CSE 560 – Practice Problem Set 9

1. Suppose there are 10 processors on a bus that each try to lock a variable simultaneously. Assume that the primitive instructions available are a load-lock (`ll`) and a store-conditional (`sc`). Further, assume that each bus transaction (read miss or write miss) is 100 clock cycles long. You can ignore the time of the actual read or write of a lock held in the cache, as well as the time the lock is held (they won't matter much!).

   Determine the number of bus transactions required for all 10 processors to each acquire and release the lock, assuming they are all spinning when the lock is released at time 0. About how long will it take to process the 10 requests? Assume that the bus is totally fair so that every pending request is serviced before a new request and that the processors are equally fast.

2. Assume that variables `x1` and `x2` are in the same cache block, which starts in the shared (S) state in the caches of processor 1 (P1) and processor 2 (P2). I.e., at some point in the past, both variables were read by both processors. The system uses a traditional MSI (3-state) cache coherence protocol.

   Assuming the following sequence of events, identify each access as a true sharing miss, a false sharing miss, or a hit. Any miss that would occur if the block size were one word (the size of `x1` or `x2`) is designated a true sharing miss.

Time	**P1**	**P2**
1	Write `x1`	
2		Read `x2`
3	Write `x1`	
4		Write `x2`
5	Read `x2`	

3. For this problem, assume that we are dealing with a bus-based shared memory multiprocessor using the MESI protocol.

   Each processor core in the system is identical, and the following information applies to each processor core in the system individually. Instructions are 32 bits wide. 50% of the instructions executed are loads or stores. Of these loads and stores, on average 70% are reads to private data, 20% are writes to private data, 8% are reads to shared data, and 2% are writes to shared data.

   Each processor has a single-level split instruction/data cache. The instruction cache is 16 KB, two-way associative, and has 16 byte lines. The data cache is 16 KB, direct mapped, and also has 16 byte lines. The hit rates in the caches are as follows: 97% for private data, 95% for shared data, and 98.5% for instructions. Cache hit time is one cycle.

The system bus has 64 data lines and 32 address lines. The bus is atomic. The bus is clocked at one-half the speed for the processor. For reads, memory responds with data 12 bus cycles after being presented the address, and supplies one block of data per bus cycle after that. For writes, both address and data are presented to memory at the same time. Thus a single-word write consumes 1 bus cycle, while a 16-byte write consumes two cycles. Assume all requests are satisfied by the memory system, not by other caches.

The processor CPI is 2.0 before considering memory penalties.

(a) We want to place as many processors as possible on the bus. What is the bus utilization of a single processor if the caches are write-through with write-allocate strategy? How many of these processors can the bus support before it saturates?

For this part of the problem, assume that a write to the bus automatically invalidates any other existing copies of the data being written. Ignore bus contention and coherence messages received from other processors (e.g., remote snoops), but do consider coherence traffic generated by the processor.

The key to approaching this problem is to recognize that the true CPI (including memory penalties) includes three components: the 2.0 base CPI (CPI_{base}), plus a component due to bus stalls on instruction cache misses (CPI_{busI}), plus a component due to bus stalls on data cache misses (CPI_{busD}). Once we have computed these bus CPI components, we can compute the bus utilization as the average fraction of the per-instruction time that the bus is busy, or

$$Bus\ Utilization = \frac{CPI_{busI}+CPI_{busD}}{CPI_{base}+CPI_{busI}+CPI_{busD}}.$$

Perform all of your calculations in processor cycles.

(b) How many processors can the bus support without saturating if the caches are write-back (and write-allocate)? Assume that the probability of having to replace a dirty block in the cache on a miss that fetches a new block is 0.3. Also assume a MESI protocol, and again ignore bus contention and coherence messages received from other processors (e.g., remote snoops), but do consider coherence traffic generated by the processor. Assume that the cache protocol supports upgrades, so a write hit to a shared block causes an invalidate transaction only, taking one bus cycle.

You may make the following two assumptions: (1) writeback is not overlapped with reading the new data on the bus, and (2) all write hits to shared data require an upgrade transaction. As in the previous part, $CPI_{base} = 2.0$, and all calculations should be in processor cycles.

(c) Assume we add a unified write-back, write-allocate second-level cache to each processor (assuming a write-through, *no-write-allocate* first level cache with the same parameters as before). The L2 line size is 32 bytes. The local miss rate for all traffic to the L2 cache is 10%. The hit time to the L2 cache (which includes the time to transfer data to the L1 cache, but not the L1 hit time) is 6 cycles. The L2 cache is clocked at the same speed as the processor. Inclusion is maintained between the caches. Again, assume that the probability of having to replace a dirty block in the L2 cache on a miss that fetches a new block is 0.3.

What is the bus utilization of a single processor now? How many of these processors can the bus support without saturating?

(d) Finally, compute the average memory access time, *in processor cycles*, for the processor described in part (c). Be sure to include both instruction and data references. First show the equation, and then the numerical results.