ECE 757
HW2
Spring 2009

Problem 1)
Real coherence controllers include numerous transient states in additions to the ones shown in Figure 0 to support split-transaction busses. For example, when a processor issues a bus read for an invalid line (I), the line is placed in an IS transient state until the processor has received a valid data response that then causes the line to transition into the shared state (S). Given a split-transaction bus that separates each bus command (bus read, bus write, and bus upgrade) into a request and response, augment the state table and state transistion diagram of the figure to incorporate all necessary transient states and bus responses. For simplicity, assume any bus command for a line in the transient state gets a NAK response that forces it to be retried after some delay.

Current State s	Event and Local Coherence Controller Responses and Actions (s' refers to next state)					
	Local Read (LR)	Local Write (LW)	Local Eviction (EV)	Bus Read (BR)	Bus Write (BW)	Bus Upgrade (BU)
Invalid (I)	Issue bus read if no sharers then s' = E else s' = S	Issue bus write s' = M	s' = I	Do nothing	Do nothing	Do nothing
Shared (S)	Do nothing	Issue bus upgrade s' = M	s' = I	Respond shared	s' = I	s' = I
Exclusive (E)	Do nothing	s' = M	s' = I	Respond shared s' = S	s' = I	Error
Modified (M)	Do nothing	Do nothing	Write data back; s' = I	Respond dirty; Write data back; s' = S	Respond dirty; Write data back; s' = I	Error

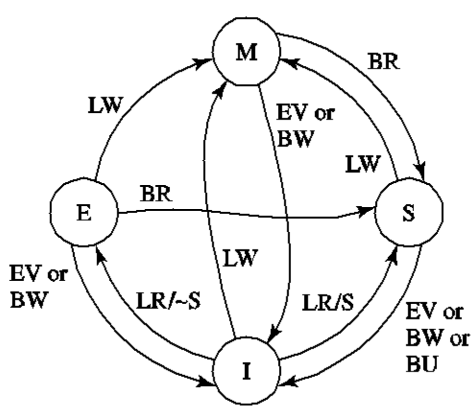

Figure 0

Problem 2)
Part A)
A Test&Test&Set lock has been implemented with load-linked store-conditional as seen below:

```
TTS   ld    lock              ; first Test, see if it is 'unlocked'
      cmp   'locked'
      beq   TTS
      ll    lock              ; load-linked
      cmp   'locked'
      beq   TTS               ; second Test, we have seen 'unlocked' in
first Test
      stc   'locked'    lock  ; set the lock to 'locked'
```

And to release the lock we simply use:

```
      st    'unlocked'  lock  ; unlock the lock
```

Three processors (P0-P2) acquire and release the lock in the following order:

A	**P0**	**First request of lock**
B	**P0**	**Acquires lock**
C	**P1**	**First request of lock**
D	**P2**	**First request of lock**
E	**P0**	**Release the lock**
F	**P1**	**Acquires lock**
G	**P1**	**Releases lock**
H	**P2**	**Acquires lock**
I	**P2**	**Releases lock**

Assume that each step is discrete (only one processor performs a bus operation at a time), and that the load-locked instruction behaves as a local read and store-conditional and store both are a local write. Fill in the table below for the Bus Events, the Requesting Processor, as well as the MESI (from Figure 0 above) state that each processor is in during any step.

Step	Bus Event	Requesting Processor	P0 State	P1 State	P2 State

Part B)
Consider the MCS lock from Mellor-Crummey and Scott's paper "Algorithms for Scalable Synchronization on Shared Memory Multiprocessors". Qualitatively describe the differences we would see from the table you have created above for the Test&Test&Set lock.

Problem 3)
The DRAM in a system is configured such that 4KB of physical address space are addressable per column access. Additionally, the DRAM chips have 2 banks and are arranged such that even and odd 4KB pages lie on separate banks. The DRAM chips have a row access time of 3 cycles, precharge time of 3 cycles, and column access time of 1 cycle.

Assume an application works with 2KB data elements which are sequentially stored in physical address space as in Figure 1. Although the data is sequentially stored in this fashion, it is travesed like a binary tree as shown in Figure 2.

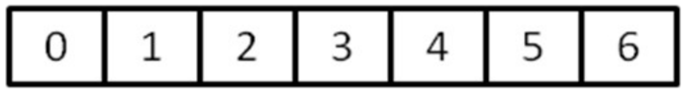

Figure 1. Data Arrangement.

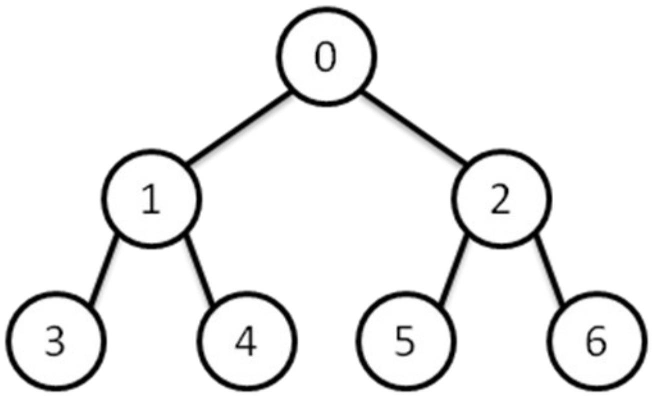

Figure 2. Data Structure.

For this problem, assume the application performs a depth-first traversal (0,1,3,4,2,5,6) of the tree accessing a single byte per element. Create two timing graphs showing the minimum read latency for this traversal assuming open and closed page memory scheduling.

Problem 4

You are the interconnection network designer for a new 16-core multicore processor. In this design you are to optimize the network such that it can transmit 1024 bit packets with the smallest possible latency. Typically this network operates under light loads, so the zero load latency (average latency assuming this packet is the only one in the network) should be indicative of performance. The networks you are considering are a fully connected crossbar, a bidirectional ring, and a two dimensional mesh. Due to the limited perimeter and other wiring constraints, each processor tile can have at most 256 wires connected. Assume the wires are capable of operating at 2 GHz and the processing time of each network hop is 6ns.

In the ring and mesh networks, the individual links are short leading to only a single cycle delay on the link. For the crossbar, assume a tiled layout for the processor cores and that the crossbar data links are pipelined that the signals travel one tile per cycle.

For each potential network calculate the average zero-load latency for the transmission of a 1024 bit packet between any two nodes within the network.

Problem 5

Assume the following program segments are executed on two processors of a multi-processor machine. In the pseudo code, capital letters are variables in memory and terms r <number> are physical registers. Initially before execution, all memory variables are equal to 0.

P0	P1
W = 1	X = 2
Y = 3	Y = 4
r1 = X	r3 = Y
r2 = Y	r4 = W

After executing these code segments on three different systems the following valued sequences are observed in the physical registers.

	r1	r2	r3	r4
System 1	2	0	4	1
System 2	2	3	3	1
System 3	0	4	4	0

For each system, draw the corresponding constraint graphs given by these executions and state if the systems consistency model satisfies 1) Sequential consistency 2) TSO consistency or 3) None.