Controllers are the parts of a mechatronic system that detect the difference between a system's current state (as measured by one or more sensors) and the state the system should be in and take action to correct the difference between the two (by operating one or more actuators). Although controllers can be mechanical or electrical, today they commonly take the form of **microprocessors** preprogrammed to perform one or more specific tasks (such as controlling the different movements of a paint-spraying robot or the different operating modes of a videocassette recorder [VCR]). These contain the final component of the system: the software that operates them according to control theory.

As microprocessors have replaced many forms of mechanical control since their popularization in the early 1970s, so the balance of mechatronics has shifted away from mechanical and toward electronic. In the 1970s, mechatronic devices were typified by automatically operated shop doors and relay-controlled vending machines. In the 1980s, microprocessor-controlled toasters, washing machines, VCRs, and antilock braking systems (ABS) on cars represented the state of the art.

During the 1990s, mechatronic devices incorporated a greater proportion of optical-electronic components, such as charge-coupled devices (CCDs), the light-sensing elements used in digital cameras and scanners, and greater use of computer networks to enable remote operation. The twenty-first century is likely to see more sophisticated mechatronic machine vision systems and an even greater use of mechatronic devices in remote and hostile environments such as space.

FURTHER READING

Åström, Karl J., and Björn Wittenmark. *Computer-Controlled Systems: Theory and Design.* Englewood Cliffs, N.J.: Prentice Hall, 1984; 3rd ed., Upper Saddle River, N.J.,1997.

Bolton, W. *Mechatronics: Electronic Control Systems in Mechanical Engineering.* Harlow, Essex, England, and New York: Addison Wesley Longman; 2nd ed., 1999.

Fraser, Charles, and John Milne. *Integrated Electrical and Electronic Engineering for Mechanical Engineers.* London and New York: McGraw-Hill, 1994.

Histand, Michael, and David Alciatore. *Introduction to Mechatronics and Measurement Systems.* Boston: WCB/McGraw-Hill, 1999.

—*Chris Woodford*

Memory Hierarchy

A computer stores data in different types of media: fast static memory chips, slow dynamic memory chips, **hard disk**, or tape. Usually the fastest devices are also the most expensive. Therefore, optimizing the cost of a computer system means that the memory has to be organized hierarchically: Starting with the processor, the devices are connected in a chain going from the fastest to the slowest. The fastest devices are smaller, since they are so expensive; the slower devices have a larger storage capacity. This is the memory hierarchy and moving information across the different levels is called *memory hierarchy management.*

The fastest "data containers" available in a computer are the registers. Data can be stored and retrieved from them in a few nanoseconds. Current **microprocessor** generations have cycle times of 2 nanoseconds (ns) (at a 500 megahertz [MHz] clock rate), and ideally, a piece of data should be retrievable in each cycle. Registers are constructed of the fastest logic elements available, and a microprocessor can have just a few or several hundred.

The next element in the memory hierarchy is the data **cache**, which can be divided in up to three levels, and then we have the main memory. Usually, dynamic random access memory (**RAM**) chips are used for main memory, which can be as large as hundreds of megabytes. The cache is made of fast and expensive chips. Data present in the main memory are replicated in the cache so that the processor can access it from the faster medium, but since the cache is smaller than main memory, special circuits must decide which data must be kept in the cache and which can be overwritten.

When the processor needs data from a memory address, it looks in the first-level (L1) cache. If it is there, the data are transferred to a register. If not, the electronic looks in the second-level (L2) cache. If the data are found there, they are transferred to the first-level cache, then to the processor. Additional cache levels only extend this strategy using more access levels. When the data are not present in any of the cache levels, they are read from main memory.

The figure provides a more detailed view of a possible memory hierarchy organization. The processor accesses only the first-level cache, which is divided into two units: The instruction cache contains only instructions; the data cache, as the name implies, contains only data. The purpose of this division is to allow the processor to access one instruction at the same time that a data word can be read or written. The first-level cache is not connected directly to the L2 cache; there are three buffers between both. The first *miss buffer* connected to the L1 instruction cache holds any data coming from the slower-level caches, until the L1 cache can absorb it. Any speed mismatches between the cache units are absorbed in this way by the stream buffer. The two other stream buffers between the L1 and L2 data cache are used to read and write data, respectively, from and to the lower-level cache. The buffers between the third-level and second-level caches have the same purpose.

The next level in the memory hierarchy is main memory itself, and then the hard disk. Almost all modern operating systems use **virtual memory**. This means that although the computer may have only 64 megabytes (MB) of memory, the **operating system** simulates that it has more by using part of the hard disk to extend main memory. To avoid having to read single words from the hard disk (which would be too slow), entire pages are transferred between main memory and the hard disk. Called *swapping*, this extends the available memory for programs.

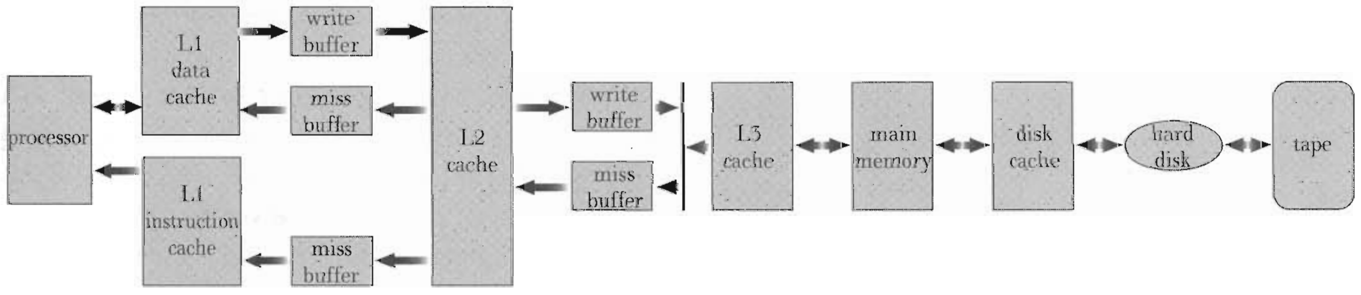
However, between the **bus** and the hard disk, we could also have a *disk cache*. A disk cache is composed of memory chips, which hold the latest sectors read from the hard disk. Just as in the case of the L1, L2, L3 caches, the purpose of the disk cache is to avoid having to retrieve often-used data from the slower unit each time these data are needed. Many hard disk controllers for PCs include a hard disk cache on the same expansion card.

The last level in the memory hierarchy depicted in the diagram is constituted by magnetic tapes stored in a special room and handled by a robotic arm. Data that have not been accessed in the last year, for example, can be copied from hard disk to tapes. The tape device has hundreds of times the storage capacity of the hard disks, but accessing the information can take several minutes.

The "distance" from the hierarchy of different elements to the processor is measured according to the number of cycles that it takes to bring a piece of data to a register, the *data retrieval latency*. According to this definition, the distance of the various elements of the memory hierarchy could be the following: L1 cache, 2 cycles; L2 cache, 5 cycles; L3 cache, 8 cycles; main memory, 30 cycles; disk cache, 1000 cycles; hard disk, 1 million cycles; tape, 300 billion cycles (10-minute access time at 500 MHz).

The size of each element of the memory hierarchy is lower when the distance to the processor is smaller. The L1 and L2 caches could hold several kilobytes, the L3 cache several megabytes. Main memory and a disk cache are measured in hundreds of megabytes, and the hard disk in gigabytes. The tape device can hold terabytes of data.

Management of the memory hierarchy is based on the *principle of locality*. This means that when programs access data from a specific address, it is highly probable that neighboring addresses will also be accessed to read or store numbers (spatial locality). It is also likely that a piece of data that has been read will be reused afterward (temporal locality). Finally, programs are written as sequences of instructions and data are stored in arrays, so that if an address is read, it is probable that the addresses immediately following will



Memory hierarchy.

also be needed (sequential locality). Therefore, whenever an address is read from main memory into a cache, an entire block (i.e., several consecutive addresses) are loaded at the same time. When these extra addresses are needed later, they will already be loaded in the cache. The same happens when a piece of data is read from the hard disk: One or more sectors are brought into the hard-disk cache, anticipating that the additional data will be needed later.

Although all computers built since 1945 had several levels of faster or slower storage, the first person to suggest the automatic management of caches (i.e., the loading of data into the faster devices without explicit programmer's intervention) was **Maurice Wilkes** (1913–) at Cambridge University, who gave credit for the idea to Gordon Scarott. The Titan, an experimental computer built from 1961 to 1964, was the first to implement *slave memory*, as Wilkes called caches. The **Atlas**, built by **Tom Kilburn** (1921–2001) in Manchester in 1962, was the first to provide virtual memory (i.e., the automatic mapping of memory pages to a magnetic drum). The **IBM** 360/86, introduced in 1967, is credited with having been the first commercial computer to include a cache. A modern microprocessor such as the **Pentium** III **from Intel** provides two levels of cache on-chip with an L2 cache of 256 MB.

The largest single problem that computer architects will have to face in the near future is the widening gap between the clock rate of the processor and the cycle time of memory elements. There are already commercial microprocessors running at 1 gigahertz, that is, 1 ns of cycle time, whereas commercial memory chips have access times between 30 and 60 ns. Also, since the **transistor** budget (the number of transistors that can be accommodated on a single chip) has increased and is expected to reach several hundred million units, the question arises if it is better to integrate main memory into the same chip as the processor. This has been called *intelligent RAM* and is one of the options being investigated to solve the processor–memory bottleneck.

FURTHER READING

Intel. *Pentium III Processor Data Sheet.* San Jose, Calif.: Intel, 2000.

McNutt, Bruce. *The Fractal Structure of Data Reference: Applications to the Memory Hierarchy.* Boston: Kluwer Academic, 2000.

Przybylski, Steven. *Cache and Memory Hierarchy Design: A Performance-Directed Approach.* San Mateo, Calif.: Morgan Kaufmann, 1990.

Wilkes, M.V. "Slave Memories and Dynamic Storage Allocation." *IEEE Transactions on Electronic Computers*, Vol. EC-14, No. 2, Apr. 1965.

—*Raúl Rojas*

Metcalfe, Robert

1946–

U.S. Electrical Engineer and Journalist

As the inventor of **Ethernet**, the world's most popular method of connecting computers to one another and to the **Internet**, Robert Metcalfe has been one of the most important figures in the computer revolution of the late twentieth century. He founded the highly successful **3Com Corporation** in the 1970s to take advantage of his invention and in the 1990s became a noted computer journalist and technology pundit.

Bob Metcalfe traces his interest in computers to an eighth-grade science project in 1959, when he constructed a primitive electronic calculator out of his model railroad controller. His other formative computer experience involved programming an **IBM mainframe** in Saturday morning science classes for high school students hosted by Columbia University. But the pattern of his life was really sealed by an earlier episode at school. Faced with an imminent deadline to write a book review, the 9-year-old Metcalfe picked out one of his father's electronics textbooks almost at random and concluded his review with a prophetic sentence indicating that he planned to go to Massachusetts Institute of Technology (MIT) to study electrical engineering.

Almost a decade later, as he had promised, Metcalfe began his studies at MIT. He was soon exposed to a variety of powerful **mainframe** computers, including the college's IBM 7094 (IBM's first transistorized mainframe) and a military-grade **Univac** that he used for an after-hours' programming job. Just as Metcalfe had benefited from Saturday morning programming classes at Columbia during his high school years, so he found