by Jeff Tyson

When you think about it, it's amazing how many different types of electronic memory you encounter in daily life. Many of them have become an integral part of our vocabulary:

- RAM
- ROM
- Cache
- Dynamic RAM
- Static RAM
- Flash memory
- Memory Sticks
- Virtual memory
- Video memory
- BIOS

You already know that the computer in front of you has memory. What you may not know is that most of the electronic items you use every day have some form of memory also. Here are just a few examples of the many items that use memory:

- Cell phones
- PDAs
- Game consoles
- Car radios
- VCRs
- TVs

Each of these devices uses different types of memory in different ways!

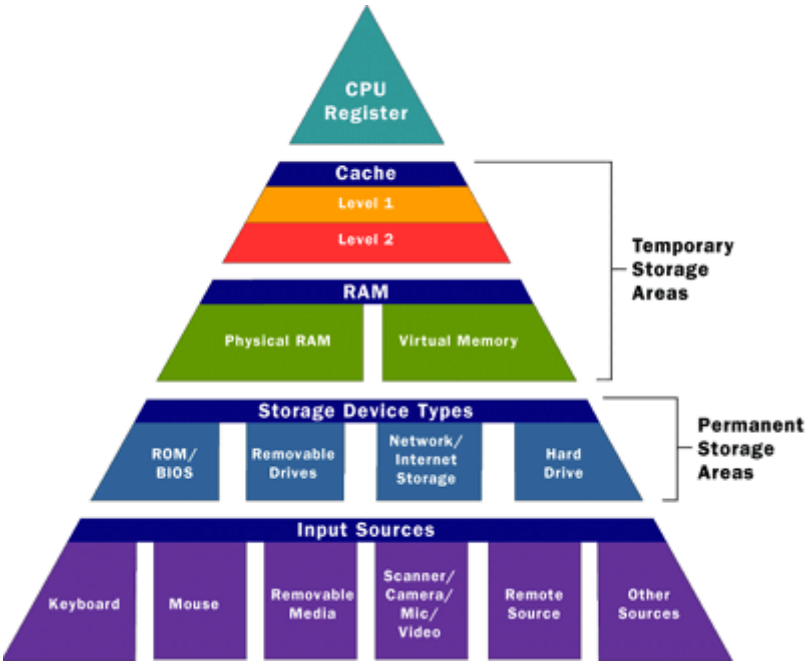
In this article, you'll learn why there are so many different types of memory and what all of the terms mean.

Computer Memory!

- **How Computer Memory Works**
- How BIOS Works
- How Caching Works
- How Flash Memory Works
- How RAM Works
- How Removable Storage Works
- How ROM Works
- How Virtual Memory Works

Memory Basics

Although memory is technically any form of electronic storage, it is used most often to identify fast, temporary forms of storage. If your computer's CPU had to constantly access the hard drive to retrieve every piece of data it needs, it would operate very slowly. When the information is kept in memory, the CPU can access it much more quickly. Most forms of memory are intended to store data temporarily.



As you can see in the diagram above, the CPU accesses memory according to a distinct hierarchy. Whether it comes from permanent storage (the hard drive) or input (the keyboard), most data goes in **random access memory** (RAM) first. The CPU then stores pieces of data it will need to access, often in a **cache**, and maintains certain special instructions in the **register**. We'll talk about cache and registers later.

All of the components in your computer, such as the CPU, the hard drive and the operating system, work together as a team, and memory is one of the most essential parts of this team. From the moment you turn your computer on until the time you shut it down, your CPU is constantly using memory. Let's take a look at a typical scenario:

- You turn the computer on.
- The computer loads data from **read-only memory** (ROM) and performs a **power-on self-test** (POST) to make sure all the major components are functioning properly. As part of this test, the **memory controller** checks all of the memory addresses with a quick **read/write** operation to ensure that there are no errors in the memory chips. Read/write means that data is written to a bit and then read from that bit.
- The computer loads the **basic input/output system** (BIOS) from ROM. The BIOS provides the most basic information about storage devices, boot sequence, security, **Plug and Play** (auto device recognition) capability and a few other items.
- The computer loads the **operating system** (OS) from the hard drive into the system's RAM. Generally, the critical parts of the operating system are maintained in RAM as long as the computer is on. This allows the CPU to have immediate access to the operating system, which enhances the performance and functionality of the overall system.
- When you open an **application**, it is loaded into RAM. To conserve RAM usage, many applications load only the essential parts of the program initially and then load other

pieces as needed.
- After an application is loaded, any **files** that are opened for use in that application are loaded into RAM.
- When you **save** a file and **close** the application, the file is written to the specified storage device, and then it and the application are purged from RAM.

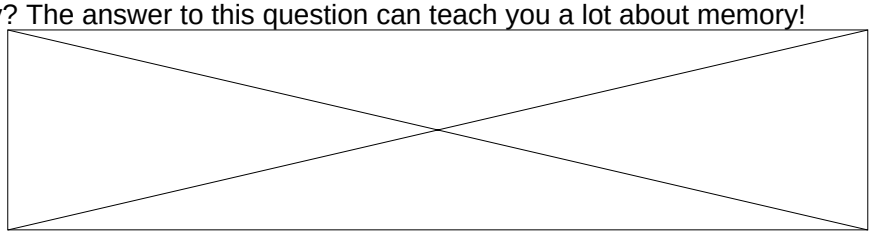
In the list above, every time something is loaded or opened, it is placed into RAM. This simply means that it has been put in the computer's **temporary storage area** so that the CPU can access that information more easily. The CPU requests the data it needs from RAM, processes it and writes new data back to RAM in a **continuous cycle**. In most computers, this shuffling of data between the CPU and RAM happens millions of times every second. When an application is closed, it and any accompanying files are usually **purged** (deleted) from RAM to make room for new data. If the changed files are not saved to a permanent storage device before being purged, they are lost.

The Need for Speed

One common question about desktop computers that comes up all the time is, "Why does a computer need so many memory systems?" A typical computer has:

- **Level 1 and level 2 caches**
- **Normal system RAM**
- **Virtual memory**
- **A hard disk**

Why so many? The answer to this question can teach you a lot about memory!



Fast, powerful CPUs need quick and easy access to large amounts of data in order to maximize their performance. If the CPU cannot get to the data it needs, it literally stops and waits for it. Modern CPUs running at speeds of about **1 gigahertz** can consume massive amounts of data -- potentially billions of bytes per second. The problem that computer designers face is that memory that can keep up with a 1-gigahertz CPU is extremely **expensive** -- much more expensive than anyone can afford in large quantities.

Computer designers have solved the cost problem by "**tiering**" memory -- using expensive memory in small quantities and then backing it up with larger quantities of less expensive memory.

The cheapest form of read/write memory in wide use today is the **hard disk**. Hard disks provide large quantities of inexpensive, permanent storage. You can buy hard disk space for pennies per megabyte, but it can take a good bit of time (approaching a second) to read a megabyte off a hard disk. Because storage space on a hard disk is so cheap and plentiful, it forms the final stage of a CPUs memory hierarchy, called **virtual memory**.

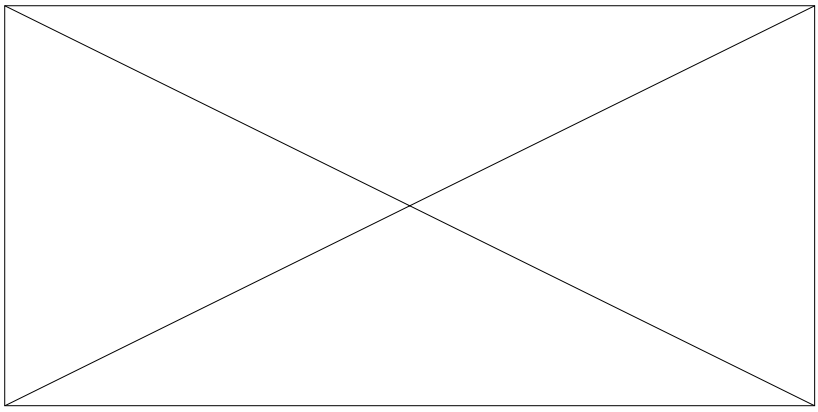
The next level of the hierarchy is **RAM**. We discuss RAM in detail in How RAM Works, but several points about RAM are important here.

The **bit size** of a CPU tells you how many bytes of information it can access from RAM at the same time. For example, a 16-bit CPU can process 2 bytes at a time (1 byte = 8 bits, so 16 bits = 2 bytes), and a 64-bit CPU can process 8 bytes at a time.

Megahertz (MHz) is a measure of a CPU's processing speed, or **clock cycle**, in millions per second. So, a 32-bit 800-MHz Pentium III can potentially process 4 bytes simultaneously, 800 million times per second (possibly more based on pipelining)! The goal of the memory system is to meet those requirements.

A computer's system RAM alone is not fast enough to match the speed of the CPU. That is why you need a **cache** (see the next section). However, the faster RAM is, the better. Most chips today operate with a cycle rate of 50 to 70 nanoseconds. The read/write speed is typically a function of the type of RAM used, such as DRAM, SDRAM, RAMBUS. We will talk about these various types of memory later.

System RAM speed is controlled by **bus width** and **bus speed**. Bus width refers to the number of bits that can be sent to the CPU simultaneously, and bus speed refers to the number of times a group of bits can be sent each second. A **bus cycle** occurs every time data travels from memory to the CPU. For example, a 100-MHz 32-bit bus is theoretically capable of sending 4 bytes (32 bits divided by 8 = 4 bytes) of data to the CPU 100 million times per second, while a 66-MHz 16-bit bus can send 2 bytes of data 66 million times per second. If you do the math, you'll find that simply changing the bus width from 16 bits to 32 bits and the speed from 66 MHz to 100 MHz in our example allows for three times as much data (400 million bytes versus 132 million bytes) to pass through to the CPU every second.



In reality, RAM doesn't usually operate at optimum speed. **Latency** changes the equation radically. Latency refers to the number of clock cycles needed to read a bit of information. For example, RAM rated at 100 MHz is capable of sending a bit in 0.00000001 seconds, but may take 0.00000005 seconds to start the read process for the first bit. To compensate for latency, CPUs uses a special technique called **burst mode**.

Burst mode depends on the expectation that data requested by the CPU will be stored in **sequential memory cells**. The memory controller anticipates that whatever the CPU is working on will continue to come from this same series of memory addresses, so it reads several consecutive bits of data together. This means that only the first bit is subject to the full effect of latency; reading successive bits takes significantly less time. The **rated burst mode** of memory is normally expressed as four numbers separated by dashes. The first number tells you the number of clock cycles needed to begin a read operation; the second, third and fourth numbers tell you how many cycles are needed to read each consecutive bit in the row, also known as the **wordline**. For example: 5-1-1-1 tells you that it takes five cycles to read the first bit and one cycle for each bit after that. Obviously, the lower these numbers are, the better the performance of the memory.

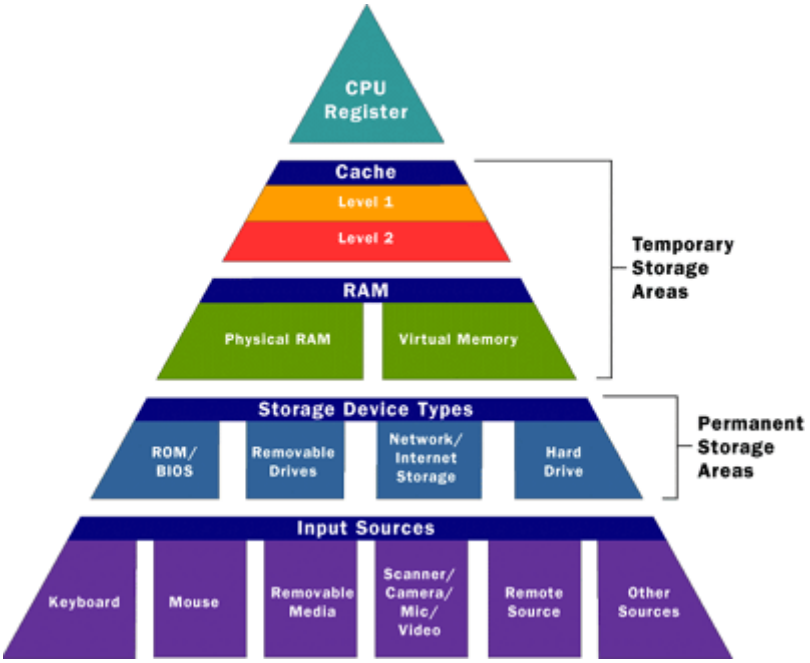
Burst mode is often used in conjunction with **pipelining**, another means of minimizing the effects of latency. Pipelining organizes data retrieval into a sort of assembly-line process. The memory controller simultaneously reads one or more words from memory, sends the current word or words to the CPU and writes one or more words to memory cells. Used together, burst mode and

pipelining can dramatically reduce the lag caused by latency.

So why wouldn't you buy the fastest, widest memory you can get? The speed and width of the memory's bus should match the system's bus. You can use memory designed to work at 100 MHz in a 66-MHz system, but it will run at the 66-MHz speed of the bus so there is no advantage, and 32-bit memory won't fit on a 16-bit bus.

Cache and Registers

Even with a wide and fast bus, it still takes longer for data to get from the memory card to the CPU than it takes for the CPU to actually process the data. **Caches** are designed to alleviate this bottleneck by making the data used most often by the CPU instantly available. This is accomplished by building a small amount of memory, known as **primary** or **level 1** cache, right into the CPU. Level 1 cache is very small, normally ranging between 2 kilobytes (KB) and 64 KB.



The **secondary** or **level 2** cache typically resides on a memory card located near the CPU. The level 2 cache has a direct connection to the CPU. A dedicated integrated circuit on the motherboard, the **L2 controller**, regulates the use of the level 2 cache by the CPU. Depending on the CPU, the size of the level 2 cache ranges from 256 KB to 2 megabytes (MB). In most systems, data needed by the CPU is accessed from the cache approximately 95 percent of the time, greatly reducing the overhead needed when the CPU has to wait for data from the main memory.

Some inexpensive systems dispense with the level 2 cache altogether. Many high performance CPUs now have the level 2 cache actually built into the CPU chip itself. Therefore, the size of the level 2 cache and whether it is **onboard** (on the CPU) is a major determining factor in the performance of a CPU. For more details on caching, see How Caching Works.

A particular type of RAM, **static random access memory** (SRAM), is used primarily for cache. SRAM uses multiple transistors, typically four to six, for each memory cell. It has an external gate array known as a **bistable multivibrator** that switches, or flip-flops, between two states. This means that it does not have to be continually refreshed like DRAM. Each cell will maintain its data as long as it has power. Without the need for constant refreshing, SRAM can operate extremely quickly. But the complexity of each cell make it prohibitively expensive for use as standard RAM.

The SRAM in the cache can be **asynchronous** or **synchronous**. Synchronous SRAM is designed to exactly match the speed of the CPU, while asynchronous is not. That little bit of timing makes a difference in performance. Matching the CPU's clock speed is a good thing, so always look for synchronized SRAM. (For more information on the various types of RAM, see How RAM Works.)

The final step in memory is the **registers**. These are memory cells built right into the CPU that contain specific data needed by the CPU, particularly the **arithmetic and logic unit** (ALU). An integral part of the CPU itself, they are controlled directly by the compiler that sends information for the CPU to process. See How Microprocessors Work for details on registers.

Types of Memory

Memory can be split into two main categories: **volatile** and **nonvolatile**. Volatile memory loses any data as soon as the system is turned off; it requires constant power to remain viable. Most types of RAM fall into this category.

Nonvolatile memory does not lose its data when the system or device is turned off. A number of types of memory fall into this category. The most familiar is ROM, but **Flash memory** storage devices such as CompactFlash or SmartMedia cards are also forms of nonvolatile memory. See the links below for information on these types of memory.

For a handy printable guide to computer memory, you can print the HowStuffWorks Big List of Computer Memory Terms.