1. An individual entry needs 5 bits for offset within block, and 10 bits for set # (2 sets @ 32 bytes each = 64 bytes per set; 64 KBytes @ 64 bytes per set = 1024 sets). This leaves 17 bits for the tag for each block. Since there are 2K blocks, the cache would need a total of 17 * 2048 = 34816 bits for tag in the entire cache.

   a. Dirty bit: required for a writeback cache. It indicates that the cache line has been modified but not written back to memory. The line must be flushed before it is replaced.

   b. Valid bit: required for any cache. DMA can change memory without updating the cache; the valid bit is used to indicate whether the cache line & tag should be "believed."

   c. Use bits: needed only for caches that aren't direct-mapped and use a non-RANDOM replacement algorithm. These bits indicate which cache line from a set should be selected for replacement if needed.

2. A TLB is a translation lookaside buffer. It's used to cache virtual to physical page translations for virtual memory. A fully-associative TLB will have a better hit rate for a given size, but will be more expensive. A set-associative TLB, on the other hand, uses less hardware, and thus could be made larger. This may partially offset the disadvantage of a worse hit rate than fully-associative.

3. The first system has an average access time of (10 ns * 0.90) + (80 ns * 0.10) = 17 ns. The second system has an average access time of (12 ns * 0.95) + (100 ns * 0.05) = 16.4 ns. This is the faster system; on average, the RAM is accessed once every (16.4 * (1/0.05) = 328 ns. Since this is longer than the cycle time of 200 ns, the DRAM should have time to cycle without slowing the processor down. However, there will be times when several misses will occur in quick succession. If this happens, overall bandwidth will drop.

4. Separate D & I caches: advantages are higher bandwidth to CPU, ability to use simpler hardware for each cache, non-interference of data & instructions for cache lines, simpler design for I cache (no writes). Disadvantages are twice as much cache hardware, possibly lower hit rate than a cache of size (D + I), inability to write self-modifying code.

   Unified cache: advantages are less hardware, fewer pins into the CPU, usually better hit rate than separate I & D caches. Disadvantages are lower bandwidth, more overhead for cached instructions (perhaps dirty bit), possible interference of I & D for locations, especially in a direct-mapped cache.

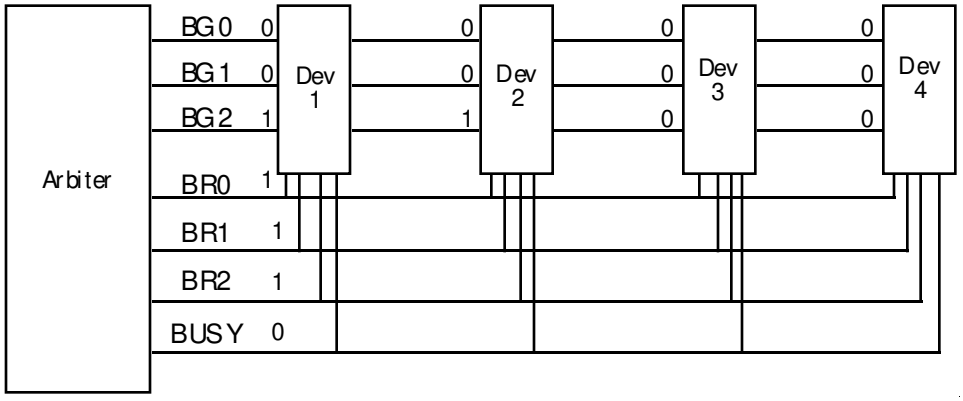
5. Programmed I/O: The CPU does all of the work, moving one word per instruction.adv = cheap & easy, disadv = slow, uses too much CPU time

   DMA: The CPU sets up the I/O device, and then lets the device transfer all of the data, freeing the CPU to do other work until the interrupt arrives. adv = relatively cheap, allows CPU to run during I/O, disadv = steals bus cycles, requires interrupts

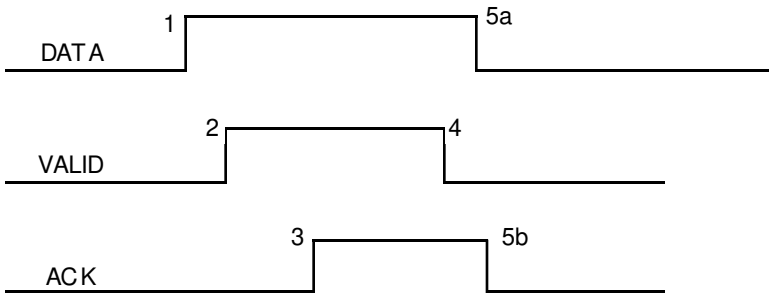
   channel-based: The CPU sends an I/O program to the device, which may include several individual I/O operations (position, read, reposition, write, scan, read). The CPU isn't notified until the program is complete. adv = allows CPU the most freedom – almost no CPU control necessary, disadv = expensive to implement (requires separate I/O channel processor)

6. Multiplexing address & data on a single set of wires in a bus allows the bus to have fewer wires. The cost is additional bus protocol complexity and lower bandwidth, especially from small transfers where the time cost of transmitting the address is amortized over few data cycles. Multiplexed buses are often used for I/O, where the number of wires is critical (cheaper connectors) and transfers are typically very large.

7. Since all of the devices make their requests simultaneously, device 2 will get the bus (and assert BUSY) because it is the device closest to the arbiter using the highest bus request level.



8.



There is no necessary ordering between 5a and 5b; they must both occur after 4, but can themselves occur in either order.

9. Overhead per transfer is 3 cycles + 1 cycle = 4 cycles. For an infinitely long transfer, bandwidth is simply 4 bytes/cycle * 1 cycle/100 ns = 40 MB/sec. If we restrict transfers to 64 bytes (16 data words), we require 4 + 16 = 20 cycles to transfer 64 bytes. This is a bandwidth of 64 bytes/20 cycles * 1 cycle/100 ns = 32 MB/s.