Memory Systems

Doug Burger

University of Wisconsin-Madison

A computer's memory system is the repository for all the information that the computer's central processing unit (CPU, or processor) uses and produces. A perfect memory system is one that can supply immediately any datum that the CPU requests. This ideal memory is not practically implementable, however, as the three factors of memory capacity, speed, and cost are directly in opposition.

By placing smaller, faster memories in front of larger, slower, and cheaper memories, the performance of the memory system may approach that of a perfect memory system—at a reasonable cost. The **memory hierarchies** of modern general-purpose computers generally contain registers at the top, followed by one or more levels of **cache memory**, **main memory** (all three are semiconductor memory) and **virtual memory** (on a magnetic or optical disk). Figure 1 shows a memory hierarchy typical of today's (1995) commodity systems.

Performance of a memory system is measured in terms of **latency** and **bandwidth**. The latency of a memory request is how long it takes the memory system to produce the result of the request. The bandwidth of a memory system is the rate at which the memory system can accept requests and produce results. The memory hierarchy improves average latency by quickly returning results that are found in the higher levels of the hierarchy. The memory hierarchy usually reduces bandwidth requirements by intercepting a fraction of the memory requests at higher levels of the hierarchy. Some machines, such as high-performance vector machines, may have fewer levels in the hierarchy—increasing cost for better predictability and performance. Some of these machines contain no caches at all, relying on large arrays of main memory banks to supply very high bandwidth. Pipelined accesses of operands reduce the performance impact of long latencies in these machines.

Cache memories are a general solution to improving the performance of a memory system. Although caches

are smaller than typical main memory sizes, they ideally contain the most frequently-accessed portions of main memory. By keeping the most heavily-used data near the CPU, caches can service a large fraction of the requests without needing accessing main memory (the fraction serviced is called the hit rate). Caches require locality of reference to work well transparently—they assume that accessed memory words will be accessed again quickly (temporal locality), and that memory words adjacent to an accessed word will be accessed soon after the access in question (spatial locality). When the CPU issues a request for a datum not in the cache (a cache miss), the cache loads that datum and some number of adjacent data (a cache block) into itself from main memory.

To reduce cache misses, some caches are associative—a cache may place a given block in one of several places, collectively called a set. This set is content-addressable; a block may or may not be accessed based on an address tag, one of which is coupled with each block. When a new block is brought into a set, and the set is full, the cache's replacement policy dictates which of the old blocks should be removed from the cache to make room for the new. Most caches use an approximation of least-recently-used (LRU) replacement, in which the block last accessed farthest in the past is the one that the cache replaces.

Main memory, or backing store, consists of banks of dense semiconductor memory. Since each memory (DRAM) chip has a small off-chip bandwidth, rows of these chips are placed together to form a bank, and multiple banks are used to increase the bandwidth out of the main memory system. When a bank is accessed, it remains busy for a period of time, during which the processor may make no other accesses to that bank. By increasing the number of banks (parallel or interleaved banks), the chance that the processor issues two conflicting requests to the same bank is reduced.

Systems generally require a greater number of memory locations than are available in the main memory (i.e. a larger address space). The entire portion of the address space that the CPU uses is stored on large magnetic or optical disks; this is called the virtual address space, or virtual memory. The most frequently used sections of the virtual memory are kept in main memory (physical memory), and are moved back and forth in units called pages. The location at which a virtual address lies in main memory is called the physical address. Since a much

larger address space (virtual memory) is being mapped onto a much smaller one (physical memory), the CPU must translate the memory addresses issued by a program (virtual addresses) into their corresponding locations in physical memory (physical addresses). This mapping is maintained in a memory structure called the page table. When the CPU attempts to access a virtual address that does not have a corresponding entry in physical memory, a page fault occurs. Since a page fault requires an access to a slow mechanical storage device (such as a disk), the CPU usually switches to a different task while the needed page is read off of the disk.

Since every memory request issued by the CPU requires an address translation, which in turn requires an access to the page table stored in memory, a translation lookaside buffer (TLB) is used to reduce the number of page table lookups. The most frequent virtual-to-physical mappings are kept in the TLB, which is a small associative memory tightly coupled with the CPU. If the mapping is found in the TLB, the translation is performed extremely quickly and no access to the page table needs to be made. Virtual memory allows systems to run much larger and many more programs than are able to fit in main memory, greatly enhancing their flexibility.

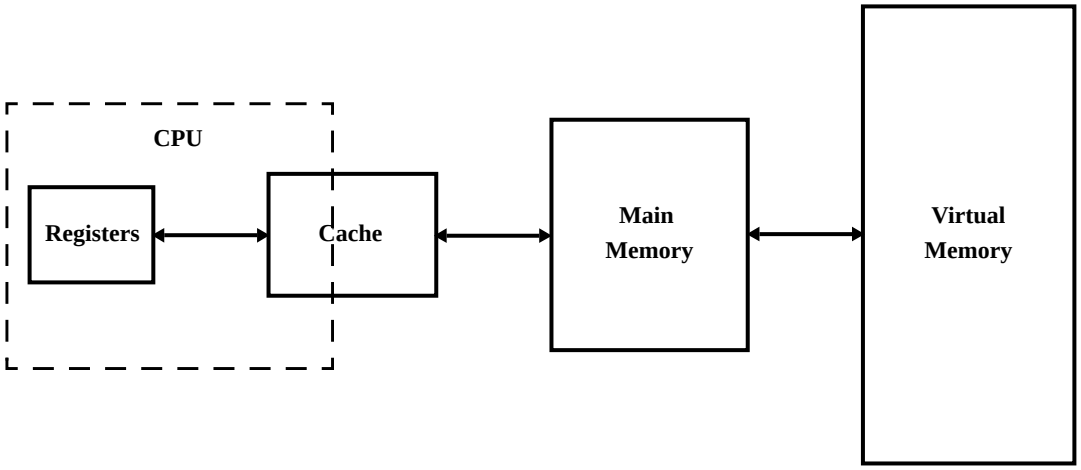
The use of a memory hierarchy allows a computer system to approximate a memory that is large, fast, and (relatively) inexpensive. The virtual memory system on disk allows a memory have a very large capacity. The use of semiconductor main memory prevents the processor from having to make frequent, extremely slow accesses to disk. Cache memory allows accesses to be made with very low latency most of the time, with an occasional request to main memory. Although the sizes, parameters, and even composition of these hierarchy levels change over time, the concept of a memory hierarchy is fundamental and will always be in use.

Defining Terms

Bandwidth: The rate at which the memory system can service requests.

Cache memory: A small, fast, redundant memory used to store the most frequently-accessed parts of the main memory.

Interleaving: A technique for connecting multiple memory modules together in order to improve the bandwidth of the memory system.



Very Fast (Electronic speeds)	Very Fast (Electronic speeds)	Fast (Electronic Speeds)	Very Slow (Mechanical Speeds)
Semiconductor SRAM	Semiconductor SRAM	Semiconductor DRAM	Magnetic/Optical
Tiny 128 bytes - 4Kbytes	Small 32 Kbytes - 4 Mbytes	Large 4 Mbytes - 512 Mbytes	Very Large 40 MBytes - 8 Gbytes

FIGURE 1

Latency: The time between the initiation of a memory request and its completion.

Main memory: The repository for the physical address space, made up of semiconductor DRAM banks.

Memory hierarchy: Successive levels of different types of memory, which attempt to approximate a single large, fast, and cheap memory structure.

Virtual memory: A memory space implemented by storing the more frequently-accessed parts in main memory and less frequently-accessed parts on disk.

Figure 1. A typical memory hierarchy in 1996

Source: Dorf, R. C. 1992. *The Electrical Engineering Handbook*, 1st ed., p. 1932. CRC Press, Inc., Boca Raton, FL. With permission.