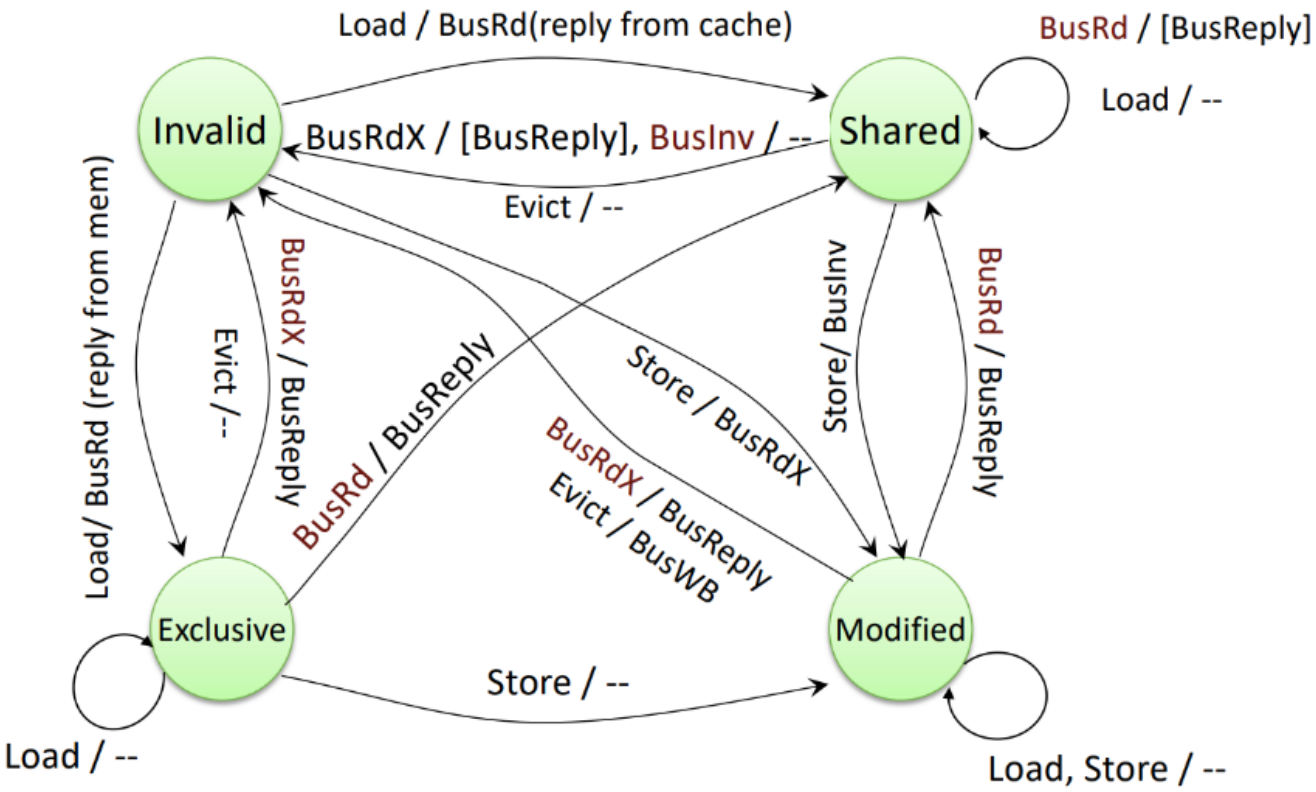
EECS 470 F24 Final Review Session Worksheet

Multiprocessors:

1. Complete the following state diagram for the MESI protocol:



2. Consider 3 processors using a snooping bus-based MESI protocol. All three processors have 2 line direct mapped caches with each line consisting of 16 bytes. The caches begin with all lines marked invalid. Complete the following table (don't worry about HIT/HITM):

Processor	Address	Read/Write	Cache Hit/Miss	Bus Transaction(s)	"4C" miss type (if any)
1	0x100	Write	Miss	BusRdX	Compulsory
1	0x200	Read	Miss	BusRd, BusWr	Compulsory
2	0x200	Read	Miss	BusRd, BusReply	Compulsory
1	0x200	Write	Miss	BusInv	Coherence
1	0x110	Write	Miss	BusRdX	Compulsory
1	0x110	Read	Hit	-	-
3	0x110	Write	Miss	BusRdX, BusReply	Compulsory
2	0x100	Read	Miss	BusRd	Compulsory
3	0x100	Read	Miss	BusRd, BusReply	Compulsory
1	0x100	Write	Miss	BusRdX	Capacity

Proc 1:

	Address	State
Set 0	0x100	M
Set 1		

Proc 2:

	Address	State
Set 0		
Set 1		

Proc 3:

	Address	State
Set 0		
Set 1	0x110	M

Virtual Memory and Caches:

1. Consider a 42-bit, byte-addressable system that uses virtual memory. The system has a maximum of 128 GB of physical memory installed with a page size of 8 KB. If we have a hierarchical page table such that each page table level must fit in a single page, how many page table levels would we need to map all of the system's virtual memory? Assume page table entries are still 8B each.

   We are given that each page table entry is 8B. Since each page table must fit within a single page and pages are 8 KB in size, there are 8 KB / 8 B per entry = 2^10 entries in each page table. Each hierarchy of the page table can be indexed with 10 bits of the address.

   With an 8 KB page, there are log2(8 KB) = 13 page offset bits. Therefore, the virtual page number (VPN) is 42 – 13 = 29 bits long.

   Since each page table level can be indexed with 10 bits of the address, there needs to be ceiling(29 bits / 10 bits per level) = **3 levels**

2. A useful formula for determining the maximum allowable cache size that avoids the synonym problem for a VIPT cache given associativity and page size is:

   page size * associativity >= cache size

   Derive the formula above from the more elementary formula that:

   page offset bits >= block offset bits + set index bits

   Doing some rearranging and algebra:

   log2(page size) >= log2(block size) + log2(number of sets)
   log2(page size) >= log2(block size) + log2(cache size / (associativity * block size))
   log2(page size) >= log2(block size * cache size / (associativity * block size))
   log2(page size) >= log2(cache size / associativity)
   Page size >= cache size / associativity
   Page size * associativity >= cache size

3. Chips'N'Dip Processors are trying to design a set-associative virtually-indexed physically-tagged cache hierarchy. They have designed a TLB with a 3ns access latency, a cache data array with a 6ns latency, a cache tag array with a 4ns latency, tag comparison logic with a 2ns latency, and output multiplexors that can select the steer the appropriate way and block offset to the cache output port in 1ns. What is the best clock frequency they could hope to achieve (assuming the cache access logic is the processor's critical path; neglect any logic that isn't described above.)?

   In a VIPT cache, there are two parallel paths. The first path is the data array lookup using the virtual index and routing to the output port. The second path is the TLB address translation followed by tag comparison of the tag from the tag array and the physical address tag. Note that TLB address translation can also occur in parallel with tag array look up. The longest latency path is data array lookup (6 ns) following by output steering (1 ns) which results in a clock period of 7 ns, or **143 MHz.**