Understanding Memory

TYPES OF MEMORY

In this study unit, you'll learn about physical and logical memory. *Physical memory* comes in two types, *random access memory (RAM)* and *read-only memory (ROM).* Typically, the term "memory" refers to RAM, or a temporary workspace for data used by hardware devices, applications, and the CPU. "Random access" means that the CPU can access any address in memory in the same amount of time as any other.

Keep in mind, though, that there's a difference between *memory* and *storage.* Memory values, for the most part, are temporary, being maintained and changed by electronic means. "Storage" refers to permanent storage such as on a hard drive or removable storage. Storage devices usually use magnetism to store values. When, for example, an accounting program is started, or *loaded,* the main pieces of the program and data locations are transferred into memory. As the CPU requires more of the program or associated data, this information is swapped in and out of memory.

Anytime the CPU needs to perform a task, it quickly accesses the addresses of the data it requires from memory, *not* the hard drive. A CPU may access its memory 50,000 times faster, or more, than it does its hard drive. To illustrate further: computers operate in *nanoseconds,* which are billionths of a second. If it takes your CPU one second to read memory, it takes about 14 hours to read the hard drive. Now let's look at memory chip types, their *packages* (that is, how they plug into the computer), their capacities, and their overall advantages.

Read-Only Memory (ROM)

ROM is an integrated circuit that's programmed with specific microcode. Considered a *nonvolatile,* or permanent, type of memory, ROM retains its contents when power to the computer is turned off. The IC chip programmed with its microcode is referred to as *firmware* because the chip contents are "firm," or rarely changed, and one needs special software or devices to update them. (Conversely, the contents of RAM are referred to as *software,* because they change easily and often.) Although ROM by general definition is "read only," that is, programmed only once, reprogrammable types do exist. You'll learn about these shortly.

ROM is used on system boards, video cards, modems, and other components to store data, instructions, and programs that must be retained when power is removed from the personal computer. For example, the *basic input/output system* (or *BIOS*) used to boot your computer is stored in ROM.

BIOS, which is stored within CMOS (a type of microchip fundamental to most CPUs), contains instructions for executing the power-on self-test (POST) instructions, loading drivers for installed devices or utilities for keyboard and monitor, activating other ROM (such as video graphics cards), and passing system control to the operating system. Once the OS is running, BIOS provides OS and application access to the installed devices.

Sometimes BIOS needs to be updated to accommodate a new device or one with larger capacity. Hard drives have increased in storage capacity, but access to the total drive space has been limited by the BIOS. There have been barriers at 528 megabytes (MB), 2.1 gigabyte (GB), 8.4 GB, and 137 GB. One solution to this problem is to use software or utilities to translate absolute hard drive addresses into logical locations. A second method is adding an adapter card with newer BIOS and capacity. The third solution is upgrading the present BIOS to be able to recognize and read more sectors on the hard drive, either by

replacing the ROM chip itself or reprogramming it. Sometimes this isn't possible because of older ROM or limitations of the chipset, in which case the whole system board needs replacing.

Now we'll look at the different characteristics of five basic types of ROM:

- ROM
- PROM (programmable read-only memory)
- EPROM (erasable programmable read-only memory)
- EEPROM (electrically erasable programmable read-only memory)
- Flash memory

Because ROM chips are read-only, the microcode programming must be totally correct and totally complete. You typically can't add bits of new code or make corrections to existing code after the ROM has been burned. Creating a workable ROM chip template is a difficult process, often based on trial and error. ROM, by definition, isn't changeable once it's programmed. If data is incorrect or new devices need to be added or updated, the ROM needs to be replaced on the system board.

But the advantages of ROM outweigh the difficulties. Once the template is tested and complete, the cost to duplicate the ROM is relatively inexpensive. ROMs are reliable, have low power requirements, and generally contain all the code necessary to run associated devices.

Now we'll look at each of these types of ROM in detail.

ROM

As we've mentioned, the information held in ROM isn't lost when power is turned off. This type of memory is called *non-volatile memory.* The manufacturer *programs* the ROM by burning it into the chip. The way a ROM chip works requires the programming of correct and complete data when the chip is created. Though it can be tricky to create an original ROM template, the process is usually worth it. With a finished template,

the chips themselves are usually very inexpensive. They don't use a lot of power, usually contain all the programming necessary to control a device, and can be depended on to perform consistently. Of course, you can't reprogram or rewrite a standard ROM chip. If it contains bad data or needs to be updated, you have to scrap the chip and create a new one.

Programmable ROM (PROM)

ROM programming limitations, the expense of creating original templates, and the cost of generating small ROM batches led to the development of reprogrammable ROM. The first type is *programmable read-only memory,* or *PROM.* PROM chips are inexpensive and formatted with a device called a *programmer.*

Like regular ROM, PROM memory addresses are arranged in rows and columns. However, a fuse is located at each address. An address where the fuse is melted is classified as *0* (signifying "off"), and an address where the fuse is *whole,* or unmelted, is classified as a *1* (signifying "on"). Prior to programming, all fuses are whole and all addresses are set as 1s. The PROM programmer tool can selectively melt fuse addresses, setting up patterns of 1s and 0s that translate into program microcode.

Because PROM consists of an array of fuses, it's more fragile than regular ROM and thus susceptible to corruption by static electricity (blown fuse). Also, to program PROM, it must be removed from the system board. Care must be taken not to damage PROM pins or stretch the holes on the system board. Inexpensive and programmable as it is, PROM is sometimes used in the trial-and-error generation of the permanent ROM template. Early disadvantages of limited reuse and chip-handling problems pushed developers toward a better, more versatile PROM.

Erasable Programmable ROM (EPROM)

EPROM is similar to PROM, except that it can be erased and reused. Software occasionally needs to change, so it became important to be able to update the ROM. To update ROM in older computers, you needed to replace the chips. Newer systems are upgraded by simply running a program. EPROMs have a small window on the top of the chip that can be uncovered and exposed to a specific frequency of high-intensity ultraviolet light. Exposure of about 15 minutes erases the EPROM (chemically melting the fuses back together and setting them all to 1s) and allows it to be rewritten with new data. The EPROM must be completely erased before you can write new code to the chip. Of course, special equipment is needed for these operations. Nevertheless, EPROM is cheaper to make and burn than PROM. If a mistake is made, EPROM chips can be erased and reused. Though EPROMs are a major improvement over PROMs, they still require dedicated equipment and a labor-intensive process to remove and reinstall them each time a change is needed. In addition, changes can't be made incrementally to an EPROM; the entire chip must be erased.

Electronically Erasable Programmable ROM (EEPROM)

Two of the drawbacks to EPROMs are (1) you still need dedicated equipment to modify the code on the chip, and (2) the chip must be physically removed from the system board. The chip is mounted in through-hole sockets on the system board. Repeatedly removing the chip contributes to *chip creep,* in which the chip gradually begins to work its way out. This can cause parity errors or ROM reliability problems. However, the development of EEPROM has bypassed these setbacks. Instead of using ultraviolet light, EEPROM applies a higher electrical charge, erasing selective bytes and allowing new data to be written. This is done with software "on

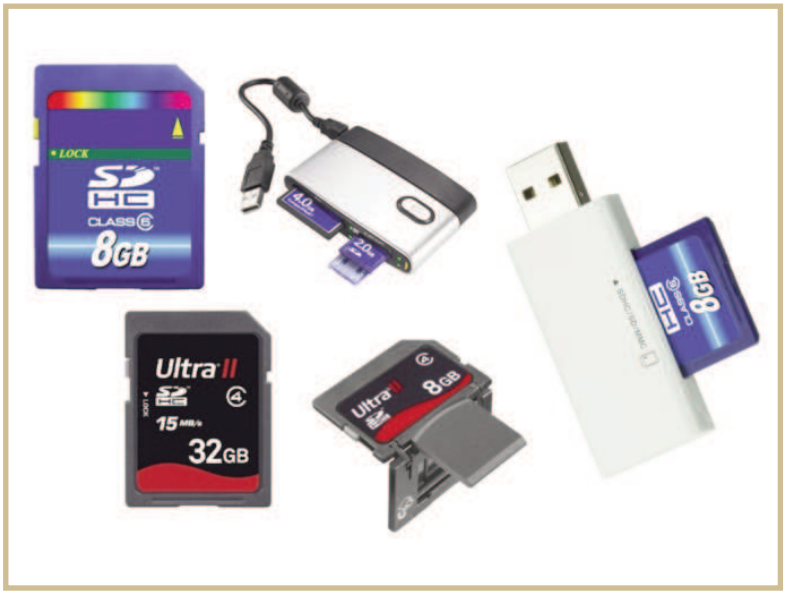
the fly," meaning that you don't have to remove the chip from the system board, and the computer system is running as the chip is being flashed.

BIOS chips are now typically EEPROMS. While versatile, EEPROMS are modified only one byte at a time, a rate that can be too slow for devices requiring rapid and repeated code updates. Flash memory was the next evolutionary step that helped solve this speed limitation.

Flash Memory

Flash memory is many times faster than EEPROM, because it can access a block of addresses—the entire chip—instead of a single byte. System board BIOS chips made since 1994 are EEPROMS or flash EEPROMS. Flash memory is also packaged as removable external PCMCIA cards for laptops, digital cameras, and game consoles (Figure 1). Because of their memory capacities (from 2 GB to 286 GB; their *nonvolatility,* or ability to maintain data for decades; and their easy transportability, flash memory cards are more like storage than memory. But unlike storage devices with moving parts, flash is *solid-state,* or totally electronic, and this adds to its durability and speed.

FIGURE 1—Flash Memory Cards



Random Access Memory (RAM)

The random access memory, or RAM, of a computer can be compared to the surface of your desk. When you're working, you have papers and books spread out and you can refer to each of them. The larger your desk, the more things you can spread out on it. In addition, while you're working, you don't have to start at one corner of the desk and check each item until you come to the book you want. Instead, you can go directly to that book wherever it's located on the desktop. When you're finished with a book, you can put it away and free up space on the desk. When you need another book for your work, you can place it on the desktop for easy reference as long as there's space left for it. When you're done with the project, you remove all of the books and papers from the desk and store them on shelves or in files.

In a computer system, running applications store information about themselves, addresses, requested services, and data locations in RAM. The central processing unit (CPU) can access a particular memory location in RAM and retrieve the data that's stored in that location. The amount of RAM can be compared to the size of the desktop. Like the desktop, RAM has a set amount of space that can be filled with data. The data stored in RAM is there only temporarily. When the computer is turned off, the contents of RAM are erased, which is a lot like clearing off the desk. If you haven't saved the data, it will be lost when the computer loses power. This would be similar to being in the middle of a project and having someone come into your office and permanently remove all of the papers and books from the desktop even though you still might need them. Because the contents of RAM are erased when you turn off the computer, RAM is considered to be *volatile* memory, that is, the data will disappear when power is turned off. Hard-drive storage is considered *nonvolatile* memory. Data saved to the hard drive remains there until it's purposely deleted.

What's the Difference between System Memory and Storage Capacity?

Some people are confused about the difference between the terms *system memory* (meaning RAM) and *storage capacity* (that is, hard-drive space). RAM should never be confused with storage capacity. When you think of storage, think of the hard drive. The hard drive is like a file cabinet. Using the desk analogy, when you're through working with the papers, you file them safely away in the filing cabinet. Then you can refer to them later by just pulling them out of the cabinet. The computer equivalent of this is storing data to a file that's placed on the hard drive. Since the contents of the hard drive don't disappear when the power is turned off, the file can be retrieved from the hard drive later and loaded into memory so the computer can work with it again (Figure 2). The file that you store on the hard drive might be a letter or sections of a document that's in process.

FIGURE 2—Retrieving data on the hard drive is similar to retrieving paper items from a filing cabinet.

System memory is generally measured in units called *megabytes (MB)* or *gigabytes (GB)*. A typical system might come with 2 GB, or 2,000,000,000 bytes of RAM. Hard-drive capacity, on the other hand, is much larger and is usually measured in *terabytes (TB)* rather than in megabytes or gigabytes.

Remember that prefixes before the word *byte* denote multiple amounts of bytes. Also, remember that there are eight binary digits, or bits, per byte. A *kilobyte* (abbreviated *K* or *KB*) equals one thousand bytes, a *megabyte* (*meg* or *MB*) equals one million bytes, a *gigabyte* (*gig* or *GB*) equals one billion bytes and a *terabyte (TB)* equals one trillion bytes or a thousand gigabytes. The motherboard chipset, the memory controller, and the width of the address bus will limit the amount of RAM accessible by the system. There are methods for getting around these limitations—upgrading ROM, utilizing special software, or logical addressing, but upper limits still exist. The 32-bit desktop versions of Microsoft Windows operating systems (2000 Professional, XP, NT 4.0) are limited to four gigabytes of physical RAM. Using Intel's Pentium IV processor and 36-bit page extended addressing (PAE-36), Advanced Server and Datacenter Server editions take an extra four bits for addressing and support up to 64 gigabytes of RAM. Current versions of Microsoft Windows operating systems (Windows Vista, Windows 7) are 64-bit, and can physically address up to 2 TB of physical RAM; however, limitations in the software restrict the amount of usable RAM to 192 GB for the Ultimate, Professional and Enterprise versions.For more detailed information, go to the Intel Web site at **http://www.intel.com.**

Professional Tip

If you hear someone say, "My computer has 80 gigabytes of memory," you should immediately suspect that the person is confusing memory with storage capacity. Most likely, the person's system has an 80 GB hard drive, since most desktop computers wouldn't have 80 GB of RAM.

How Computers Communicate

In the English language, we use 26 letters and 10 numeric digits—0 through 9—to communicate. The computer operates, like millions of switches, in the "on" or "off" position, thus communicating with only two digits, *0* (off) and *1* (on). All requests, messages, and commands from devices or applications are translated into a machine language, composed of arrays of ones and zeroes, called *binary* ("two").

Suppose we have the three binary numbers 01001111, 11100110, and 10101010. Let's see how they would be stored in memory. Table 1 shows how they would look.

Our first number is stored in address 1, the second in address 2, and the third in address 3. Each numeral, whether 0 or 1, is a *binary digit or bit*, and each eight-bit number is a *byte.*

Software that controls the storing of the numbers records the address in which each is stored. Then, when the data is needed, the software can access a specific number by recalling the address. When the address is accessed, the binary word is put onto the data bus of the system. This process is called *reading.*

Table 1

EIGHT-BIT ADDRESSING

	D7	D6	D5	D4	D3	D2	D1	D0
Address 1	0	1	0	0	1	1	1	1
Address 2	1	1	1	0	0	1	1	0
Address 3	1	0	1	0	1	0	1	0

Let's assume that you're working with word processing software and preparing a document. Each letter has a binary equivalent that's stored in a memory address. Then you discover you've made a typing error, so you go back to the letter that's incorrect and put in the correct letter. Now the new letter would have to be stored to the memory address previously occupied by the old letter. The binary equivalent of the new letter would be put on the data bus and moved to the memory address. This process is called *writing*.

In an ideal situation, the reading and writing of data would happen instantly, but in reality, that's not the way things work. It takes time to access memory. How much time? That varies, which explains why memory is rated with an access time. *Access time* is the time it takes to read from or write to a memory location. It's the delay from the time the memory chip is given an address to the time when the data is available at the data lines. This time is measured in *nanoseconds (ns).* One nanosecond equals one billionth of a second (that is, 0.000000001 second). That doesn't sound like very much time in human terms, but in a fast computer system, a few nanoseconds can mean a big performance increase or decrease. Memory chips have their access time marked on them. For example, a notation of "70" on a chip refers to a 70 ns access time. A notation of "60" indicates a 60 ns access time. The smaller the number, the less access time and the faster the chip. Most current RAM has access times in the 3.4–5 ms range.

You're probably wondering if you could increase the computer's speed by simply replacing slower chips with faster chips. That's a good thought, but it won't work. The system board design and CPU determine the speed at which the computer can function. When the 80286 processor came out, system designers found that the memory that was in use for computers based on the 8086/8088 processor-based computers wasn't fast enough for the 286 CPU. The 286 could read from and write to memory locations faster than the memory could operate. This situation meant that the CPU would try to output data to memory so fast that it would be sending new data before the last data had been written into a memory address. You can see how that would lead to errors. Attempts to read from memory also caused problems since requests

would be sent faster than data could be made available from memory. To remedy the situation, wait states were designed into the system board. A *wait state* is a time delay before the CPU accesses the memory. This means that the CPU would output data, and then wait for a specified time so the CPU could safely write the data to memory before more data would be output. In the case of reading from memory, the CPU would request data from a specific address, then wait for the predetermined time before making another request. The time delay compensated for the slower memory access times.

Adding wait states slows down the system. However, that's the whole idea. It doesn't make sense to have a fast system that makes many errors and loses data. The addition of wait states allows the system to operate reliably. It can accurately read from or write to memory. Older systems used wait states to solve the problem of a CPU that could operate faster than the memory could function. The disadvantage of the wait state solution was that a faster CPU didn't really give a system much of an advantage. Newer systems use a different approach, involving a cache, to better use faster CPUs.

Types of RAM

Let's review a few things that you should understand by now about RAM. RAM is the term that's used to describe the computer's working memory. RAM has two distinct characteristics. First, RAM is *volatile*. This means that if the chip doesn't have power applied to it, all information stored in the chip will be lost. RAM is ideal for loading and running programs and data needed for the computer to operate. Second, RAM can be *written to* and *read from*, so the information stored at any particular address is subject to change.

Some storage media are sequential, such as magnetic tape. To go to a location on the tape, you much first go through all previous locations. The term *random access memory* refers to the fact that all locations in memory can be accessed equally and in the same amount of time. This term applies to all the different types of RAM.

Dynamic Random Access Memory (DRAM)

Dynamic random access memory, or *DRAM,* is one of the cheapest types of memory to manufacture. It's more dense (contains more memory cells), and consumes less power than static RAM (SRAM). Most DRAM on system boards today are stored on *dual inline memory modules,* or *DIMMs*.

DRAM holds the binary ones and zeros on very small capacitors inside each chip. Each bit (0 or 1) has its own storage area. Imagine this storage area as a bucket with a hole in it. To keep the bucket full, you must keep adding water to compensate for the water that's draining out through the hole. In a similar way, zeros and ones are placed into DRAM electronically, but the data drains out as the electronic charge grows weaker. Because of this drain, DRAM must constantly be recharged or *refreshed* on a regular basis. While DRAM is being refreshed, it can't be read from or written to. Naturally, this slows things down. To compensate for this slowdown, methods were devised to access DRAM.

Bursting, Paging, and Fast Page Mode

As stated previously, DRAM capacitors constantly lose their charge and must be refreshed every 4 nanoseconds (ns) to maintain their contents. While the memory cells are being refreshed, they can't be read or written to. This causes delays with the faster CPU as it waits for the RAM. Memory cells are arranged in arrays of rows and columns.

To read or write to an address, first the row is found, or *strobed,* by way of the row access strobe (RAS). Then the column address is found with the similarly titled *column access strobe,* or CAS. To access another memory location, the process is repeated. The waiting times required to strobe these rows and columns are known as *latency times.*

In the array, there are four bytes to a row—the first byte location and three adjacent bytes. RAM latency times might be represented as 5-3-3-3, meaning it takes five ns to access the first byte address, three ns to access the second adjacent byte, and so on. The smaller the number, the faster the RAM. Much of the development work to increase RAM speed has focused on decreasing these latency times.

Bursting, or *burst mode,* is one such method that works by keeping the address of the row open, or constantly refreshed, and reading across the adjacent three bytes in four-byte bursts, eliminating the row-access time of the adjacent three bytes. *Paging* is a method of dividing memory into pages of 512 to 4096 bytes. Paging circuits allow access to locations on the page with fewer CPU waits. In *fast page mode* (FPM), RAM incorporates the previous two methods to access memory addresses even more quickly, eliminating some of the access and refresh time. (You'll find FPM in older, pre–66 MHz systems.)

Interleaving

Interleaved memory access is faster than paged memory access. When memory is interleaved, odd bytes are in one bank, while even bytes are in another. While one bank is accessed, the other can be refreshed, saving time. By using interleaving, memory access can be almost doubled without needing faster memory chips.

To use interleaved memory, the system must have two full banks of memory chips. For example, a Pentium with a 64-bit-wide data path needs two 64-bit banks to operate interleaved. A 486 with a 32-bit-wide data path would need two 32-bit banks. If you're not sure whether a system is using interleaved memory, consult the system board manual. It may help you to know that interleaving was popular on 32-bit-wide 486 systems, but unpopular on Pentium systems because of the large amount of memory needed (two 64-bit blanks) for operation.

Extended Data Output (EDO)

Extended data output (EDO) is an enhanced version of fast page mode RAM and is sometimes referred to as *hyper page mode RAM.* Normally, DRAM receives an address and outputs the data from that address onto the data bus. The next request can't be accepted until the data has been read from the bus. The EDO memory has a buffer to hold the requested data. After a request comes in, the data is put in the output buffer

while the next request is read. Thus, EDO RAM can operate much faster than regular DRAM. EDO RAM works well with systems that have bus speeds up to 66 MHz (that is, PCs that were manufactured prior to 1998). However, the market for EDO RAM declined due to the emergence of SDRAM, which we'll discuss in the next section.

EDO also uses burst mode to decrease access times. This reduces overall cycle time by five percent, which is a significant amount when dealing with MHz speeds. EDO uses the same technology as FPM and costs roughly the same, but EDO is faster; this is why FPM has become obsolete. Burst EDO, or BEDO, was a short-lived enhancement with even faster burst-speed features. Intel developed BEDO DRAM, but only one system board chipset supported it, and the enhancement was soon replaced by synchronous DRAM technology.

Synchronous Dynamic Random Access Memory (SDRAM)

In normal DRAM, the CPU waits for the RAM capacitors to refresh and then access memory addresses. As explained earlier, because RAM speed falls short of the speed of the CPU, delays (or waits) occur that increase the overall processing time. *Synchronous DRAM,* or *SDRAM,* eliminates these waits by working within the cycle time of the system clock. Because the RAM is synchronized with the clock cycle, the memory controller knows exactly when the requested data will be available for the CPU; thus, the CPU no longer has to wait. System clock pulses are generated by an oscillator clock on the motherboard, and the CPU and RAM are synchronized with this clock rate via controllers. For machines developed before the adoption of DDR memory, which will be explained in a later section, a typical clock rate might be 100 megahertz (MHz), which is 100 million clock ticks, or cycles, per second. PC100 SDRAM supports this bus speed and, as specified by Intel, has an access time of 8 ns. (Remember that a nanosecond is one billionth of a second.)

In the above 100 MHz example, dividing one billion nanoseconds per second by 100 million clock cycles per second yields 10 ns per clock cycle. And yet, to successfully meet all timing parameters, the PC 100 RAM is actually faster than 10 ns—it's 125 MHz, or 8 ns.

Older SDRAM that supported 66 MHz systems could be called PC66 RAM (15 ns). PC133 RAM (7.5 ns) is supported by a 133 MHz bus and chipset. SDRAM uses burst mode and *pipelining,* in which the memory controller simultaneously reads, sends, and writes data, to further increase speed. In our example, the PC100 latency times would be eight clock cycles (8 ns) to read four bytes of data. This is almost twice as fast as older FPM RAM.

Enhanced SDRAM (ESDRAM)

RAM enhancements up to this point have tried to increase RAM speed by decreasing RAS and CAS delays. *Enhanced SDRAM* (from Enhanced Memory Systems) achieves this by incorporating some faster SRAM within the DRAM chip to work as row register cache. This allows faster internal functionality of the memory module. The SRAM also has a wide bus directly connecting it to the DRAM, which increases bandwidth. This architecture can run at 166 MHz or 6 ns (latency times 2-1-1-1). Typically, ESDRAM is used for L2 cache and competes with DDR SDRAM as a replacement for Socket 7 processor SDRAM.

Double-Data-Rate Synchronous DRAM (DDR SDRAM)

Double-data-rate synchronous DRAM (DDR SDRAM), sometimes called SDRAM II, is a next-generation SDRAM technology. It allows the memory chip to perform transactions on both the rising and falling edges of the clock cycle. For example, with DDR SDRAM, a 100 or 133 MHz memory bus clock rate yields an effective data rate of 200 MHz or 266 MHz.

New naming conventions based on speed are used with DDR SDRAM. Remember that *giga* means "billion," so *1 GB/s* refers to a transfer rate of approximately one billion bytes,

or 1024 megabytes per second. DDR200 equals 1.6 GB/s and is referred to as *PC1600;* DDR333 equals 2.7 GB/s and is referred to as *PC2700.* The speed is roughly derived by multiplying the amount in MHz by 8 bytes (a 64-bit bus can transfer 8 bytes per cycle). Thus, DDR333 is

333 MHz (million cycles/second) × 8 bytes

or approximately 2.7 GB/second. Note that placing PC2700 in a PC1600 system board will yield no more than PC1600 performance-wise.

Since the introduction of DDR memory, there have been two revisions to the memory standard—DDR2 and DDR3, the current memory standard as of 2011 (Figure 3), although DDR and DDR2 memory modules are still in use. These memory types run at much higher frequencies, ranging from 400 MHz up to 2500 MHz for high-end memory. DDR2 and DDR3 memory comes in 240-pin modules, requiring a different socket type than the 184-pin DDR memory, but these aren't compatible with each other. These memory types also use lower voltages; DDR2 modules run at approximately 1.8 V and DDR3 runs at about 1.5 V, as compared to DDR's 2.5 V requirement.

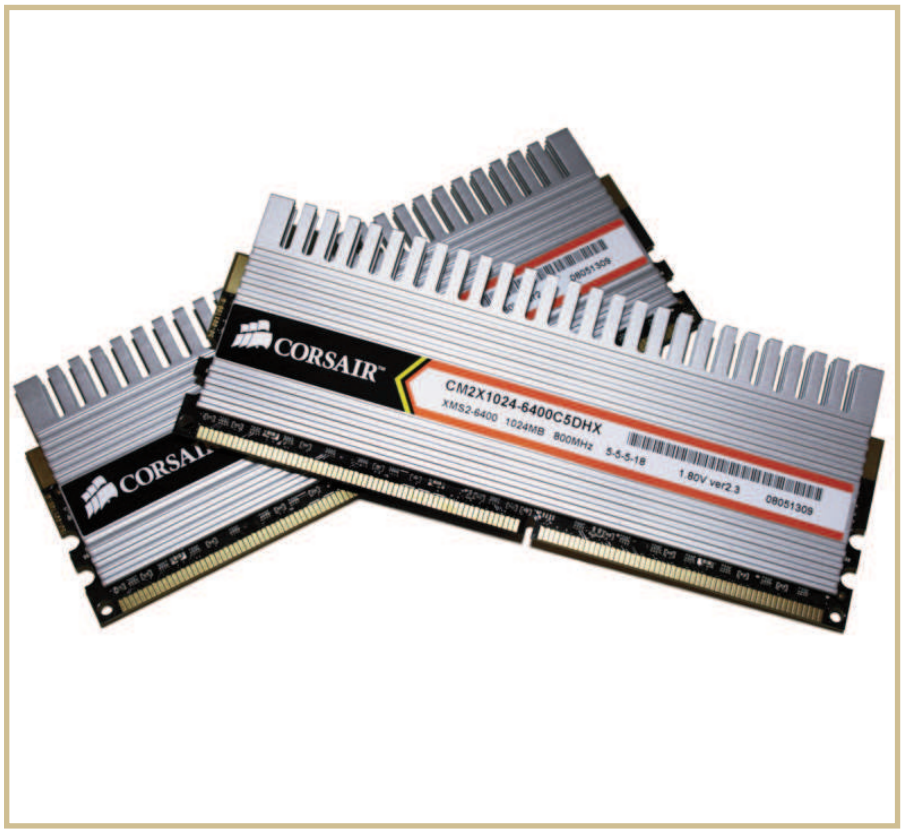


FIGURE 3—DDR3 Memory

Static RAM (SRAM)

The term *static* means "fixed" or "unmoving." *Static RAM,* or SRAM, doesn't have to be refreshed to maintain its contents. SRAM is composed of six transistors at each memory cell. The memory contents are maintained as long as power is applied to the transistors. Unlike DRAM, SRAM doesn't have capacitors that need refreshing; therefore, row/column latency isn't an issue. This drastically lowers access times and makes SRAM much faster than DRAM. Access times for SRAM can be less than 2 ns, while typical access times for DRAM are between 60 and 70 ns. This is why SRAM is used in small amounts for cache and DRAM is used as the main system memory. You may ask, why not use SRAM for all of the system memory if it's so much faster? Unfortunately, SRAM costs more to manufacture than DRAM, consumes more power, and has a higher *density.* This means that 2 MB of SRAM takes up the same physical space as 64 MB of DRAM.

Rambus DRAM (RDRAM)

So far, approaches to increasing system memory performance have included

1. Tweaking RAM access times—decreasing latency times, increasing data flow with interleaving, burst pipelining, and synchronization
2. Adding SRAM caches on DRAM memory modules, on CPUs, and externally on the system board
3. Increasing the width of the buses

The Rambus approach to breaking the barriers of RAM speeds and bus speeds was designing a new architecture. At the time of introduction, the memory bus standard was only 16 bits wide and transferred data at a rate of two bytes per cycle. A single-channel module (168 or 184 pins) had a typical bus speed of 800 MHz, or 1.6 GB/second. Dual-channel 800 MHz RDRAM incorporated two channels on one 242-pin module and ran at 3.2 GB/second. This is comparable to the dual-channel PC3200 DDR RAM platform. However, Rambus memory had some technical deficiencies; RDRAM module, or RIMM, architecture required all memory

slots to be filled, necessitating a continuity RIMM (C-RIMM) be installed if there were more memory slots than modules available. RIMMs were also expensive—two to three times as expensive as regular RAM—ran very hot, and had high latency periods. Due to these technical issues, a number of high-profile intellectual-property disputes with other memory manufacturers and the development of faster memory technologies, RIMMs never became popular in the computing community, and are largely discontinued from use, except in certain specialized applications.

What Is Cache?

A *cache* is a small amount of high-speed memory, the size of which varies. This high-speed memory is known as static RAM (SRAM). Static RAM operates much faster than dynamic RAM (DRAM), which is used for the system's main memory. When the CPU requests information, a *cache controller* that's designed into either the CPU or system board loads chunks of the DRAM into the SRAM. The next time the CPU needs information, the cache is checked first; if the needed data is there, it's loaded into the CPU, speeding up the process. The cache uses a "best guess" method to anticipate what the CPU will need next. It loads chunks of the DRAM into the SRAM while the CPU is busy doing calculations. This operation dramatically increases the speed of the system because the access time of the SRAM is faster than the access time of the DRAM. The CPU doesn't have to wait as long between requests for data.

Cache can be designed into the system board, with SRAM chips installed on the system board. Alternatively, the cache can be inside the CPU, which increases the performance of the system even more.

There are two main types of cache: Level (L1) and Level 2 (L2). A *Level 1 cache* (often called an i*nternal cache* or *L1 cache*) is built into the CPU itself. This significantly increases the speed of the CPU. A *Level 2 cache* is designed outside the CPU on the system board and is often called an e*xternal cache* or *L2 cache*. L2 cache also increases the system's speed, but not always as much as a L1 cache. Many systems have both a Level 1 and a Level 2 cache incorporated into

them. This combination gives an even greater performance boost than either type of cache alone. Although not as common yet, there is a third type of cache, Level 3 (L3) cache, which is the cache memory farthest from the core, but still part of the overall processor system. L3 cache is most prevalent on multi-core processor systems. Cache memory will be discussed in more detail in the next section.

Parity/ECC

Error checking and correcting (ECC) and parity memory modules don't perform any error detection or correction functions themselves. Error checking and correction functions are carried out on the system board, not on the memory module itself. The modules simply provide the space required to store the extra bits of data that represent the condition of the real data. The computers always calculate parity and ECC data for every read and write.

Parity

Parity is an older, low-level, single-bit error detection method. The default parity method was called *odd parity* and will be used as our example.

Recall that memory is an array of capacitors that act like switches, turned on or off. "On" is represented by the binary digit (bit) 1, and "off" is represented by 0. For each byte (8 bits) of a memory word, one extra bit (the 9th) is used to keep track of whether the total number of 1s, or "on" bits, is an odd or even number. If it's odd, then the parity bit stays 0, and if it's even, the parity bit is set to 1. The number of on bits plus the parity bit will always be an odd number. As data is written to memory, the parity generator/checker generates the parity bit. When the data is read from memory, the checker circuit tallies the number of one bits, including the parity bit. If the number is even, the circuit knows that an error has occurred, and a parity error message is triggered. An interrupt also occurs, which halts the system. The system must then be restarted.

Because parity identifies only odd or even quantities, two incorrect bits can go undetected. In addition, there's no way to tell which bit is in error, and bad bits aren't corrected. One positive aspect of parity checking is that it doesn't slow system performance, since it's done in tandem with memory read or write operations.

Parity has lost favor over the years for several reasons:

- Parity can't correct errors.
- Some errors go undetected.
- Errors result in complete system shutdown, which can cause damage to data.
- DRAM has become significantly more reliable.
- Current computer operating systems don't handle parity errors as smoothly as older operating systems.
- For many users, the low level of protection doesn't justify the cost of the extra chips.

ECC

ECC, or *error checking and correcting,* permits error detection as well as correction of certain errors. Typically, ECC can detect single- and dual-bit errors, and can correct single-bit errors. Error detection and correction are done "on the fly" without the operating system even being aware of it. The memory controller chip on the system board performs the correction and always sends corrected data to the CPU. The memory controller can inform the operating system when errors are corrected, but most have no means of logging the corrections or informing the user. So, the user may never know an error ever occurred.

Multibit errors are so rare that further detection and correction capabilities are required only in extreme cases, and would require custom memory components. For that level of protection 100% redundancy would probably be less expensive by using single-bit correction memory modules on two separate memory subsystems. Double-bit detection/

single-bit correction ECC functions may require more or fewer extra bits than parity depending on the data path. For example:

- 8-bit data path requires 5 bits for ECC or 1 bit for parity.
- 16-bit data path requires 6 bits for ECC or 2 bits for parity.
- 32-bit data path requires 7 bits for ECC or 4 bits for parity.
- 64-bit data path requires 8 bits for ECC and 8 bits for parity.
- 128-bit data path requires 9 bits for ECC or 16 bits for parity.

There's a break-even point at 64 bits. At this point, the security of ECC costs the same (in DRAM) as the less capable parity. The efficiency of ECC versus parity in today's 64-bit processors and the inevitably wider data paths of future processors make continued use of parity highly improbable.

Take a few moments now to test your understanding by completing *Self-Check 1.*

Self-Check 1

At the end of each section of *Understanding Memory,* you'll be asked to pause and check your understanding of what you have just read by completing a "Self-Check" exercise. Answering these questions will help you review what you've studied so far. Please complete *Self-Check 1* now.

1. What part of the manufactured system board is likely flash memory?

   _____

2. Which memory is faster, EDO or BEDO?

   _____

3. Explain why static RAM (SRAM) is faster than DRAM.

   _____

   _____

4. Describe the difference between what happens when parity checking discovers an error and when ECC discovers an error.

   _____

   _____

   _____

Check your answers with those on page 31.