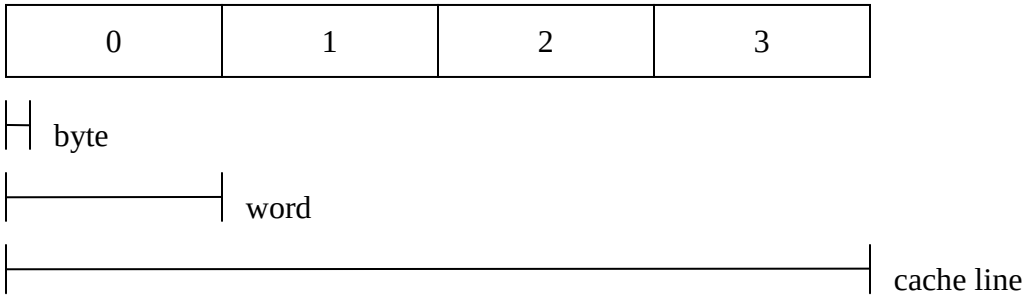
Critical word first or Sequential cache miss refill policy?

Some CPUs are configured to support "critical word first" cache miss line requests (i.e. wrapping bursts). Other CPUs are configured to fill cache line misses sequentially from the beginning of the line that had the miss. In a system in which cache line refills create 4 cycle bursts, if cache misses refill the line sequentially from the beginning of the line, a cache miss at an address in the last ¼ of the cache line will require 3 more cycles to receive its data from the bus as a cache miss at an address in the first ¼ of the cache line. If, in the same system cache misses refill the line starting at the word containing the address of the cache miss and then filling the remaining words of the cache line in order, wrapping to the beginning of the cache line then a cache miss will always be served with the same minimum number of cycles.

That seems like an improvement. Why doesn't everybody use critical word first cache refills?



While critical word first cache refills give the minimum latency, it requires the memory subsystem, when the first word of the burst is not the first word of the cache line, to begin the DDR burst from the word with the miss and then perform a second DDR burst to fill the first and any remaining words of the cache line. Performing two, instead of one, DDR burst for a cache refill adds overhead to the DDR accesses and decreases DDR bandwidth utilization efficiency. In the case of 4 cycle bursts, 75% of cache misses require 2 instead of just 1 DDR access.

Fortunately, most programs execute instructions mostly linearly and most functions access large data structures mostly linearly. This means that critical word first cache refills will have misses most frequently on the first word of cache lines and the efficiency cost to DDR bandwidth will be much less than 75% in the example above.

How to decide the cache miss refill policy?

System analysis and simulation is always the best way to make the determination. However, that might be difficult since the software to be run is rarely known at the time that the CPU architecture is set. In general, software with small data structures and software with short functions and/or long branches will have better performance with critical word first cache miss refills.

A word to the wise: some compiler optimizations favor fewer long instruction and data sequences. These will generally give better DDR efficiency, regardless of cache refill policy. On the other hand, some compiler optimizations minimize the number of cache misses by creating more compact instruction and data sequences, at the expense of DDR efficiency for the smaller number of cache misses.

-- Jonah Probell