network for N=8 cores:



Name _____

Fill in the following metrics for a Benes network for N cores.
Assume that N is a power of 2.

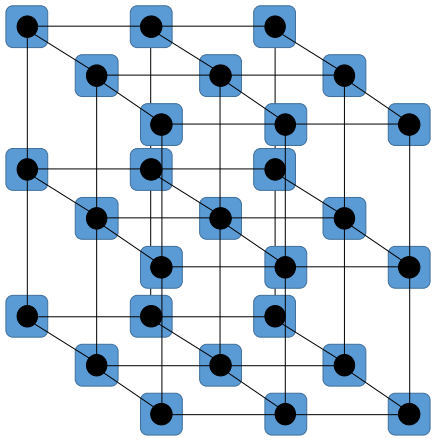
	Benes Network
Diameter	*2logN*
Average distance	*2logN*
Bisection bandwidth	*N*

Name _____

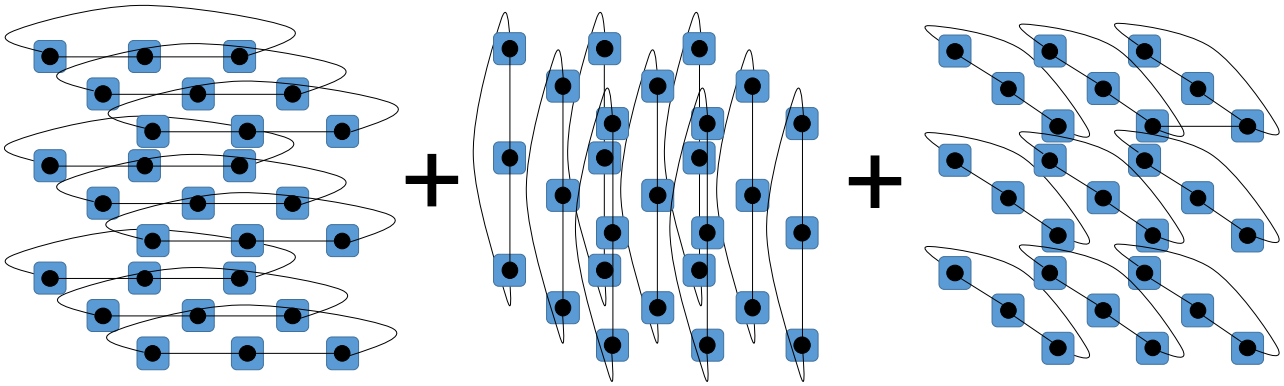
Question 3 (8 points)

Consider 3-dimensional mesh and torus networks with N nodes.

3-D Mesh Topology



3-D Torus Topology



Note that we pictorially describe the 3-D torus topology as a combination of 3 discrete sets of links, which each show the connections in a single dimension.

Name _____

Assume that N = k^3 where k is an even integer. Fill in the table below as a function of the number of nodes in the network. The units for each cell are hops or links. For average distance, assume uniform random traffic (where each node sends $1/N^{th}$ of the traffic to each destination, including itself). You can use N and k in your formulas.

You only need to provide the asymptotic growth for the number of links and average distance for the 3D mesh. For everything else, you need to provide the exact number to receive full credit, and asymptotic growth will get you partial credit.

	3D Mesh	**3D Torus**
Number of links	*O(N)*	*3N*
Diameter	*3(k - 1)*	*3k/2*
Average distance	*O(k)*	*3k/4*
Bisection bandwidth	*k^2*	*2k^2*

Name _____

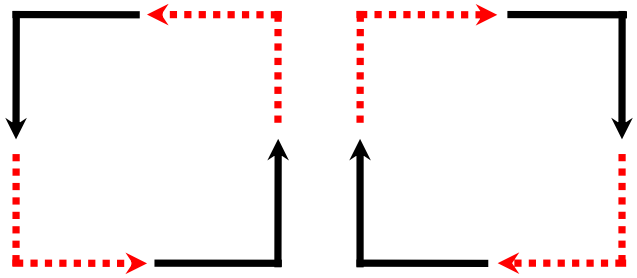
Question 4 (7 points)

Consider the XY dimension-order routing described in class. If a packet wants to go from coordinate (a,b) to (a', b'):

- The packet is first minimally routed in the X dimension until its X-coordinate equals a'.
- Next, the packet is minimally routed in the Y dimension until its Y-coordinate equals b'.

Is the routing algorithm deadlock free for a 2D-torus topology? Explain your reasoning with the turn model, clearly stating which turns are allowed and which are forbidden.

Just like in meshes, 2 turns are disallowed for both clockwise and counter-clockwise directions, so this does not result in a deadlock:



XY Model

Name _____

Scratch Space

Use these extra pages if you run out of space or for your own personal notes. We will not grade this unless you tell us explicitly in the earlier pages.

Name _____

Scratch Space