

Chapter 02

Authors: John Hennessy & David Patterson

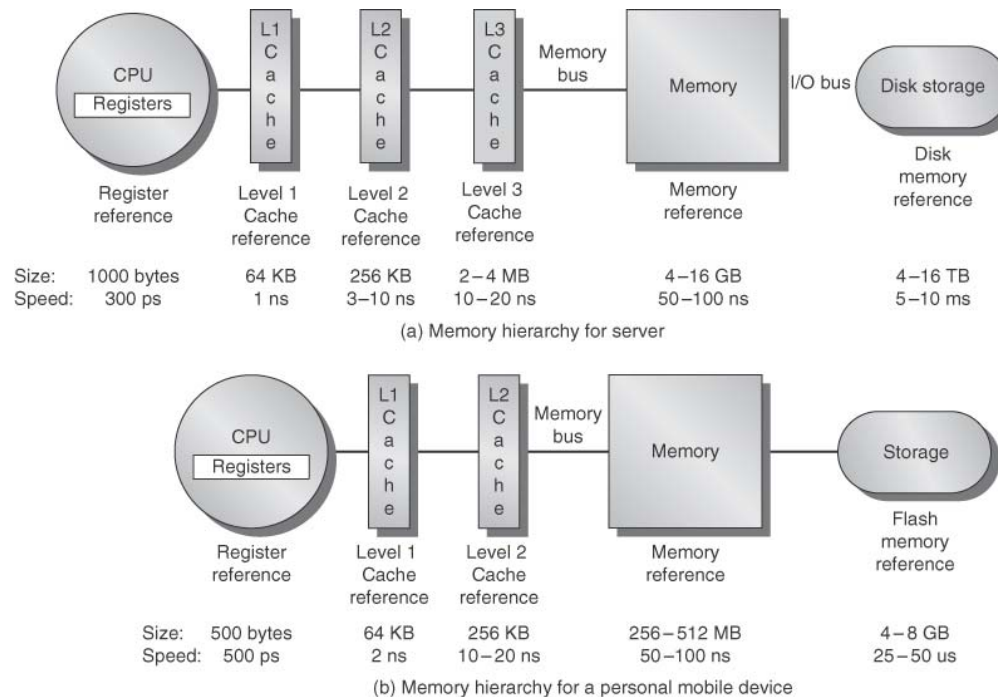


Figure 2.1 The levels in a typical memory hierarchy in a server computer shown on top (a) and in a personal mobile device (PMD) on the bottom (b). As we move farther away from the processor, the memory in the level below becomes slower and larger. Note that the time units change by a factor of 10^9 —from picoseconds to milliseconds—and that the size units change by a factor of 10^{12} —from bytes to terabytes. The PMD has a slower clock rate and smaller caches and main memory. A key difference is that servers and desktops use disk storage as the lowest level in the hierarchy while PMDs use Flash, which is built from EEPROM technology.

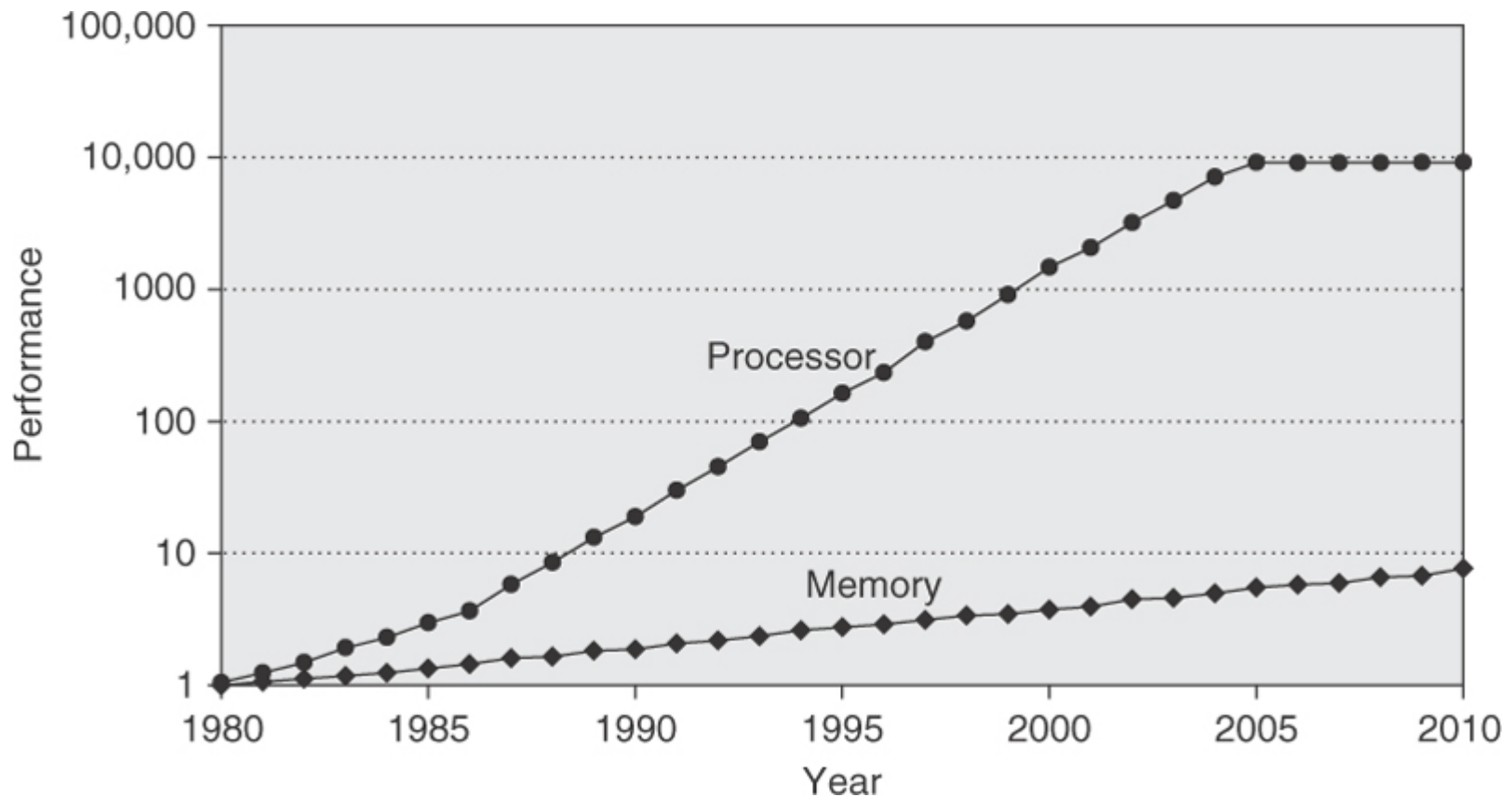


Figure 2.2 Starting with 1980 performance as a baseline, the gap in performance, measured as the difference in the time between processor memory requests (for a single processor or core) and the latency of a DRAM access, is plotted over time. Note that the vertical axis must be on a logarithmic scale to record the size of the processor–DRAM performance gap. The memory baseline is 64 KB DRAM in 1980, with a 1.07 per year performance improvement in latency (see Figure 2.13 on page 99). The processor line assumes a 1.25 improvement per year until 1986, a 1.52 improvement until 2000, a 1.20 improvement between 2000 and 2005, and no change in processor performance (on a per-core basis) between 2005 and 2010; see Figure 1.1 in Chapter 1.

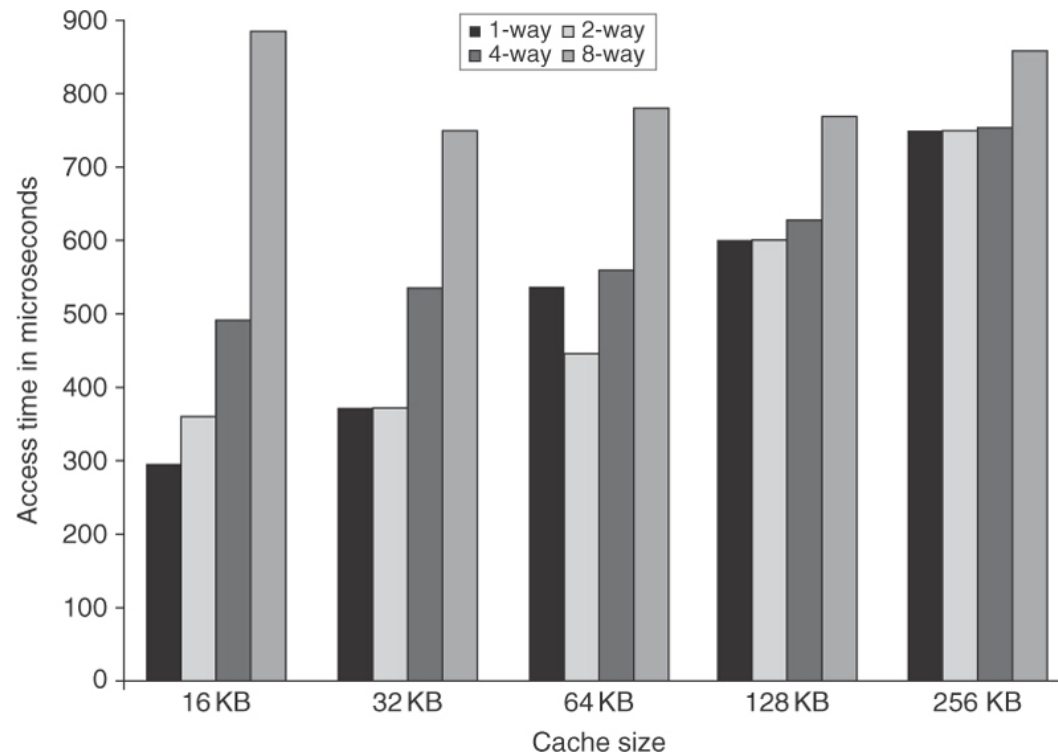


Figure 2.3 Access times generally increase as cache size and associativity are increased. These data come from the CACTI model 6.5 by Tarjan, Thoziyoor, and Jouppi [2005]. The data assume a 40 nm feature size (which is between the technology used in Intel’s fastest and second fastest versions of the i7 and the same as the technology used in the fastest ARM embedded processors), a single bank, and 64-byte blocks. The assumptions about cache layout and the complex trade-offs between interconnect delays (that depend on the size of a cache block being accessed) and the cost of tag checks and multiplexing lead to results that are occasionally surprising, such as the lower access time for a 64 KB with two-way set associativity versus direct mapping. Similarly, the results with eight-way set associativity generate unusual behavior as cache size is increased. Since such observations are highly dependent on technology and detailed design assumptions, tools such as CACTI serve to reduce the search space rather than precision analysis of the trade-offs.

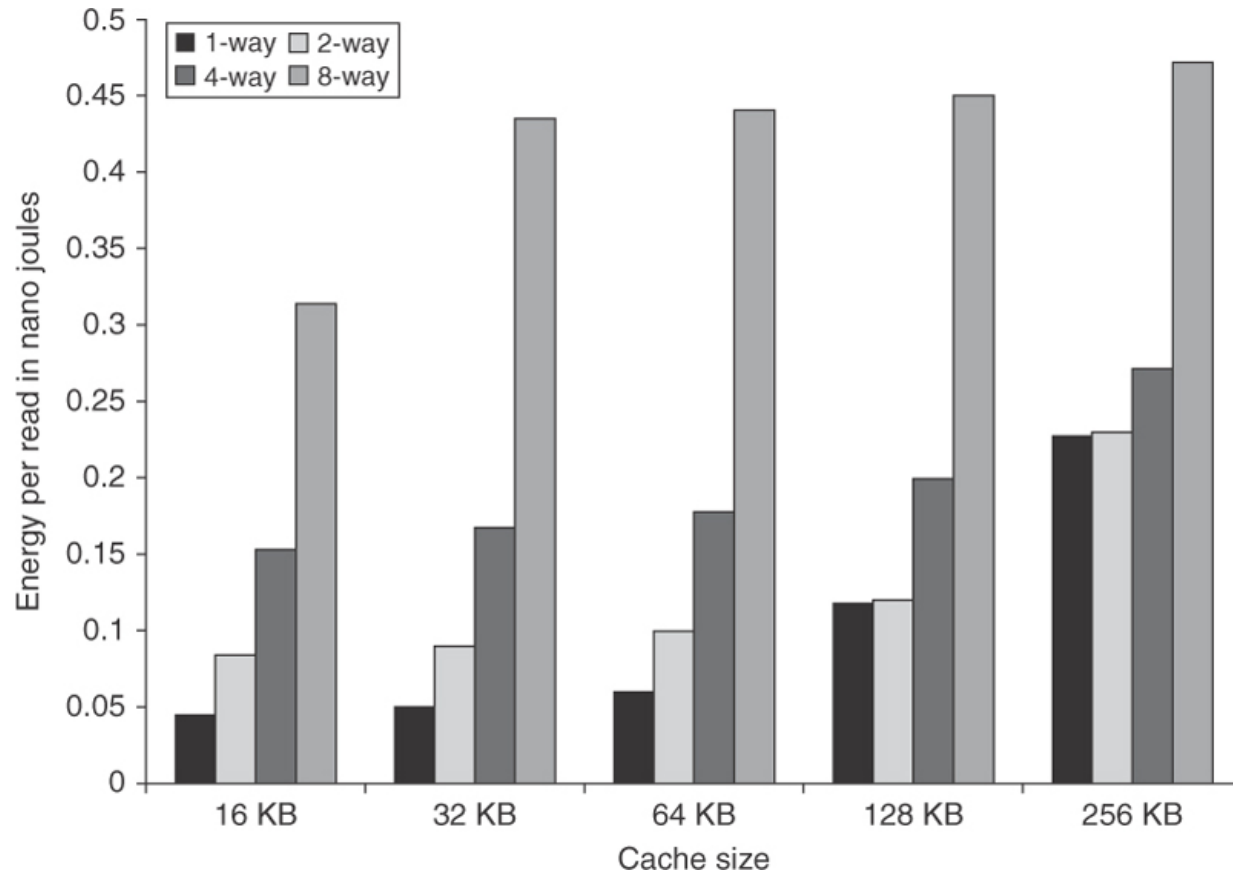


Figure 2.4 Energy consumption per read increases as cache size and associativity are increased. As in the previous figure, CACTI is used for the modeling with the same technology parameters. The large penalty for eight-way set associative caches is due to the cost of reading out eight tags and the corresponding data in parallel.

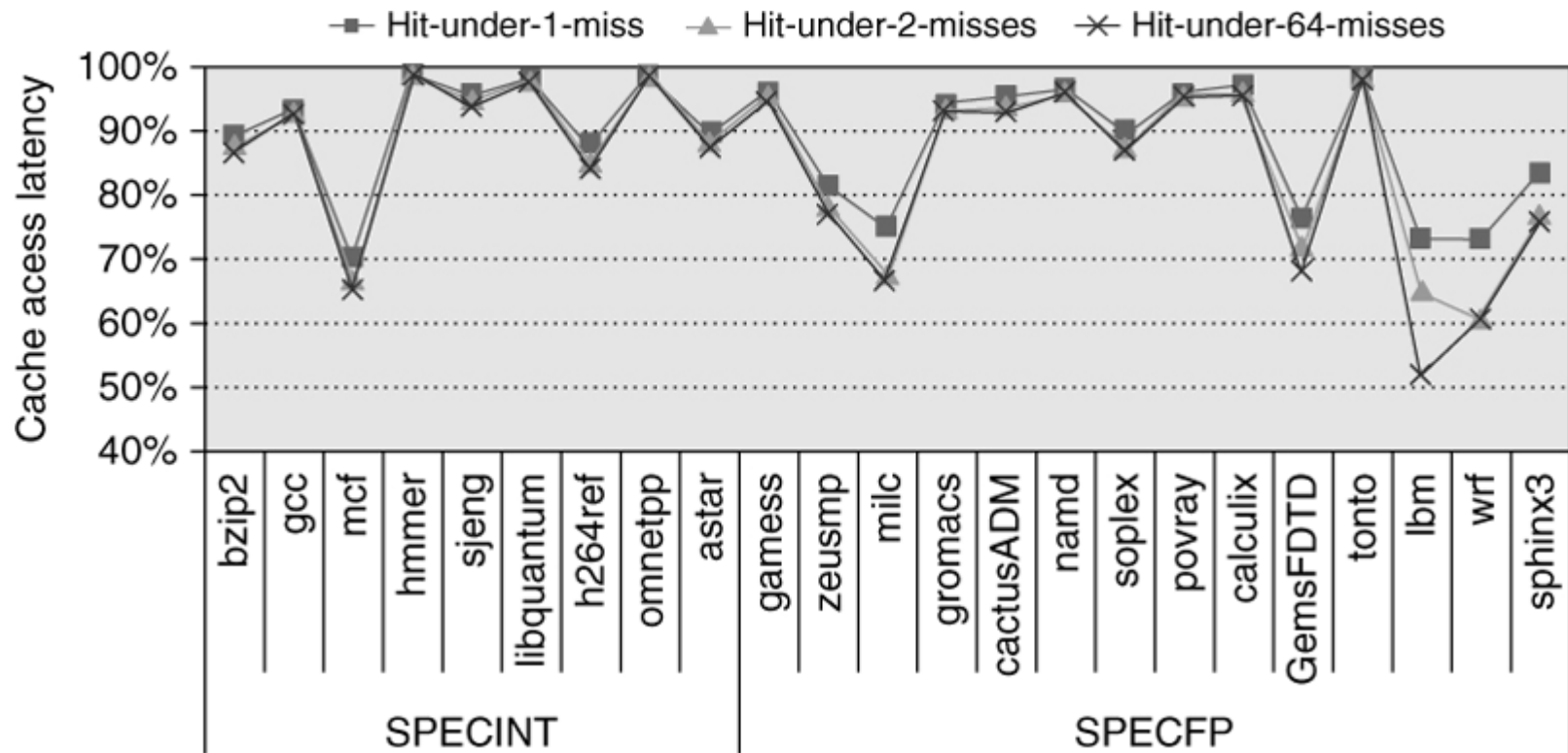


Figure 2.5 The effectiveness of a nonblocking cache is evaluated by allowing 1, 2, or 64 hits under a cache miss with 9 SPECINT (on the left) and 9 SPECFP (on the right) benchmarks. The data memory system modeled after the Intel i7 consists of a 32KB L1 cache with a four cycle access latency. The L2 cache (shared with instructions) is 256 KB with a 10 clock cycle access latency. The L3 is 2 MB and a 36-cycle access latency. All the caches are eight-way set associative and have a 64-byte block size. Allowing one hit under miss reduces the miss penalty by 9% for the integer benchmarks and 12.5% for the floating point. Allowing a second hit improves these results to 10% and 16%, and allowing 64 results in little additional improvement.

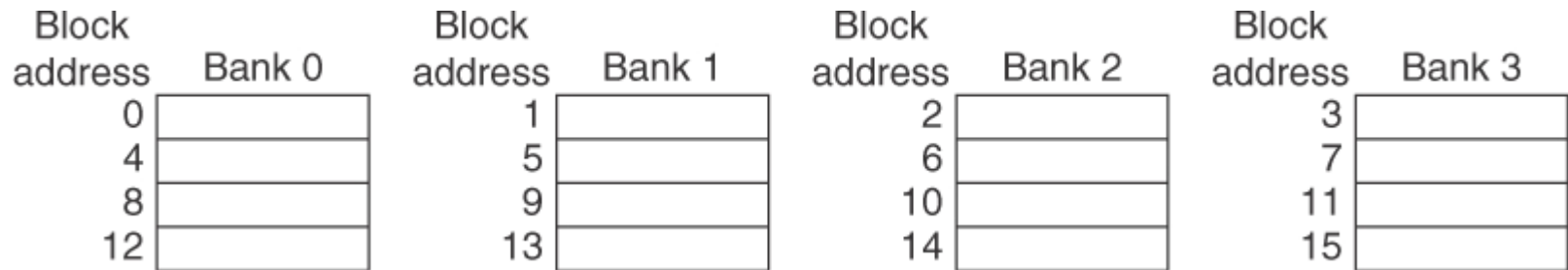


Figure 2.6 Four-way interleaved cache banks using block addressing. Assuming 64 bytes per blocks, each of these addresses would be multiplied by 64 to get byte addressing.

Write address	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

Write address	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

Figure 2.7 To illustrate write merging, the write buffer on top does not use it while the write buffer on the bottom does. The four writes are merged into a single buffer entry with write merging; without it, the buffer is full even though three-fourths of each entry is wasted. The buffer has four entries, and each entry holds four 64-bit words. The address for each entry is on the left, with a valid bit (V) indicating whether the next sequential 8 bytes in this entry are occupied. (Without write merging, the words to the right in the upper part of the figure would only be used for instructions that wrote multiple words at the same time.)

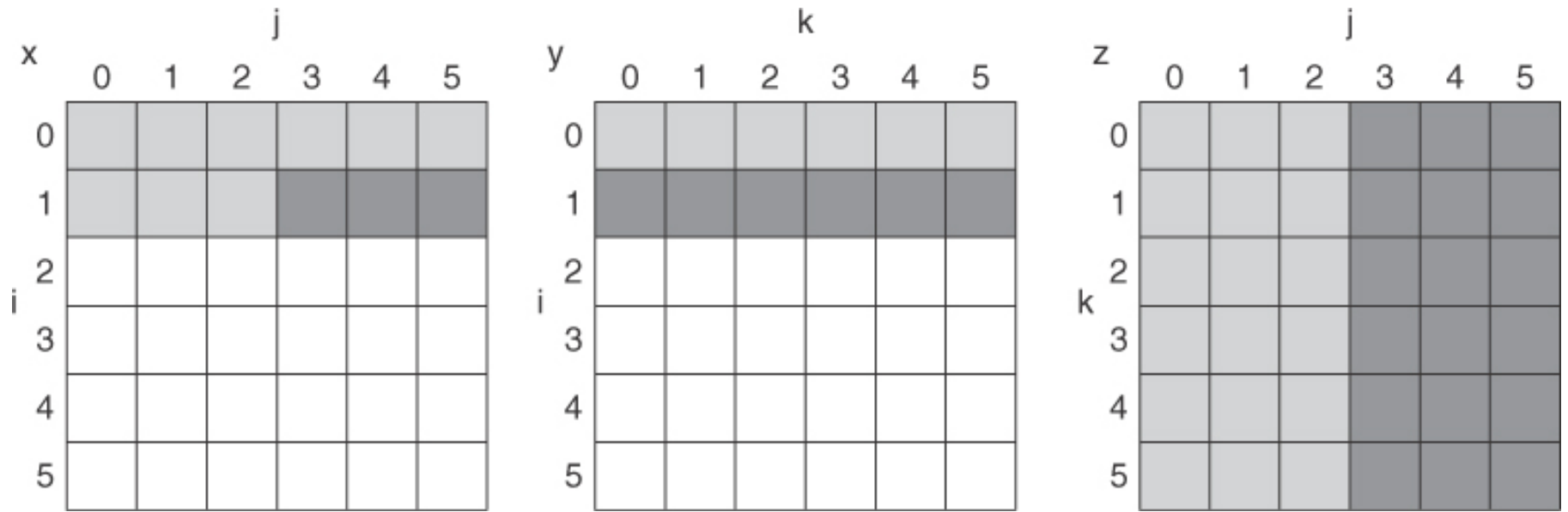


Figure 2.8 A snapshot of the three arrays x, y, and z when $N = 6$ and $i = 1$. The age of accesses to the array elements is indicated by shade: white means not yet touched, light means older accesses, and dark means newer accesses. Compared to Figure 2.9, elements of y and z are read repeatedly to calculate new elements of x. The variables i, j, and k are shown along the rows or columns used to access the arrays.

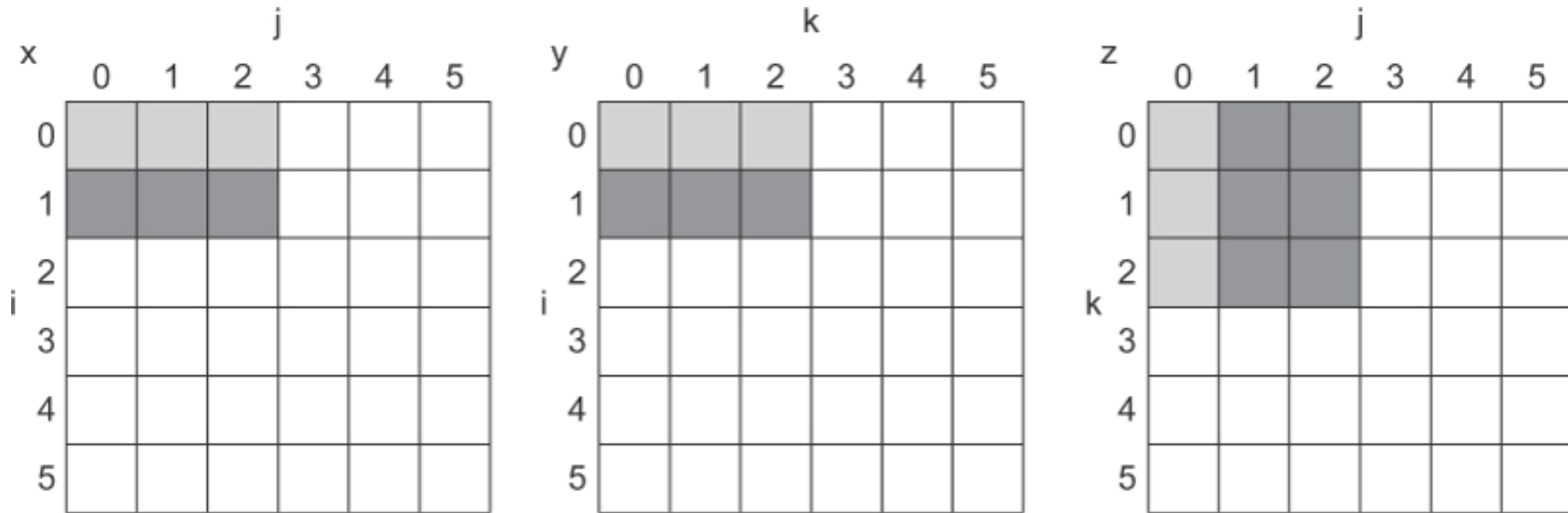


Figure 2.9 The age of accesses to the arrays x , y , and z when $B = 3$. Note that, in contrast to Figure 2.8, a smaller number of elements is accessed.

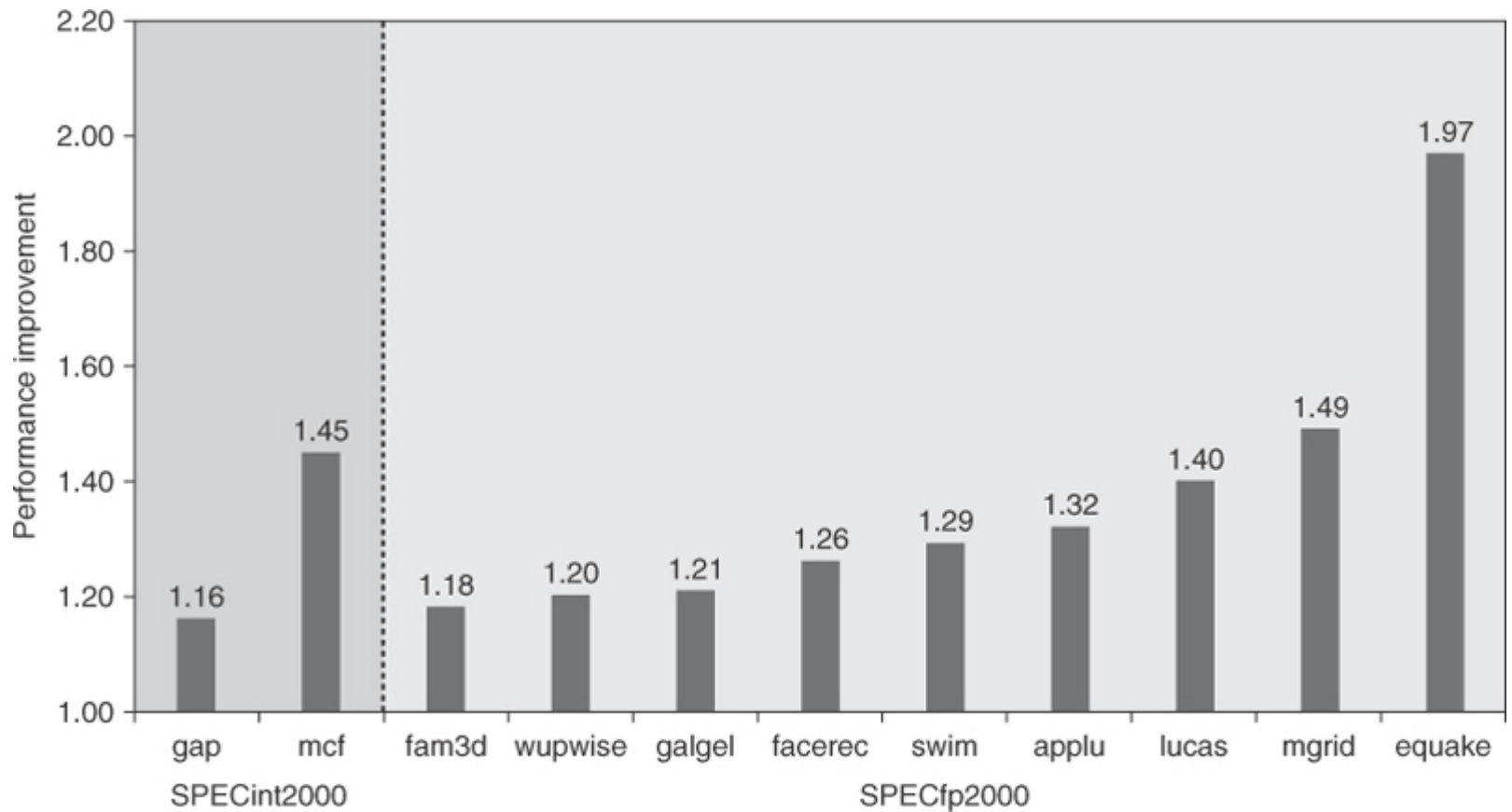


Figure 2.10 Speedup due to hardware prefetching on Intel Pentium 4 with hardware prefetching turned on for 2 of 12 SPECint2000 benchmarks and 9 of 14 SPECfp2000 benchmarks. Only the programs that benefit the most from prefetching are shown; prefetching speeds up the missing 15 SPEC benchmarks by less than 15% [Singhal 2004].

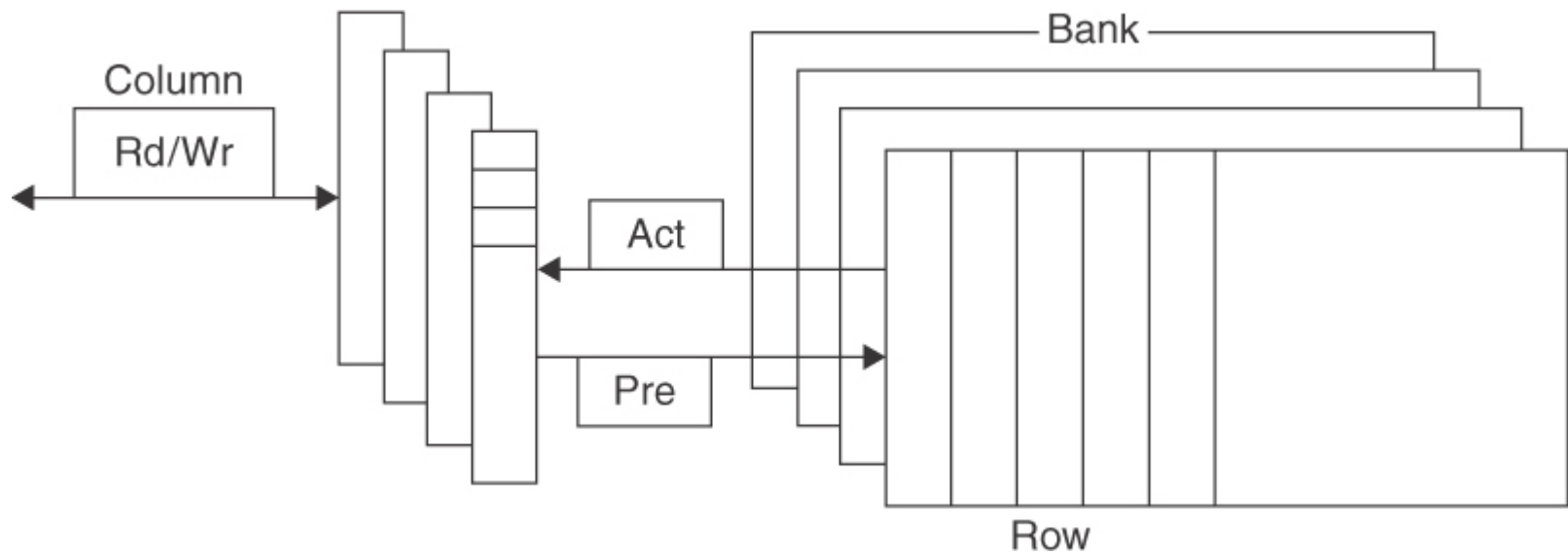


Figure 2.12 Internal organization of a DRAM. Modern DRAMs are organized in banks, typically four for DDR3. Each bank consists of a series of rows. Sending a PRE (precharge) command opens or closes a bank. A row address is sent with an Act (activate), which causes the row to transfer to a buffer. When the row is in the buffer, it can be transferred by successive column addresses at whatever the width of the DRAM is (typically 4, 8, or 16 bits in DDR3) or by specifying a block transfer and the starting address. Each command, as well as block transfers, are synchronized with a clock.

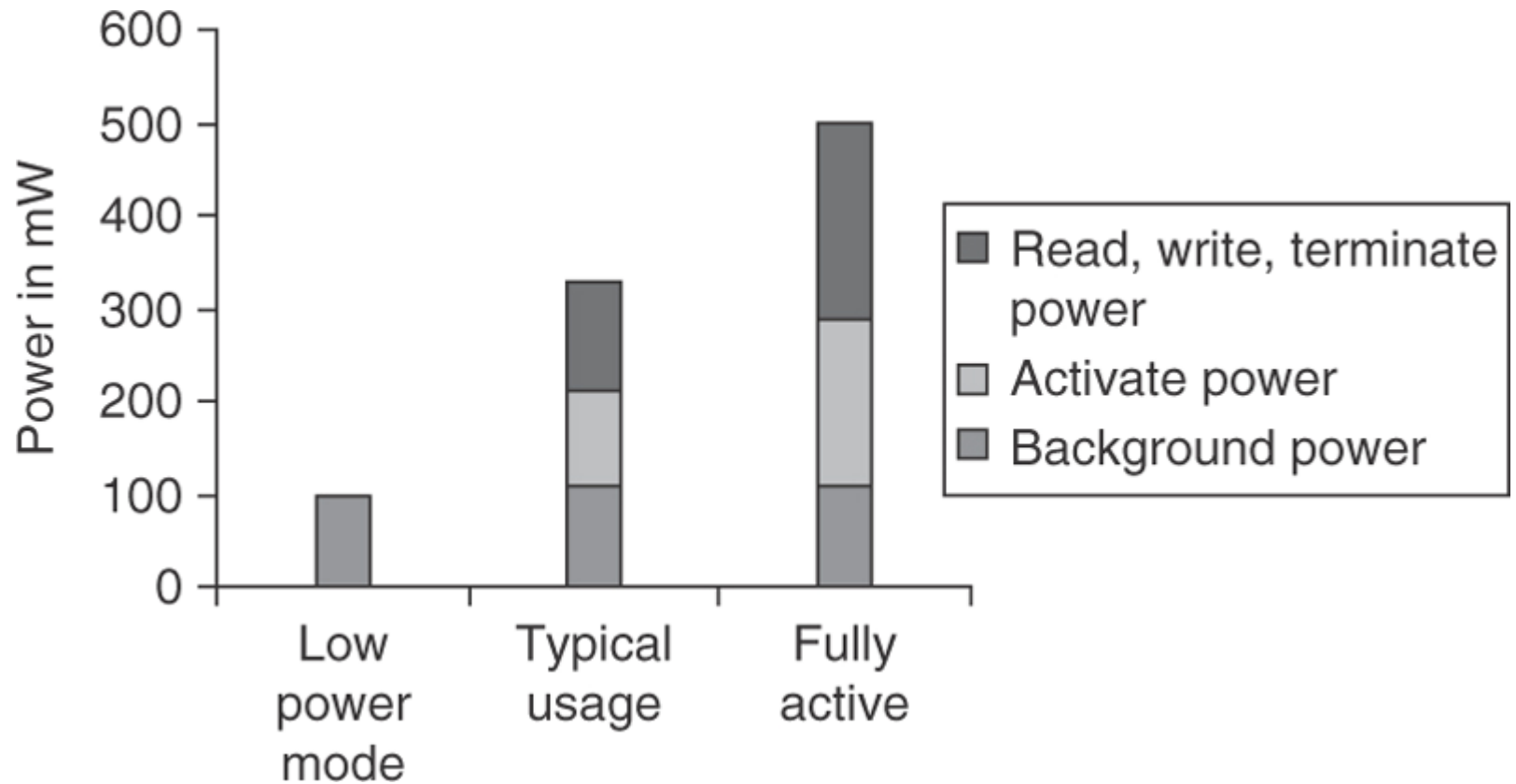


Figure 2.15 Power consumption for a DDR3 SDRAM operating under three conditions: low power (shutdown) mode, typical system mode (DRAM is active 30% of the time for reads and 15% for writes), and fully active mode, where the DRAM is continuously reading or writing when not in precharge. Reads and writes assume bursts of 8 transfers. These data are based on a Micron 1.5V 2Gb DDR3-1066.

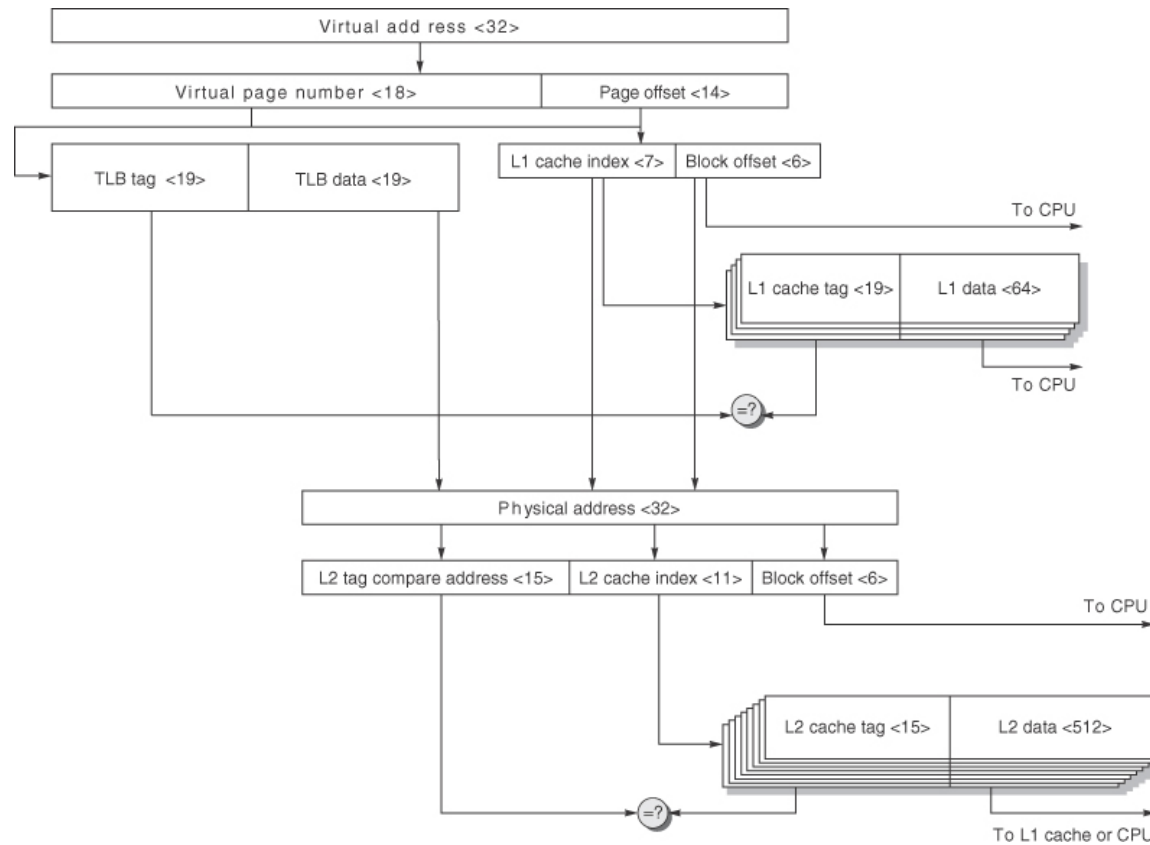


Figure 2.16 The virtual address, physical address, indexes, tags, and data blocks for the ARM Cortex-A8 data caches and data TLB. Since the instruction and data hierarchies are symmetric, we show only one. The TLB (instruction or data) is fully associative with 32 entries. The L1 cache is four-way set associative with 64-byte blocks and 32 KB capacity. The L2 cache is eight-way set associative with 64-byte blocks and 1 MB capacity. This figure doesn't show the valid bits and protection bits for the caches and TLB, nor the use of the way prediction bits that would dictate the predicted bank of the L1 cache.

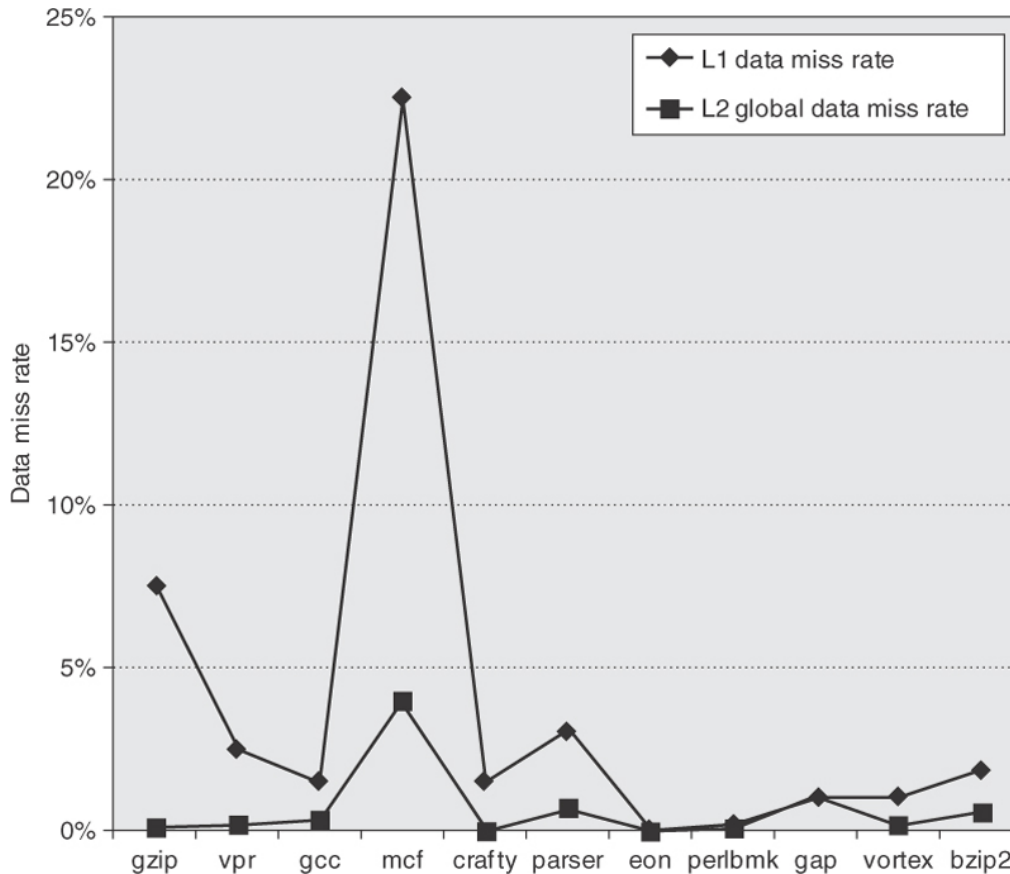


Figure 2.17 The data miss rate for ARM with a 32 KB L1 and the global data miss rate for a 1 MB L2 using the integer Minnespec benchmarks are significantly affected by the applications. Applications with larger memory footprints tend to have higher miss rates in both L1 and L2. Note that the L2 rate is the global miss rate, that is counting all references, including those that hit in L1. Mcf is known as a cache buster.

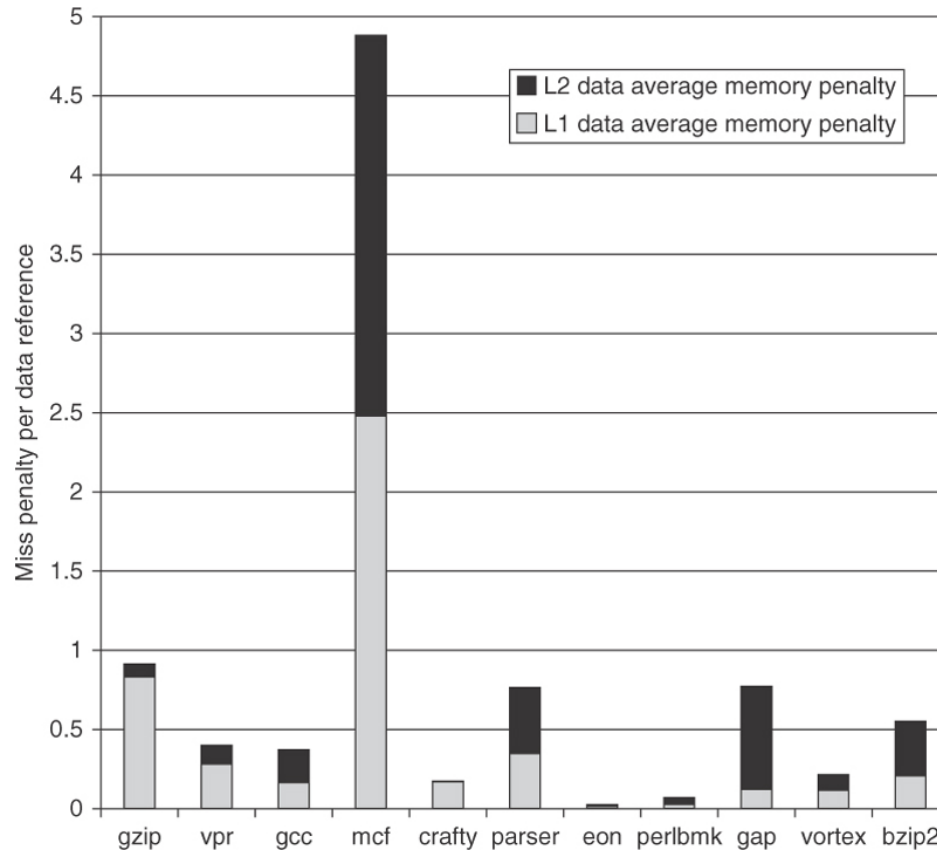


Figure 2.18 The average memory access penalty per data memory reference coming from L1 and L2 is shown for the ARM processor when running MinnieSpec. Although the miss rates for L1 are significantly higher, the L2 miss penalty, which is more than five times higher, means that the L2 misses can contribute significantly.

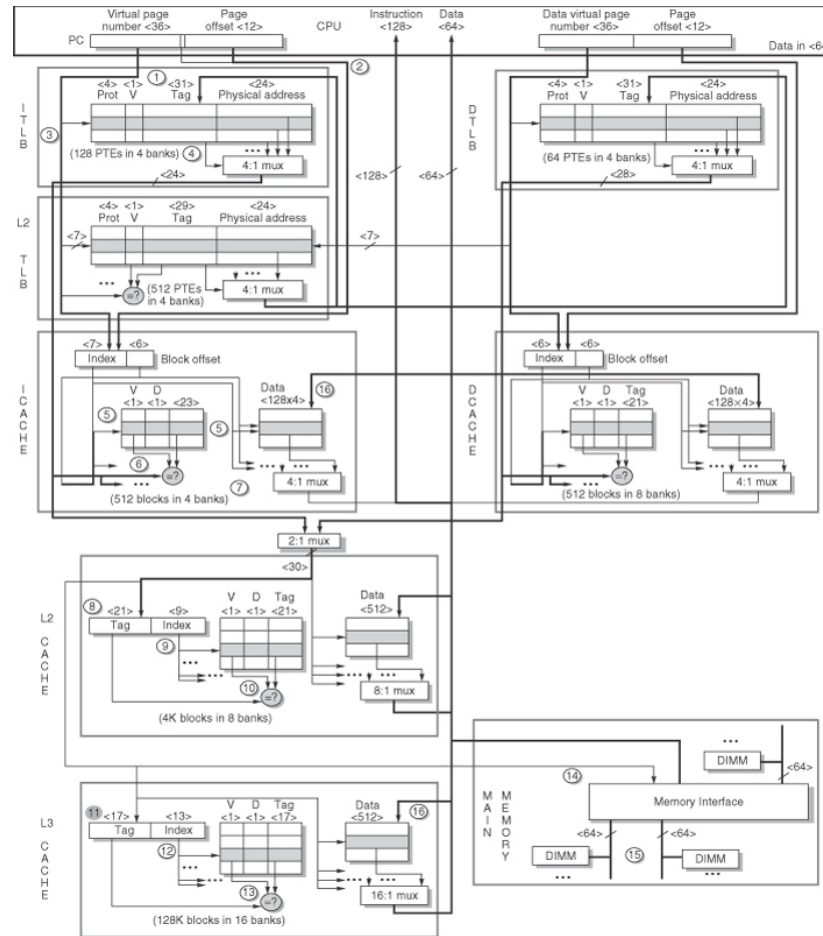


Figure 2.21 The Intel i7 memory hierarchy and the steps in both instruction and data access. We show only reads for data. Writes are similar, in that they begin with a read (since caches are write back). Misses are handled by simply placing the data in a write buffer, since the L1 cache is not write allocated.

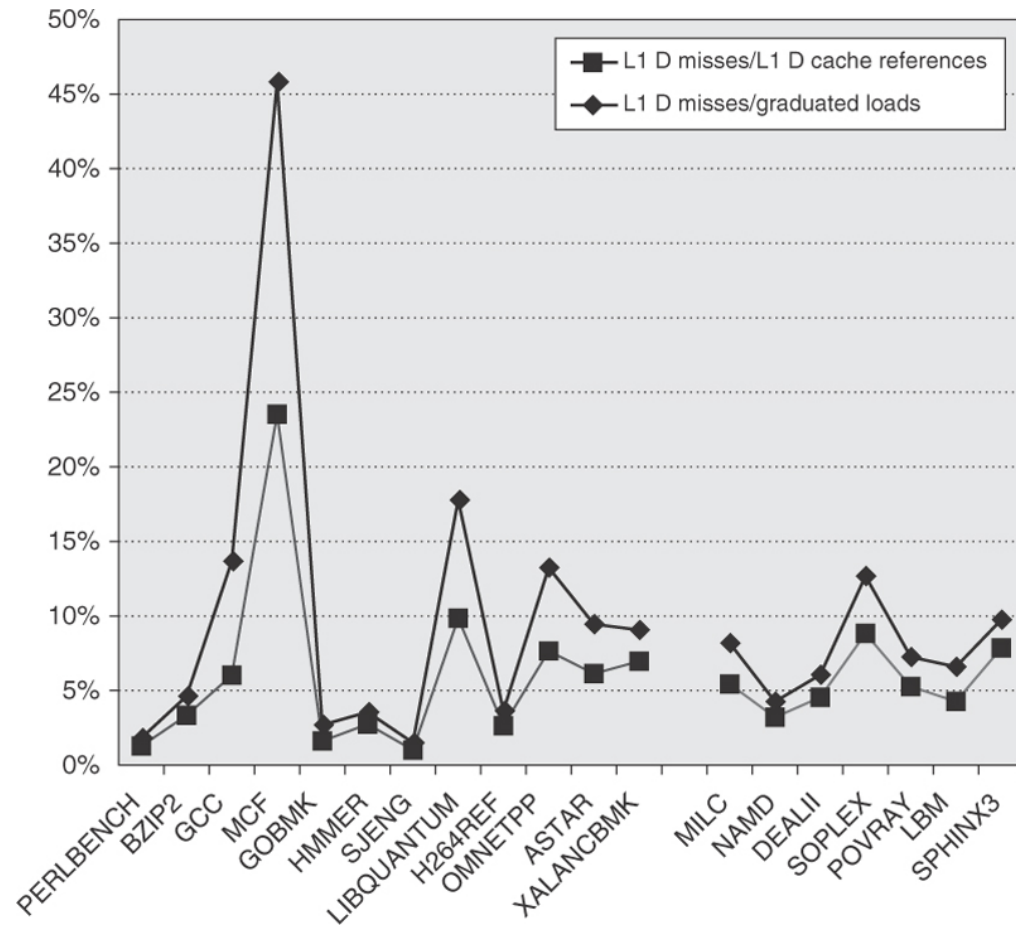


Figure 2.22 The L1 data cache miss rate for 17 SPEC CPU2006 benchmarks is shown in two ways: relative to the actual loads that complete execution successfully and relative to all the references to L1, which also includes prefetches, speculative loads that do not complete, and writes, which count as references, but do not generate misses. These data, like the rest in this section, were collected by Professor Lu Peng and Ph.D. student Ying Zhang, both of Louisiana State University, based on earlier studies of the Intel Core Duo and other processors (see Peng et al. [2008]).

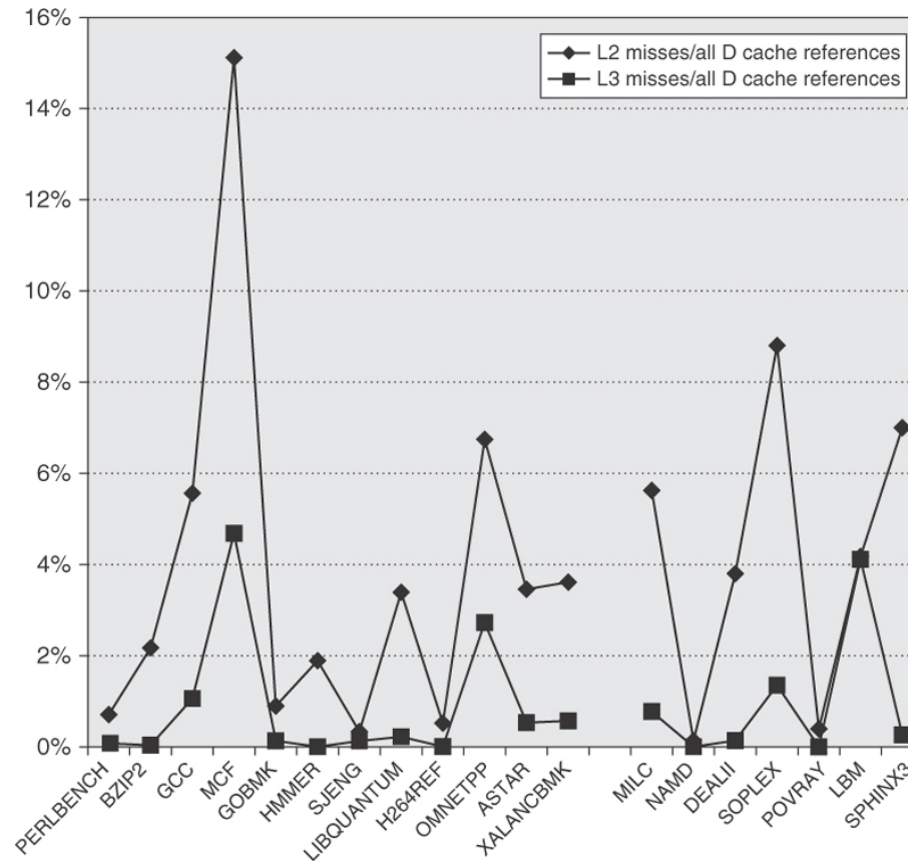


Figure 2.24 The L2 and L3 data cache miss rates for 17 SPEC CPU2006 benchmarks are shown relative to all the references to L1, which also includes prefetches, speculative loads that do not complete, and program-generated loads and stores. These data, like the rest in this section, were collected by Professor Lu Peng and Ph.D. student Ying Zhang, both of Louisiana State University.

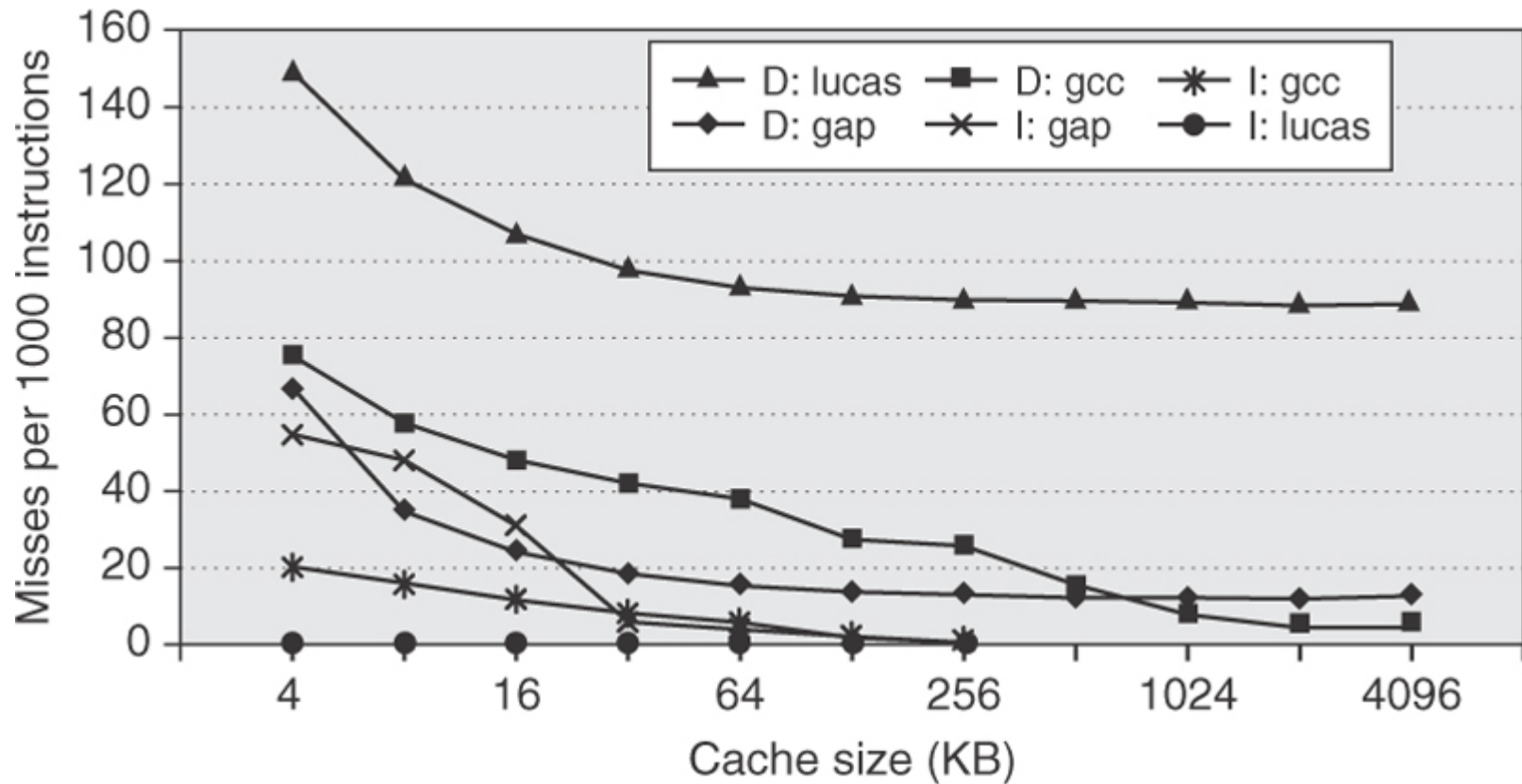


Figure 2.26 Instruction and data misses per 1000 instructions as cache size varies from 4 KB to 4096 KB. Instruction misses for gcc are 30,000 to 40,000 times larger than lucas, and, conversely, data misses for lucas are 2 to 60 times larger than gcc. The programs gap, gcc, and lucas are from the SPEC2000 benchmark suite.

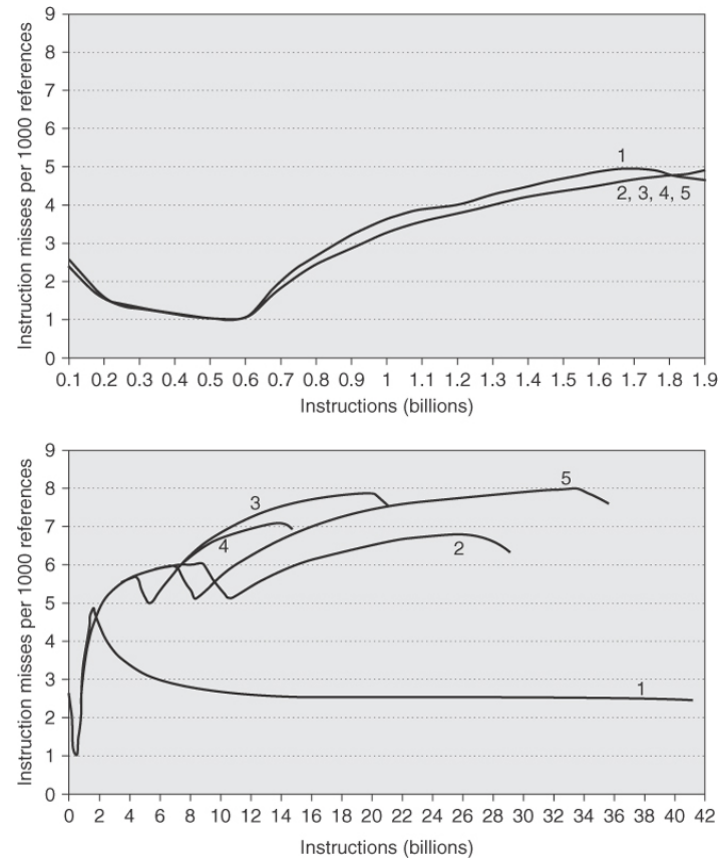


Figure 2.27 Instruction misses per 1000 references for five inputs to the perl benchmark from SPEC2000. There is little variation in misses and little difference between the five inputs for the first 1.9 billion instructions. Running to completion shows how misses vary over the life of the program and how they depend on the input. The top graph shows the running average misses for the first 1.9 billion instructions, which starts at about 2.5 and ends at about 4.7 misses per 1000 references for all five inputs. The bottom graph shows the running average misses to run to completion, which takes 16 to 41 billion instructions depending on the input. After the first 1.9 billion instructions, the misses per 1000 references vary from 2.4 to 7.9 depending on the input. The simulations were for the Alpha processor using separate L1 caches for instructions and data, each two-way 64 KB with LRU, and a unified 1 MB direct-mapped L2 cache.

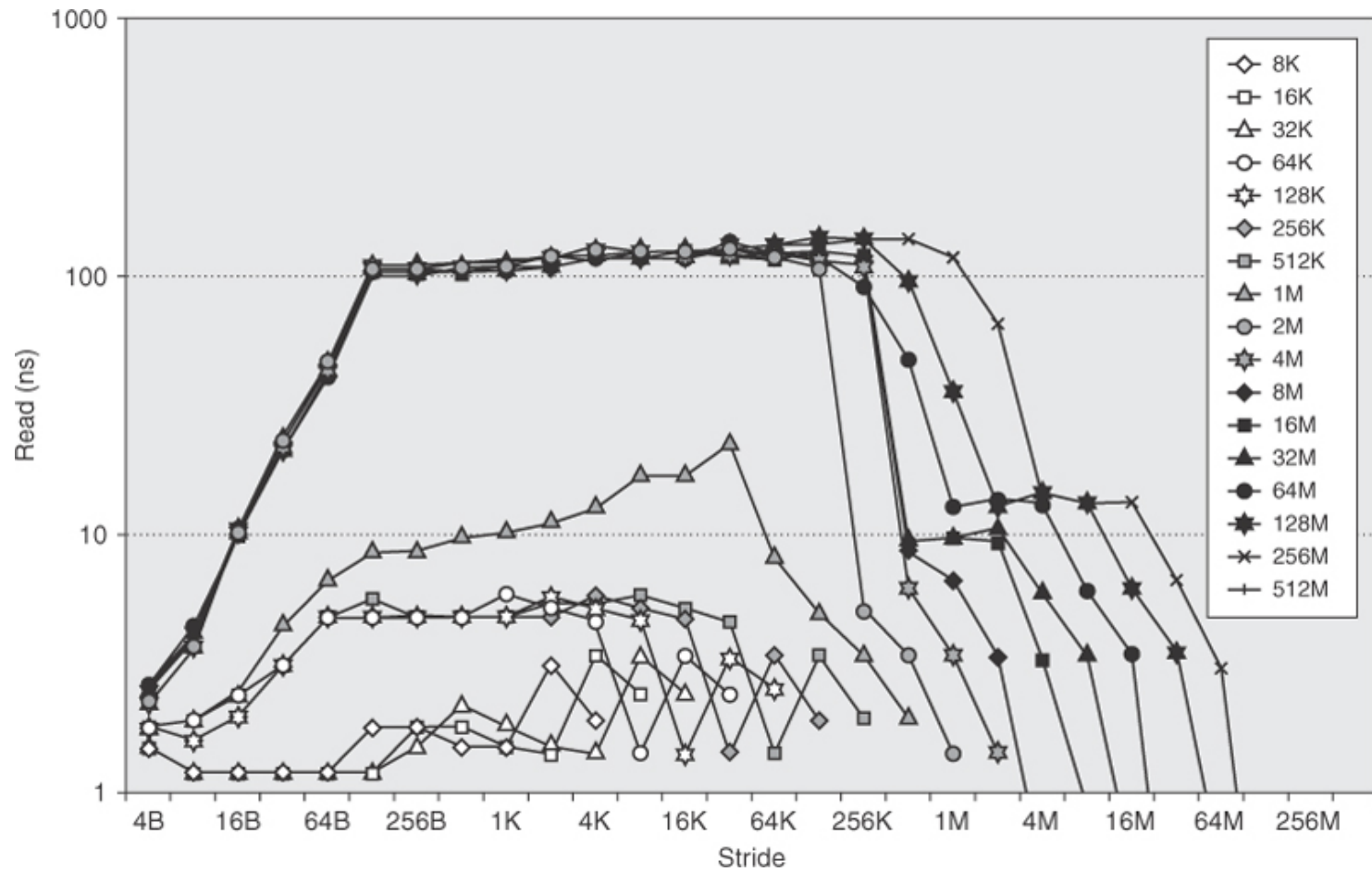


Figure 2.30 Sample results from program in Figure 2.29.

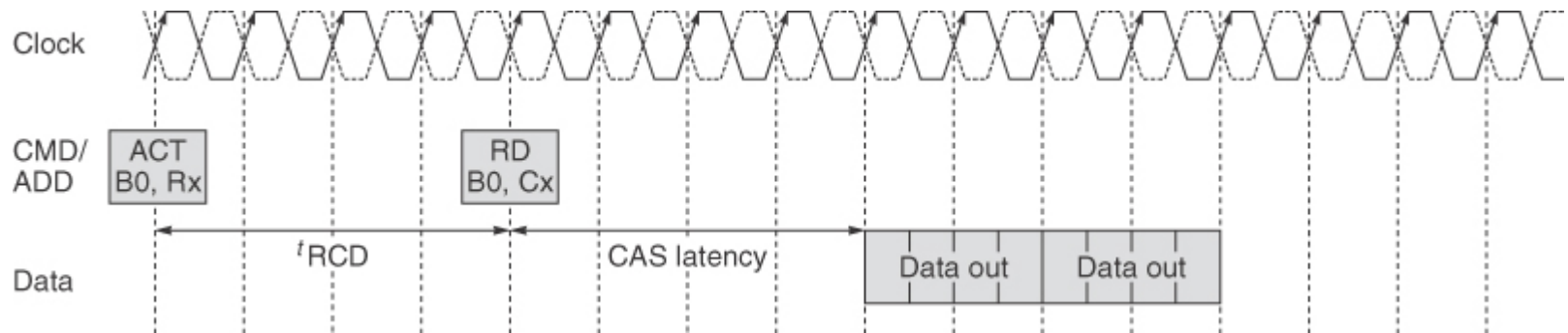


Figure 2.31 DDR2 SDRAM timing diagram.

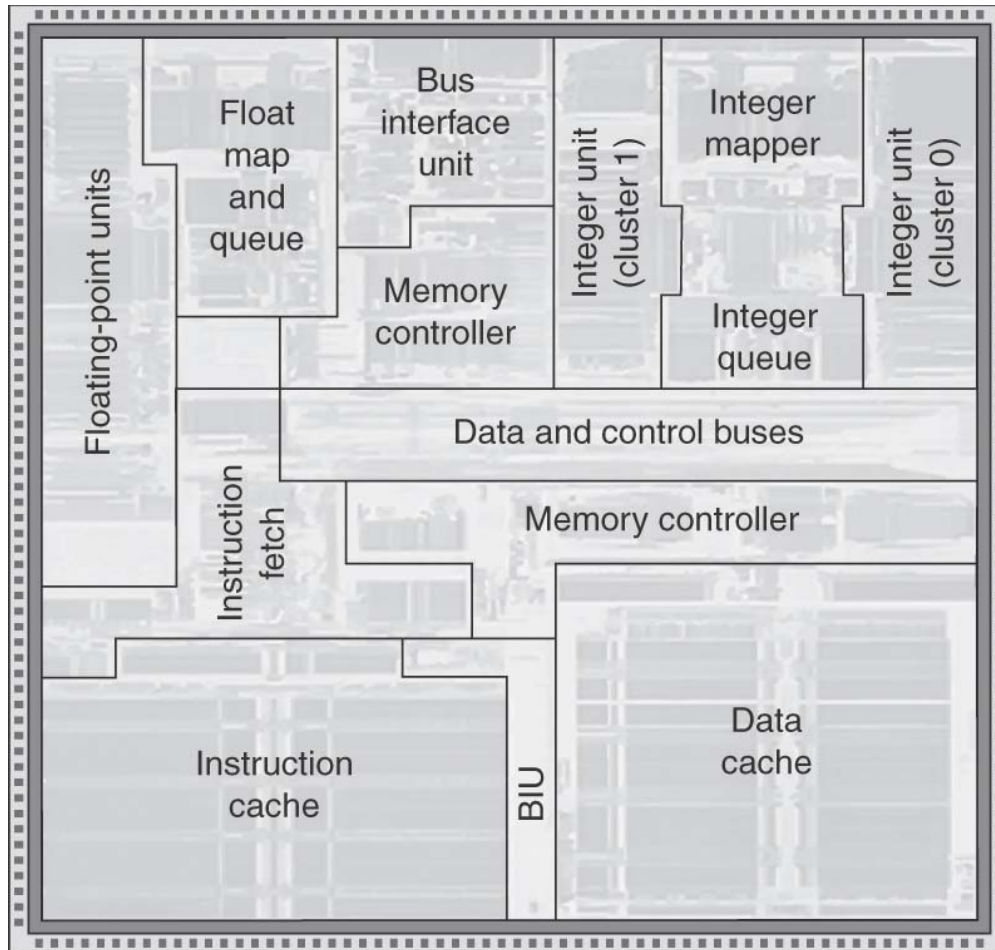


Figure 2.33 Floorplan of the Alpha 21264 [Kessler 1999].