Chapter 3: Memory

Terms you’ll need to understand:

- Memory controller
- Read-only memory (ROM) and erasable programmable ROM (EPROM)
- Capacitor
- Bus
- RAM, DRAM, SRAM, SDRAM, VRAM, RDRAM
- DIP, SIMM, DIMM, RIMM

Concepts you’ll need to master:

- Volatile versus nonvolatile memory
- Megahertz (MHz) and nanoseconds (ns.)
- Memory address
- Synchronization to clock cycle (clock tick)
- Memory module versus memory chip
- Odd and even parity checking

Computers use many different microchips and processors, with perhaps the most familiar being the CPU and the main memory—what people commonly refer to as the computer's RAM. Memory is just a temporary place to store information until the CPU can get to it. This information can be program instructions, data, or both. A typical instruction might be a request to store a number or an event somewhere. Another might be to retrieve that information from a particular place—an address. Volatile memory can only hold information when a normal electrical current is present. Nonvolatile memory can hold information without any electrical current.

Caution - Volatile, from the Latin "to fly," means that information "flies away" when there's no electricity to keep it in place. Television reporters often refer to an explosive situation as a volatile situation, meaning that it could change at any second. Nonvolatile memory, because it is not volatile, stays the
same without any need for electricity.

Conceptual Overview

Memory is fairly easy to understand, once you’ve grasped the basic concepts. In a nutshell, a CPU moves bits of data into registers (storage places inside a chip). After it’s dealt with these data bits to its satisfaction, the CPU works together with a memory controller to move the results out to memory cells (storage places on a memory chip). Both registers and memory cells have memory addresses, and every time a bit of data goes somewhere, it crosses a bus of some kind. That’s it! Now go pass the exam.

All right, so it’s a bit, so to speak, more complicated than that. Most memory began as dynamic random access memory (DRAM), and the main changes have been to either speed up the memory to match the CPU, or to speed up the CPU to match the memory. For the most part, the history of memory development revolves around synchronizing these two subsystems.

Note - When we refer to speeding up memory, this usually means increasing either the actual speed of the chips or increasing the clock speed of associated buses.

Memory involves several basic concepts, the first of which is a grid or matrix. Because of this, we’re going to put Table 3.1 to a slightly different use, making it a sort of "mind map." If you can see the way the overall concepts break down on a grid, then perhaps they’ll be easier to remember.

Note - A matrix is nothing more than an arrangement of columns and rows, like a spreadsheet or an Etch-a-Sketch. Columns go across the page, and rows go down the side. Cells going left to right (horizontally) have an X coordinate. Cells going up and down (vertically) have a Y coordinate. The direction of rows or columns is called the axis. Combining both the X and Y coordinates gives us an address in the grid, like a cell address in a spreadsheet.

Table 3.1 Mind map of basic memory concepts.
Read-Only Memory (ROM)

Every computer uses both read-write (RW) memory and read-only (R) memory. Optical disks use a similar designation with CD-RW and CD-R designations. Although the acronym RAM stands for random access memory, for the moment you should think of it as read/write memory. ROM is Read-Only Memory. Information can be temporarily stored in RAM, and then a moment later, it can be taken out and new information can be written to the same place (address). ROM doesn’t allow changes. When information is placed in memory, we say that we are "writing to" a memory address. When information is retrieved out of memory, we "read from" that memory.

Picture a bulletin board under glass at the back of a classroom. One way to think of ROM is that it’s like the hard-copy notes placed under the glass. At the end of the day, they remain unchanged. The next day, the notes are exactly the way they were the day before. We were able to only read them.

If you think of RAM as a blackboard, it starts out blank. During the day, information is written on it, read from it, and maybe even erased. When something is erased, new information is then written to the same place on the board. If a lot of writing and erasing takes place, a chalk buildup forms on the blackboard. This buildup is similar to memory fragments, which can cause computer lockups.
When you go home at the end of the day, you turn off the lights, wash a blackboard clean, and whatever data was on the board goes away forever. This is what happens when you turn off the power to a computer; RAM no longer has the electrical current available to sustain the data in its memory cells.

Caution - ROM can have information written into it only one time. From that point on, we can only read the information. No electrical current is required for the information to remain stored—it's nonvolatile. ROM is mostly used for BIOS, although the same concept and acronym apply to commercial compact disks. These are CD-ROM, with the ROM standing for read-only memory.

In some instances, ROM can be changed through the use of certain tools. Flash ROM is nonvolatile memory that occasionally can be changed, such as when a BIOS chip must be updated.

A single letter can really mess you up on the exam if you don't pay close attention. We've seen questions like, "RAM BIOS is used to permanently store instructions for a hardware device: True or False?" (The answer is false.) Keep your eyes peeled, and remember that RAM sounds like RANdom. RAM is never used in BIOS. Since the BIOS instructions are permanent, they almost always use ROM.

RAM is to a computer, like your attention span (i.e., short-term memory) is to your mind. When you cram for this exam, you'll fill your short-term memory with facts and figures just long enough to write them out to a piece of paper in the exam room. Once the data is on the sheet of paper, you can "forget" the information and concentrate on new data, such as the meaning of an exam question. More realistically, your attention span is like a cache (discussed in a moment), and the piece of paper is like RAM. If you were to engrave the information on the surface of the exam-room desk (this is not allowed), it would be more like ROM.

ROM is more like your long-term memory; the things you remember from your past. When you wake up in the morning, you know your name and address. This is like the information stored in BIOS. On the other hand, you may not remember how many glasses of water you had the day before. That information was stored in your short-term memory.

RAM, short-term memory, or attention span, is like a holding tank for data on its way to becoming information. Data becomes information when it takes on context (surrounding circumstances). 76, on its own, means nothing other than the fact that it's a number. Surround that number with context, "Tomorrow, the temperature is expected to reach 76," and it becomes information.
If someone were to ask you to repeat every word on the last page of the previous chapter, you would have to stop what you were doing; turn the pages of this book until you found the requested page, and then read each word. If you were a computer, you would then pause and wait for a new instruction. A human mind continually extracts meaningful information from words, paying little attention to the words themselves. A computer only manipulates bits of data.

**Basic Input/Output System (BIOS)**

When you turn on a PC, the processor first looks at the basic input/output system (BIOS) to determine the machine’s fundamental configuration and environment. This information is stored in a ROM BIOS chip, and largely determines what peripherals the system can support. BIOS instructions are updated regularly by the manufacturer, and if the chip is made to be updated (re-programmed) by the end-user, it is often called *Flash BIOS*. These programmable chips are often referred to as EEPROM (pronounced *ee-prom*) chips, which we discuss later in this chapter.

---

**Note** - We examine the boot process in depth in Chapter 11, "Booting, Windows 3.x, and Memory."

---

**CMOS Settings**

Although a basic motherboard can be pretty standard, the system can vary in components like hard drives; floppy, CD, or DVD drives; memory; and so forth. The complimentary metal oxide semiconductor (CMOS) is a small memory chip that stores the optional system settings (e.g., hard drive specifications, amount of memory, and so forth). Because these settings are held in CMOS memory with a small electrical charge, CMOS is volatile. However, this trickle charge comes from a battery installed on the motherboard, so even when the main power is turned off, the charge continues. If the battery power fails, all CMOS information vanishes.

---

**Note** - Technically, CMOS is different from ROM BIOS in that the CMOS settings require some source of electrical power. Nonvolatile memory doesn’t require electricity at all.

---

Older computers, such as some IBM PS/2 models and the original IBM AT, required a setup program stored on a special floppy disk. When you ran the program, a setup screen
allowed you to configure the machine. These configurations were stored in special files on the hard drive. Compaq continued the idea of putting a setup program on a disk. Typically, Compaq’s CMOS settings were held in a dedicated 3–4MB, non-DOS partition on the hard drive.

Caution - Today, most computers use a keystroke combination, such as ESC, Del, F1, or F2 to access the CMOS. The keys are pressed at startup, before the BIOS transfers control to the operating system. (This is not the same as accessing Windows Safe Mode.)

The CMOS settings are essential to the hardware configuration of any personal computer. A typical symptom of a fading CMOS battery is that the system date begins to fluctuate, sometimes by months at a time. Backing up files and software are a standard part of keeping a current backup, but you should also have a report of the current CMOS settings. On many modern PCs, this can sometimes be done by turning on a local printer, restarting the machine (as opposed to a first-time boot) and going into the CMOS settings. At each screen, press the Print Screen key.

When you exit out of the CMOS setup, the machine will most likely restart. If you boot to DOS, you can send an end-of-form page request to the printer to print the last page being held in printer memory. This can be done by typing "echo ^L > prn" (without the quotes). The "^L" is actually created by pressing the Ctrl+L key. From within Windows, open a text editor (for example, Notepad) and print a blank page. The stored page in the printer will come out as part of the print job.

Note - If the Print Screen function doesn't load on a particular machine, the only other way to store the CMOS settings is to manually write them down on a piece of paper. We discuss problems with CMOS in Chapter 14 "Troubleshooting."

Flash BIOS

With advances in technology, most BIOS chips became Flash EEPROM (electrically erasable programmable ROM). These new EEPROM chips made it easier to change the BIOS. Rather than pulling out the ROM chip and replacing it with an updated one, upgrades could be downloaded through the Internet or a bulletin board service (BBS). A small installation program changed the BIOS programming, eliminating the need for
pulling apart hardware.

**Caution** - BIOS determines compatibility. Modern BIOS is often stored in the CMOS, whereas older BIOS was stored in nonvolatile ROM, often soldered into the motherboard. Remember that the CMOS is almost always where the computer’s configuration is stored. BIOS is where the software instructions for the basic input/output (I/O) operations are stored (for example, COM and LPT ports, expansion bus, and so forth).

---

**Programmable ROM**

Compact disks (CD-ROMs) offer another common use for one-time, read-only memory. In the same way that rewriteable CDs (CD-RW) changed the way that we use the disks, programmable ROM chips changed the way BIOS was stored. The formal name for a chip that cannot be modified is *mask ROM* (from the manufacturing mask). ROM chips have a varying capacity for change, named in the following manner:

- **Programmable ROM (PROM)** — Requires a special type of machine called a *PROM programmer* or *PROM burner* (like a CD burner) and can only be changed one time. The original chip is blank, and the programmer burns in specific instructions. From that point, it can't be changed.

- **Erasable programmable ROM (EPROM)** — Uses the PROM burner, but can be erased by shining Ultraviolet (UV) light through a window in the top of the chip. Normal room light contains very little UV light.

- **Electrically erasable programmable ROM (EEPROM)** — Can be erased by an electrical charge and then written to by using slightly higher-than-normal voltage. EEPROM can be erased one byte at a time, rather than erasing the entire chip with UV light. Because these chips can be changed without opening a casing, they are often used to store programmable instructions in devices, such as printers and other peripherals.

---

**Flash ROM** - This type of chip is sometimes called *Flash RAM* or *Flash memory* and stores data much like the EEPROM. It uses a super-voltage charge to erase a block of data (rather than a byte). Flash ROM and EEPROM can perform read/write operations, but can only be erased a certain number of times. Flash BIOS relies on this ability to change the program instructions in a ROM chip.
Memory—an Analogy

One of the best illustrations of memory evolution that we’ve seen, was created by the memory experts over at Crucial Technology, a division of Micron Technology, Inc. (http://www.crucial.com). We’ve modified their original inspiration, expanding it to include many of the related concepts found throughout this book. Imagine a printing business. Back when it started, there was only "the guy in charge" and a few employees. They did their printing in a small building, and things were pretty disorganized.

A motherboard is a lot like this printing business, in that the CPU is in charge of getting things done. The other components on the board have all been developed to lend a helping hand. In the way that a business can make more money by choosing different growth paths, system performance has improved using different evolutionary technologies. One path is by getting things done faster. Speeding things up means shipping out more stuff (technical term) in a given work period. More stuff means more money, and the business grows.

Another way is to provide more time in which to get things done. In the world of computers, events take place according to clock cycles (ticks). If it takes 10 ticks to move one byte, then 5 ticks to move the same byte would mean faster throughput. We can either keep the byte the same size and move it in less time (multipliers and half ticks), or we can increase the size of the byte and move more data bits at the original time (bus widths). The difference in each process underlies Rambus and DDR memory technology (discussed in this chapter).

Memory Matrix

Back in the days of DRAM, when this printing business was just getting started, the boss (CPU) would take in a print job, then go running back to the pressman to have it printed. Think of the press man as the memory controller, and the printing press as a RAM chip. The pressman would examine the document and then grab lead blocks carved with each individual letter (bit). He’d put each block into a form (a grid), letter by letter.

Once the form was typeset, the press man would slop on the ink; put a piece of paper under the press, and crank down a handle, printing a copy of the document. (A bit of trivia: the space above and below a line of printing is called the leading—pronounced as "led-ding." This space was the extra room on a lead block surrounding each carved letter.)

Nowadays, you can buy a toy printing kit from a shop (sort of like buying a 386 machine), where each letter is engraved on a piece of rubber that slides into the rail of a wooden stamp handle. When you've inserted a complete line of letters, you ink them with
an ink pad and stamp the line onto a piece of paper. But suppose you could insert an entire line into the rail at once. Wouldn’t that be a whole lot faster? That was the idea behind certain memory improvements we’ll look at later in this chapter (i.e., FPM and EDO).

**Wait States**

One of the big problems with DRAM, to follow our story, was that at any given time, the boss wouldn’t know what the press man was doing. Neither did the pressman have any idea of what the boss was doing. If the boss ran in with a print job while the pressman was re-inking the press, he’d have to wait until the guy was done before they could talk. This is like the CPU waiting for the memory controller to complete a memory refresh.

Memory cells are made up of capacitors that can either hold a charge (1) or not hold a charge (0). One of the problems with capacitors is that they leak (their charge fades). This is similar to how ink comes off each block on that old printing press. A memory refresh is when the controller checks with the CPU for a correct data bit address and recharges a specific capacitor. When a memory refresh is taking place, the CPU must wait for the controller before it can pass on new data. This is a wait state.

You can see that if there were some way to avoid the leakage, the memory controller wouldn’t have to constantly waste time recharging memory cells. The CPU could transfer data more often, using fewer wait states, thereby generating faster throughput. SRAM works with transistors, rather than capacitors. Although they don’t leak their charge, transistors are more expensive to use on a memory chip.

**Interrupts (INT)**

Another problem with DRAM was that if the press man had a box of printed documents ready to go, he’d come running out to the front office and interrupt whatever was going on. If the boss was busy with a customer, then the press man would stand there and shout, "Hey boss! Hey boss! Hey boss!" until eventually he was heard (or punched in the face—an IRQ conflict). Once in awhile, just by luck, the press man would run into the office when there were no customers, and the boss would be free to talk.

Interruptions are known as Interrupt Requests (IRQs) and, to mix metaphors, they are like a two-year-old demanding attention. One way to handle them is to repeat "not now...not now...not now" until it’s a good time to listen. Another way to handle an interruption is to say, "Come back in a minute, and I'll be ready to respond then." We'll look at IRQs in Chapter 5.

**Timing**

One day the boss had a great idea. There was a big clock in the front office (the
motherboard oscillator), and he proposed putting in a window to the pressroom. That way, both he and the press man could see the clock. The boss would then be able to call out to the press man that he had a job to run, and the press man could holler back, "I'll be ready in a minute."

This could also work the other way around, where the press man could finish a job and call out that he was ready to deliver the goods. The boss could shout back that he needed another minute before he could take them, because he was busy with a customer. Both of them could watch the clock for a minute to go by, doing something else until they were ready to talk.

Data transfers are more efficient when the CPU and memory controller can plan out a specific time to communicate. SDRAM provided a way for the memory controller and CPU to understand the same clock ticks and adjust their actions to match each other. The controller became synchronized with the clock, and memory chips became known as Synchronous DRAM (SDRAM).

**DMA Channels**

This plan worked very well. Business increased, and the company expanded. The building grew to fill the property, and new problems began cropping up. To begin with, when the press room moved to the other side of the building, the press man couldn’t see the clock in the front office anymore. That meant they had to install a separate clock in the press room. When the boss had a print job, he would call down to the press room and schedule a time to meet.

The printing department used one clock to synchronize with the boss, but also had a separate clock used to time print jobs to improve efficiency. CPUs have an internal clock, separate from the motherboard clock. Modern components can synchronize to different clocks, depending upon processing requirements. In other words, a memory subsystem might use the processor’s bus to synchronize internal operations. When the resulting information is ready to move out across the motherboard, the system clock becomes the controlling factor.

With business improving, the boss was getting busier and busier. He hired a couple of secretaries to handle walking a print job over to the press room, and the press man hired some assistants to work with some additional printing presses. This is similar to how memory modules came about, where a series of memory chips work together on a single IC card.

Eventually, the boss and the pressman stopped needing to consult about every single piece of paper involved in a particular job. The press man suggested that he be given the authority to make certain decisions as to how to set up the press. You can imagine that by freeing up the boss from having to come down to the press room every few minutes, the
boss ended up with more time to work on business matters. This bypass is essentially what Direct Memory Access (DMA) is all about.

When the CPU agrees, certain operations can bypass the ordinary processing channels and access memory directly. We'll examine DMA channels in Chapter 5, but as devices took on more intelligence of their own, the CPU didn’t have to waste time on simple routines. The ATA specification and UDMA are an outgrowth of this idea.

**Bus Clocking**

Another way the boss and the pressman got things done was to widen the hallways and install some conveyor belts. Back when it was a small business, the CPU transferred data over a memory bus at about the same speed as the memory could handle the bits. With increased processor and memory speeds, the transfer process was limited by buses staying at slower speeds.

In our story, the conveyor belts allowed for higher speed movements up and down the hallways. When the boss had a job to send to the press room, he'd hand it to the "bus boy" and tell him to run it over to the printers. The bus boy would jump on a conveyor belt and go sailing off to the other end of the building. Everyone in the building was talking about how fast they could go, and bus boys began timing each other.

A CPU processes some number of instructions per clock tick. We know that synchronous RAM means that memory controllers can also be set to process events according to clock ticks. CPUs used to run faster than memory, but sometimes memory ran faster than the CPU. Either way, both the CPU and memory were faster than the transfer buses.

Eventually, with some new conveyor belts, a bus boy could walk three steps and be carried all the way down the hall to the other side of the building. Surely, you’ve experienced the thrill of stepping on a conveyor belt and enhancing the speed by walking at the same time as you’re being carried? This is somewhat like the concept of clock multipliers.

Modern bus technology brings together the idea of clock ticks and multipliers, using timing cycles to transfer information. Bits can be moved as a clock tick begins and when the clock tick ends, making for two bits per tick. If the clock is ticking at one million ticks per second (1MHz), we can transfer two million bits of data (2Mbps).

**Cache Memory**

As the company expanded, there was more and more paperwork; with copies of financial statements and records being sent to the accounting department and the government. For a short time, the boss used to send these jobs to the press room—after all, they were a printing company—but that was costing the company money. Finally, he bought some
laser printers for his secretary so they could do these quick print jobs on their own.

Whenever the boss was working up a price quote for a customer, he could set up various calculations and have his secretary print them off. Because they didn’t have to go all the way to the press room (main memory), these temporary jobs were extremely quick. The CPU uses Level 1 and Level 2 caching in a similar fashion.

Level 1 (primary) cache memory is like the boss’ own personal printer, right there by his desk. Level 2 (secondary) cache memory is like the secretary’s printers in the next room. It takes a bit longer for the secretary to print a job and carry it back to the boss’ office, but it’s still much faster than having to run the job through the entire company.

**Memory Buses**

We measure memory speed in nanoseconds (billionths of a second). On the other hand, we measure CPU speed in megahertz (millions of cycles per second) or gigahertz (billions of cycles per second). Blending the two, we come up with how many million instructions per second (MIPS) a processor can complete. Disk speed is measured in milliseconds (thousandths of a second), and a hard disk typically reads information at around 100 reads per second—not to be confused with Revolutions Per Minute (RPM). A floppy disk generally performs 10 reads per second, while RAM can make a billion reads per second. So moving data in and out of RAM is extremely fast—much faster than moving it to and from a disk.

Modern chipsets use a North-South bridge architecture, as we discussed in Chapter 2. The CPU no longer directly connects with the system memory, but works in combination with the North bridge and the memory controller to move data bits in and out of main memory. In Figure 3.1 we can see how the CPU uses additional buses to connect with the L-1 cache inside the processor housing and the L-2 cache outside the housing. In modern computers, the L-2 cache is usually internal to the housing, and we might find an external Level 3 (L-3) cache. The A+ exam covers only Level 1 and Level 2 caches, and the L-2 cache is considered to be outside the CPU.

A CPU has a number of very small places inside the housing where it stores bits and bytes of data. We’ve also seen that memory modules store bits of data in capacitors or transistors. Technically speaking, the storage places inside a CPU are called *registers*. The data storage places on a DIMM are called *cells*. Data is constantly moving in and out of registers and being temporarily stored in main memory or cache memory. Figure 3.1 is a highly stylized drawing of the internal registers, the internal L-1 cache, the external L-2 cache, and the two memory buses. Remember that the PCI bus, or expansion bus, is connected to the other end of the North-South bridge—the South bridge. Refer to Figure 2.10 in Chapter 2.

**Figure 3.1**
Different memory pathways.

**Front Side Bus**

The North bridge handles fast data transfers into and out of system memory and the AGP, working with the memory controller over a bus. This bus is the subject of a fair amount of controversy, with some people calling it the system bus and others calling it the Front Side Bus (FSB). You also may find references to a processor bus or a memory bus. For the moment, we'll refer to it as the front side bus. System performance is based on a timing relationship between the CPU and the FSB, with the bus being clocked at a reduced multiple of the processor. In other words, if you have an 800MHz Pentium 4, with an FSB clocked at 133MHz, the FSB is running at one sixth the speed of the chip. Doubling the speed of FSB throughput, making it 266MHz, increases the bus speed to one third of the chip speed. Much of today’s performance ratings are based on increasing the clock speed of the front side bus.

Here’s another example of FSB speed in relation to CPU speed. Suppose that you have an 800MHz Celeron processor in a machine with a front side bus clocked at 3:1, or a third of the processor speed. Data transfers take place at 266MHz. Now suppose you have a machine with a 1GHz Pentium 4 and an FSB clocked at 4:1, or a quarter of the processor speed. In this instance, data transfers take place at 250MHz (1,000 / 4). Which is the better system, the 800MHz or 1GHz machine? Can we say that a Pentium 4 is always better than a Celeron?

You begin to see that performance is very much tied to the speed of the front side bus. This isn’t to say that the entire system is hanging on the FSB, but to point out that performance is becoming a combined measure of many different components.

**Backside Bus**

The backside bus, shown in Figure 3.1, connects the CPU with an external L-2 cache. Because the data path is extremely short and Level 2 caches are usually comprised of SRAM, data transfers take place at about the same speed as the CPU. We discuss the L-1 and L-2 caches, as well as how a cache operates, in the next sections, but you should have a picture of how the various components of memory are connected. We discuss SRAM in the "Types of RAM" section.

**Cache Memory**

Cache memory is a type of high-speed memory designed to speed up processing. Cache (pronounced "cash") is derived from the French word cacher, meaning to hide. A cache attempts to predict which information is about to be used, using an algorithm (logical formula) based on probabilities and proximity. Proximity means how close something is to something else; in this case, instructions or data bytes.
Typically, a memory cache is a separate SRAM chip, running much faster than DRAM. Whichever instruction or data is most likely to be used next, is stored in the cache. When the CPU looks for the next instruction, the chances are good that it will find it faster in small cache memory than in large main memory.

**The Memory Hierarchy and Caches**

A cache is like an expectation. If you expect to see a piece of information and it’s right beside you, you can access that information much faster than if you had to go look for it. When you open a book and look at page 22, logic dictates that you'll look at the top of the page, then at the middle of the page, then at the bottom of the page, and then at the top of page 23. That’s how computer caching operates.

Think of the example at the beginning of this chapter—the one where you're asked to repeat the words on the last page of the previous chapter. If you were expecting to be asked this question, you could cache the page by putting your finger in the book at that location. This would create a pointer to the page you were going to be asked to read, and you would be waiting for the probable next request—the instruction to read the page.

Because memory size is always increasing, more time is needed to decode increasingly wider addresses and to find stored information. Larger numbers mean we can have more addresses, but a memory register can store only one digit of an address. The larger the numbers, the wider the registers must be, and the wider the corresponding data bus used to move a complete address.

One solution is a memory hierarchy. "Hierarchy" is a fancy way of saying "the order of things; from top to bottom, fast to slow, or most important to least important." Memory hierarchy works because of the way that memory is stored in addresses. Going from fastest to slowest, the memory hierarchy is made up of registers, caches, main memory, and disks. When a memory reference is made, the processor looks in the memory at the top of the hierarchy (the fastest). If the data is there, it wins. Otherwise, a so-called *miss* occurs, at which time the requested information must be brought up from a lower level of hierarchy.

A miss in the cache (that is, the desired data isn't in the cache memory) is called a *cache miss*. A miss in the main memory is called a *page fault*. When a miss occurs, the whole block of memory containing the requested missing information is brought in from a lower, slower hierarchical level. Eventually, the information is looked for on the hard disk—the slowest storage media. If the current memory hierarchy level is full when a miss occurs, some existing blocks or pages must be removed for a new one to be brought in.

A hierarchical memory structure contains many levels of memory, usually defined by access speed. A small amount of very fast SRAM is usually installed right next to the
CPU, matching up with the speed and memory bus of the CPU. As the distance from the CPU increases, the performance and size requirements for the memory are reduced.

---

**Caution** - SMARTDRV.SYS and SMARTDRV.EXE are DOS program utilities that provide disk caching. The efficiency of a cache is reported as its hit ratio. To send an efficiency report to the screen, issue the command `SMARTDRV /S` from a DOS command prompt.

---

## L-1 and L-2 Cache Memory

The Intel 486 and early Pentium chips had a small, built-in, 16KB cache on the CPU called a *Level 1* (L-1), or *primary cache*. Another cache is the Level 2 (L-2), or *secondary cache*. The L-2 cache was generally (not all the time, nowadays) a separate memory chip, one step slower than the L-1 cache in the memory hierarchy. L-2 cache almost always uses a dedicated memory bus, also known as a *backside bus*.

---

**Caution** - For the purposes of the exam, you should remember that the primary (L-1) cache is internal to the processor chip itself, and the secondary (L-2) cache is almost always external. Modern systems may have the L-1 and L-2 cache combined in an integrated package, but the exam differentiates an L-2 cache as being external. Up until the 486 family of chips, the CPU had no internal cache, so any external cache was designated as the "primary" memory cache. The 80486 introduced a 16KB internal L-1 cache. The Pentium family added a 256KB or 512KB external, secondary L-2 cache.

---

Technology tends to move toward consolidating components, for speed and cost efficiencies. The Super I/O chips combined many of the original XT adapters (for example, keyboard, COM and LPT ports) into a single package, and central processors soon moved in the same direction. Although caches were originally placed outside the chip die, new developments paved the way to move them inside the chip. A die, sometimes called the *chip package*, is essentially the foundation for the multitude of circuit traces making up a microprocessor. Today, we have internal caches (inside the CPU housing) and external caches (outside the die).

### Internal and External Memory

When we speak of a chip’s internal bus, we mean that the bus is cast right on the
manufacturing die, along with the chip. These chip packages are sort of like an extremely small motherboard, in that they’re the foundation for the many transistors, diodes, buses, caches, and a host of other electrical components we call a central processor. Don’t confuse a chip package with a chipset—the entire set of chips used on a motherboard to support a CPU. The CPU is a chip package.

---

**Note** - A manufacturing mask is the photographic blueprint for the given chip. It is used to etch the complex circuitry into a piece (chip) of silicon.

---

An external bus is the place where information moves out of the chip die to another destination (for example, an L-2 cache). Because bus width is typically measured by the number of bits that the bus can process at one time, we can have an 8-, 16-, 32-, and 64-bit bus. The number of data lines, each carrying a stream of bits, indicates the width of the bus. The bus width generally depends upon where the processor is directing information.

### Memory Pages

Typically, memory is divided into chunks (or blocks). At the main memory level, a chunk is referred to as a *memory page*. At the cache level, a chunk is called a *cache block* or a *cache line*. Keep in mind that if a register is 8 bits wide, a cell contains one of the bits in that register. Technically, only a CPU has registers. Memory chips have capacitors and transistors.

The memory controller keeps track of the state and location of the charges held in the capacitors and transistors of a memory chip, as we indicated earlier, in our printing business story. This combination of states and locations is called an *address*. The charges in the memory chips are created by the original byte transmissions, coming from the CPU's registers. The CPU sends data to memory in order to empty its registers for more calculations.

In other words, the CPU has some information it wants to get rid of. It sends that information to the memory controller. The memory controller shoves it into whichever capacitors are available and keeps track of where it put the data. It keeps track of the bits by assigning memory addresses to each bit of information.

### Ranges

Changes in memory chips and controllers are similar to how flat, one-page spreadsheets developed the concept of named ranges. DRAM is usually accessed through paging. A
page is a related group of bytes (with their bits), similar to a range on a spreadsheet. It can be from 512 bits to several kilobytes, depending on the way the operating system is set up.

Without ranges, a spreadsheet formula must include every necessary cell in the spreadsheet. For example, we might have a formula something like =SUM (A1+B1+C1+D1+E1). Now suppose that we assign cells C1, D1, and E1 to a range, and call that range "LastWeek." We can now change the formula to include the range name: =SUM(A1+B1+"LastWeek").

When a spreadsheet formula uses a named range, this is analogous to the memory controller giving a unique name to a range of charges. This range of charges is called a page address, and with a page address, the controller doesn’t have to go looking for data in every single capacitor or transistor.

**Fast Page Mode (FPM)**

Dynamic RAM (DRAM) originally began with Fast Page Mode (FPM) back in the late 1980s. In fast page mode, the memory controller makes an assumption that the data read/write following a CPU request will be in the next three pages (ranges), very much like a cache. This is somewhat like having a line of letters all ready to go in the toy stamp we spoke about in our printing business story.

Using FPM, the controller doesn’t have to waste time looking for a range address for at least three more times: it can read-assume-assume-assume. Note the three pauses, as we’ll mention burst cycles in a moment.

---

**Note** - When the controller passes through the memory chip, it turns off something called a data output buffer when it reads the page it just read or wrote to. This process takes approximately 10 nanoseconds.

---

Fast Page Mode is capable of processing commands at up to 50 ns. Fifty nanoseconds is fifty billionths of a second, which used to be considered very fast. Remember that the controller first moves to a row; then to a column; then retrieves the information at an X-Y coordinate—a matrix address.

Once the information is validated, the controller hands it back to the CPU. The column is then deactivated, and the data output buffer is turned off. Finally, the column is prepared for the next transmission from the CPU. The memory enters a 10 ns wait state while the capacitors and transistors are precharged for the next cycle.
Extended Data Output (EDO) RAM

FPM evolved into Extended Data Out (EDO) memory. The big improvement in EDO was that once the column of data was deactivated, the data remained valid until the next cycle began. In other words, FPM removed the data bits in the column address and deactivated the column (data output buffer). EDO, on the other hand, kept the data output buffer active until the beginning of the next cycle, leaving the data bits alone.

EDO memory is sometimes referred to as hyper-page mode, and is a specially manufactured chip that allows a timing overlap between successive read/writes. The data output buffers are not turned off when the memory controller finishes reading a page. Instead, the CPU determines the start of the deactivation process when it sends a new request to the memory controller. The result of this overlap in the process is that EDO eliminated the 10 ns. per cycle delay of fast page mode, generating faster throughput.

Both FPM and EDO memory are asynchronous. (In the English language, the "a" in front of synchronous is called a prefix. The "a" prefix generally mean "not," or "the opposite." ) In asynchronous memory, the memory controller and the system clock are not synchronized. DRAM is asynchronous memory. In asynchronous mode, the CPU and memory controller have to wait for each other to be ready before they can transfer data. Remember that everything on an asynchronous motherboard listens to the clock ticks coming from the motherboard oscillator.

Originally, motherboard clocks ran between 5–66MHz. The early x86 processors ran at 5–200MHz. When Pentium Pro motherboards stabilized at 66MHz, CPU speeds went up to between 200MHz and 660MHz. Suppose a motherboard has a 66MHz clock with a clock multiplier of 2. In this situation, the CPU runs at 133MHz (66 * 2). Remember, the CPU runs at the speed of the motherboard clock.

Now suppose a memory chip is synchronized to that same motherboard's 66MHz clock. The memory controller will have to wait 2 clock ticks before it can interrupt the CPU, unless it accidentally happens to catch the CPU at exactly the right time. Think about it: the CPU is hearing two ticks for every one tick the memory controller is hearing. If the CPU processes one instruction for every clock tick, then it will do two things before it's ready to be interrupted by the memory controller. This makes the controller seem a bit slow in the head.

Supplementary Information

If you choose, you may jump to the "Types of RAM" section at this time. This section is a more technical discussion of wait states and the memory standards used in rating memory performance.

A 60 ns. DRAM module, using fast page mode, might run in 5-3-3-3 burst mode timing.
The first access takes 5 cycles (on a 66MHz system), or about 75 ns. The next three accesses take only 3 cycles, because they "assume" the range addresses. It happens that this works out to about 45 ns, with a ninety nanosecond savings in time (about 40 percent). Regular old page mode would be 5-5-5-5 burst mode.

EDO RAM became popular for a time, with a typical burst cycle of 5-2-2-2, yielding about a 22 percent savings in time over FPM memory. 50-nanosecond FPM isn’t used anymore. Both FPM and EDO memory eventually gave way to Synchronous DRAM (SDRAM), then to Rambus memory, then to DDR-SDRAM (discussed later in this chapter). SDRAM developed a burst cycle of 5-1-1-1, which is even faster than EDO RAM, so SDRAM became the most popular type of memory toward the end of the 1990s.

**The PC100 Standard**

The speed of the actual memory chips in a module is only part of how we evaluate memory speed. The other factor is the underlying printed circuit board. Due to the physics of electricity and electronic parts, a module designed with parts that can run at 100MHz may never reach that speed. It takes time for the signals to move through the wire, and the wire itself can slow the signal speed. This led to the same old ratings problems that processors were causing.

Motherboard speeds eventually increased to 100MHz, and CPU speeds went beyond 500MHz. The industry decided that the original SDRAM chips should be synchronized at 100MHz. Someone had to set the standards for the way memory modules were clocked, so Intel developed the PC100 standard. This initial version of the standard made sure that a 100MHz module was really capable of, and really did run at 100MHz. Naturally, this created headaches for memory manufacturing companies, but the standard really helped in determining system performance.

The PC100 SDRAM modules required 8 ns DRAM chips, capable of operating at 125MHz. This provided a margin of error, making sure that the overall module would be able to run at 100MHz, according to the standard. The standard also called for a correctly programmed EEPROM, on a properly designed circuit board.

At 100MHz and higher, timing is absolutely critical, and everything from the length of the signal traces to the construction of the memory chips themselves is a factor. The shorter the distance the signal needs to travel, the faster it runs. Noncompliant modules, those that didn’t meet the PC100 specification, could significantly reduce the performance and reliability of the system. The standard caught on. Note that unscrupulous vendors would sometimes use 100MHz SDRAM chips and label the modules as PC100 compliant, a similar situation occurred with processor chips (mentioned in the next chapter). This didn’t necessarily do any harm, but consumers weren’t getting what they were told they were getting.
As memory speeds increased, the PC100 standard was upgraded to keep pace with the new modules. Intel released a PC133 specification, synchronized to 133MHz, and so it went. PC800 RDRAM was released to coincide with the 800 series chipset, running at 800MHz, and these days, we see a PC1066 specification, designed for high-speed RDRAM memory. As bus speeds and module designs change, so too does the specification.

**Summary—Memory**

We’ve seen that memory can be broadly divided into two categories: memory that the system can change, and that which the system cannot change. RAM and ROM are the beginning concepts for understanding memory. The BIOS is where the motherboard remembers the most basic instructions about hardware, but the CMOS is where basic system settings are stored. BIOS and CMOS are different, in that CMOS requires a small amount of electricity to maintain its settings.

BIOS and CMOS can be changed, but not without some effort. Make sure you know the acronyms associated with these chips and the ways in which they can be updated. Additionally, you should have a comfortable sense of understanding about the following points, having to do with memory:

- The central processing unit, the memory controller, cache memory, and the system clock
- How timing affects performance, and the difference between asynchronous and synchronous data transfers
- The North bridge and South bridge architecture, and how the front side bus stands between the CPU and the North bridge (see Chapter 2).

Be sure that you have a good understanding of how data can be stored in memory cells, using ranges and range addresses. You should be able to differentiate between FPM and EDO memory, and dynamic (asynchronous) versus synchronous RAM. You won’t be asked to calculate or remember burst cycles, but you should understand the concept of moving data as opposed to waiting cycles. A critical concept is the relationship between clocks (oscillators) and data transfers. If you can’t remember how the motherboard uses timing cycles, go back and skim Chapter 2.

**Types of RAM**

As you’ve probably noticed, RAM is a sort of universal name for memory. Beyond that, each new name refers more to the mode of operation than an actual type of memory. The "D" in DRAM refers to the dynamic access. The "S" in SRAM and SDRAM refers to
synchronizing the chip to the motherboard. EDO RAM refers to the extended data output whereby the column address isn’t turned off after a read.

**Dynamic RAM (DRAM)**

First, *dynamic* means moving or always changing. In dynamic RAM, electricity always has to be moving through the chip to keep refreshing the memory. DRAM is the basic type of memory chip, and everything that came later was mainly a way to get address information faster. Dynamic also means that data can be moved into or out of memory, over and over again, for as long as power is available.

Data in a DRAM chip is stored in very compact form, with each bit using only its own capacitor and accompanying transistor. Once again, if the capacitor has a charge, the computer reads that as a 1. If it does not have a charge, it’s a 0. Capacitors have a tendency to leak their charge fairly quickly. For that reason, they often have to be refreshed. The controller refreshes the DRAM sequentially during any given operation.

**Synchronous RAM (SRAM)**

SRAM introduced a clock that was actually built onto the memory module. This allows the module to be synchronized to the motherboard clock, introducing the term synchronous, or synchronized memory. Another feature of SRAM chips was that they weren’t as dense as DRAM. The chips used a matrix of 6-transistors and no capacitors. Transistors don’t require power to prevent leakage, so SRAM didn’t have to be refreshed on a regular basis. By staying charged, the data changed only when a register changed, making the chips a lot faster than DRAM.

Because of the extra space in the matrix, SRAM uses more chips than DRAM for the same amount of storage space. Manufacturing costs are higher, so although DRAM was slower, the higher density and lower cost still made them popular in many machines. However, combining the best of both technologies quickly led to the development of SDRAM chips.

---

**Caution** - L-2 memory caches are usually on SRAM, which is extremely fast (as fast as 7–9 ns and 2–5 ns for ultra fast SRAM). Level 2 cache is usually installed in sizes of 256KB or 512KB.

SRAM is also used for CMOS configuration setups and requires a small amount of electricity, provided by a backup battery on the system board, to keep its data. It comes on credit-card-sized memory cards, is available in 128KB, 256KB, 512KB, 1MB, 2MB, and 4MB sizes, and has a battery life of 10 or more years.
Synchronous DRAM (SDRAM)

When the concept of synchronizing a memory chip to the motherboard proved successful, the idea was retrofitted to DRAM chips. Synchronous DRAM synchronizes the interrupt requests to the motherboard and works just like SRAM, only with the higher density (and lower cost) of the DRAM chip. Most SDRAM controllers are built into the North bridge of the motherboard chip set.

SDRAM uses the synchronization feature to eliminate the waiting problems between the CPU and the memory controller. When the CPU is ready to access data from memory, it goes to a specified clock point. The CPU knows when operations are going to be completed and when data is going to be available. No wait states means better performance (approximately 20 percent better than EDO memory, in most cases).

As we’ve seen, DRAM, fast page mode, and extended data output mode all measured memory speeds in nanoseconds (billionths of a second). A 70 ns part would be a "7," a 60 ns part a "6," and so on. The lower the number, the faster the memory. With the introduction of SDRAM, this measurement became less accurate. At such short intervals, per-second time began to lose any real value. Instead, the transfer speed of the front side bus began to make more sense as a measure of speed. For this reason, SDRAM modules began to be rated as 66MHz, 100MHz, 133MHz, 800MHz, and so on.

Rambus DRAM (RDRAM)

RDRAM comes out of technology developed originally by Rambus, Inc., for the Nintendo 64 gaming system. It’s not that new, but it seems new because Intel started to use it with its Pentium 4 processors and 800-series chipset. Rambus memory is integrated onto Rambus Inline Memory Modules (RIMMs). The modules use Rambus DRAM (RDRAM) chips. We discuss the Rambus modules later in this chapter.

RDRAM chips are synchronized to the processor’s memory bus (not the motherboard clock). Therefore, the processor won’t request something at mid tick (the reverse of an interrupt). In DRAM, the CPU is synchronized to the motherboard (which is fairly slow), but the memory chip is not. In SRAM and SDRAM, both the CPU and the memory are synched to the motherboard. In RDRAM, the synchronization is no longer tied to a slow motherboard, but is synchronized to the memory bus. The memory bus clock is derived from the processor, but it still uses a multiplier of the motherboard clock.

Video RAM (VRAM)

VRAM and WRAM have been supplanted by DDR memory chips, but you may find a question about VRAM (pronounced "vee-ram") on the exam. Video RAM was designed
to provide two access paths to the same memory address. It’s as if VRAM were a café that has two doors, one in the front and one in the back. Information came in one "entrance" at the same time that other information flowed out the other "exit." When the video controller read the memory for information, it accessed an address with one of the paths. When the CPU wrote data to that memory, it accessed the address via the other path. Because of these two access paths, we say that VRAM was dual-ported.

Manipulating graphics is processing-intensive, and so this ability to push data in and out of the chip at the same time helps a moving image appear continuous. VRAM chips were about 20 percent larger than DRAM chips, due to extra circuitry requirements. Modern computers usually have modest graphics processing integrated right onto the motherboard, with the AGP as a convenient way to add an extra video card.

---

**VRAM, WRAM, and AGP** - The AGP acronym stands for Accelerated Graphics Port. Most computers include this accelerated port, which is an integrated part of the I/O system. An AGP is not the same thing as VRAM or a video accelerator card, nor is it the same thing as today's integrated graphics. Although some video cards still use the expansion bus, most connect with the port.

To say that a computer has "AGP memory," or "comes with AGP," can be confusing at best. At worst, it can demonstrate a faulty knowledge of the distinction between video memory and I/O subsystems. AGP is discussed in Chapter 7.

WRAM is short for Windows RAM, and has no connection with Microsoft, even though the acronym includes the word "Windows." WRAM, like VRAM, was dual-ported, but used large block addressing to achieve higher bandwidth. Additional features provided better performance than video RAM at lower manufacturing costs. With the advent of AGP and DDR RAM, both video and Windows RAM have faded from the marketplace. That's not to say that add-on graphics accelerator cards have vanished.

---

**Packaging Modules**

We've discussed how memory chips work on the inside, but you'll need to know how these chips are installed on a motherboard. Once again, most of the changes came about either to make maintenance easier, or to avoid bad connections. Keep in mind that installing a combined "unit" or module of some kind is less expensive than having many individual units to install.
Dual Inline Package (DIP)

Originally, DRAM came in individual chips called dual inline packages (DIPs). XT and AT systems had 36 sockets on the motherboard, each with one DIP per socket. Later, a number of DIPs were mounted on a memory board that plugged into an expansion slot. It was very time-consuming to change memory, and there were problems with chip creep (thermal cycling), where the chips would work their way out of the sockets as the PC turned on and off. Heat expanded and contracted the sockets, and you’d have to push the chips back in with your fingers.

To solve this problem, manufacturers finally soldered the first 640 kilobytes of memory onto the board. Then the problem was trying to replace a bad chip. Finally, chips went onto their own card, called a single inline memory module or SIMM. On a SIMM, each individual chip is soldered onto a small circuit board with an edge connector (Holy Pentium II architecture, Batman!).

Connectors: Gold Vs. Tin - SIMMs and DIMMs come with either tin (silver-colored) or gold edge connectors. Although you may assume that gold is always better, that’s not true. You want to match the metal of the edge connectors to the metal in the board’s socket. If the motherboard uses gold sockets, use a gold SIMM. Tin sockets (or slots) should use tin edge connectors. The cost difference is minimal, but matching the metal type is critical.

Although it’s true that gold won’t corrode, a gold SIMM in a tin connector will produce much faster corrosion in the tin connectors. This quickly leads to random glitches and problems, so look at the board and match the color of the metal.

It’s important to note, too, that each module is rated for the number of installations, or insertions. Each insertion causes scratches, and the more metal that is scratched off, the worse the connection becomes. In flea market exchanges and corporate environments, modules are subjected to constant wear and tear, and nobody is looking at the rated number of insertions.

Single Inline Memory Modules (SIMMs)

When DRAM chips were placed in a line on their own circuit board, it gave rise to the term inline memory. Once the chips were formed into a module, the entire module would fit into a socket on the board. These modules and sockets are referred to as memory banks. Depending on how the chips are connected on their own little circuit board, the
module is called either a single, or dual inline memory module (SIMM or DIMM).

Caution - SIMMs come in both 30-pin and 72-pin versions. The 30-pin module is an 8-bit chip, with 1 optional parity bit. The 72-pin SIMM is a 32-bit chip, with 4 optional parity bits.

The memory bus grew from 8 bits to 16 bits and then from 32 bits to 64 bits wide. The 32-bit bus coincided with the development of a SIMM, which meant that the 32-bit-wide data bus could connect directly to 1 SIMM (4 sets of 8 bits). However, when the bus widened to 64 bits, rather than making a gigantic SIMM, we started using two SIMMs in a paired memory bank. The development of the 64-bit-wide DIMM superseded the SIMMs, and only used one module per socket again.

Caution - SIMMs and DIMMs are sometimes referred to as chips, but they are really a series of chips (modules). DRAM itself is a chip, and DRAM chips are grouped together to form SIMMs and DIMMs. SIMMs can come with a varying number of pins, including 30-pin and 72-pin. (Even though the 72-pin module could have chips on both sides, it was still a SIMM.)

Be careful when you read a question on the exam that you don’t accidentally agree that a SIMM is a memory chip. A good way to keep alert is that chips have RAM in their name.

Dual Inline Memory Modules (DIMMs)

Dual inline memory modules are very similar to SIMMs, in that they install vertically into sockets on the system board. DIMMs are also a line of DRAM chips, combined on a circuit board. The main difference is that a DIMM has two different signal pins, one on each side of the module. This is why they are dual inline modules.

The differences between SIMMs and DIMMs are as follows:

- DIMMs have opposing pins on either side of their board. The pins remain electrically isolated to form two separate contacts—a dual set of electrical contacts.
- SIMMs also have opposing pins on either side of the board. However, the pins are connected, tying them together. The connection forms a single electrical contact.
DIMMs began to be used in computers that supported a 64-bit or wider memory bus. Pentium MMX, Pentium Pro, and Pentium II boards use 168-pin modules. They are 1 inch longer than 72-pin SIMMs, with a secondary keying notch so they'll only fit into their slots one way.

**Don’t Mix Different Types of Memory** - Mixing different types of SIMMs or DIMMs within the same memory bank prevents the CPU from accurately detecting how much memory it has. In this case, the system will either fail to boot, or will boot and fail to recognize or use some of the memory.

You can, however, substitute a SIMM with a different speed within the same memory bank, but only if the replacement is equal to or faster than the replaced module.

All memory taken together (from all memory banks) will be set to the speed of the slowest SIMM.

**Rambus Inline Memory Modules (RIMMs)**

Rambus inline memory modules (RIMMs)—as opposed to dual inline—use Rambus Dynamic RAM (RDRAM) chips. Previously, on a standard bi-directional bus, data traveled down the bus in one direction, then returning data reversed back up the same bus in the opposite direction. It did this for each bank of memory, with each module being addressed separately. A wait occurred until the bus was ready for either the send or reverse.

RIMMs use a looped system, where everything is going in one direction (unidirectional). In a looped system, data moves forward from chip to chip and module to module. Data goes down the line, then the results data continues forward on the wire in the same direction. The results data doesn't have to wait for downstream data to finish being sent.

These chips are set on their modules contiguously (next to each other in a chain) and are connected to each other in a series. This means that, if an empty memory bank socket is in between two RIMM chips, you must install a continuity module, which is a low-cost circuit board that looks like a RIMM, but that doesn't have any chips. All it does is allows the current to move through the chain of RDRAM chips. It's like Christmas tree lights wired in series, where one bulb is missing.

**Note** - Latency is a combination of two things: You don't have to wait for the
bus to turn around, and the cycle time is running at a fast 800MHz, so you
don’t have to wait very long for the next cycle. Using RDRAM chips, signals
go from one module to the next to the next, and the throughput is triple that of
100MHz SDRAM. You can also use two to four RDRAM channels (narrow
channel memory) at the same time. This can increase throughput to either
3.2GB or 6.4GB.

Why Memory Becomes Corrupted - Most DRAM chips in SIMMs or
DIMMs require a parity bit because memory can be corrupted in the following
two ways: Alpha particles can disturb memory cells with ionizing radiation,
resulting in lost data, or electromagnetic interference (EMI) can change the
stored information.

A DRAM cell (transistor) shares its storage charge with the bit line. This
creates a small voltage differential, which can be sensed during read access.
The differential can be influenced by other nearby bit line voltages. This, along
with other electrical noise, can corrupt the electrical charge in the memory cell.

Supplementary Information

Throughout this book we’ve taken the time to explore technical details outside the scope
of the A+ exam. In some instances, these supplementary segments serve as real-world
examples of how the concepts you’ll be tested on are used out in the field. In other
instances, you should be aware that low level physics, electronics, engineering, and
design schemes have produced long-term consequences. You’ll be faced with questions
that result from how modern computers have come to terms with these consequences, and
you can either remember "just the facts," or have a sense of history as to how the facts
came into being. If you choose, you may skip to the "Memory Diagnostics—Parity"
section at this point.

The following section is a more technical examination of how Rambus memory went in
one direction, and DDR memory went in another. Each type of memory develops
problems based on RIMMs using serial transfers and DDR using parallel transfers. By
seeing the problems, you should end up with a much better understanding of some of the
other concepts covered in this book, and the underlying reasoning behind many of the
exam questions. If you can figure out the reason for the question, many times you can
also figure out the answer—even if you can't remember the specific details.

Serial Transfers and Latency
Aside from some other technical features we'll explore in a moment, RDRAM has a latency factor. In order to understand latency, let's take a look at the difference between serial and parallel transfers. We'll refer to these concepts again in Chapter 9, as they are fundamental concepts to computer technology. Think of a train, like the ones you often see in movies. When the hero has to chase the bad guy through the train, he starts at one end and goes from car to car. This is a serial process, where he moves through a series of cars. He begins at one end of a car, then walks all the way through it, usually looking for someone, and leaves by a door connecting to the next car in the series. Then the process starts all over again, until he either reaches the end of the train or someone gets killed.

Now suppose we take that same train, but this time we don't have a hero chasing a bad guy. Instead, let's imagine a train full of people on their way to work. If there was only one door at the back of the train, it would take forever when the train pulled into a station and everyone wanted to get off. To fix that problem, each car has its own door. When the train comes to a stop, everyone turns to the side facing the platform: The doors in each car open up, and a stream of people leaves each car simultaneously. This is a parallel transfer, with eight passengers leaving their own car in parallel (side by side) with each other.

One of the problems with Rambus memory is that the RIMMs are connected to the bus in a series. A data item has to pass through all the other modules before it reaches the memory bus. The signal has to travel a lot farther than it does on a DIMM, where the bus uses parallel transfers. The longer distance introduces a time lag, called latency. The longer the delay before the signal reaches the bus, the higher the latency. In a game, data generally moves in long streams, so serial transfers aren't a problem. But in a typical PC, data routinely moves in short bursts, and latency becomes a problem.

In a DDR or SDRAM system, each DIMM is connected, individually and in parallel, to the data bus. Regardless of whether the system uses a single DIMM or multiple DIMMs, transfer times are essentially the same. We'll discuss DDR memory at the end of this chapter.

**Narrow Channel Bus**

Earlier Pentiums had a data bus up to 64 bits wide and transferred data using a parallel process. The DIMM was also 64 bits wide, which meant that data could be moved across the memory bus in 64-bit chunks, or 8 bytes, per second. Remember that a byte is equal to 8 bits. Another way of looking at it is that a bit is one-eighth of a byte. Therefore, 64 bits divided by 8 equals 8 bytes.

All the memory systems that we've talked about are known as **wide channel systems** because the memory channel is equal to the width of the processor data bus. RDRAM is known as a **narrow channel system** because data is transferred only 2 bytes (16 bits) at a time. This might seem small, but those 2 bytes move extremely fast. The Rambus data
bus is 18 bits wide, as opposed to the standard 32 or 64 bits, but the system sends data more frequently. It reads data on both the rising and falling edges of the clock signal, a process also used in Dynamic Data Rate (DDR) memory (discussed at the end of this segment).

Up until RDRAM, the fastest chips had a throughput of 100MHz. Remember that 8 megabytes moving in parallel means that 8MB is transferred every 1 second. At 100 clock ticks (cycles) per second, that means 100 times 8, or 800 megabytes per second. RDRAM chips move 2 megabytes per cycle (one on the up tick, one on the down tick). At 800MHz, that means RIMMs move 1,600MB per second (800 times 2 bytes), which translates to 1.66 gigabytes (GB) or a billion bytes—about twice as fast as SDRAM.

Another way to think of it is an example we heard, involving driving to the store. If you go out of your way, you can take 10 minutes to drive 10 miles on a highway at 60 mph. If you go directly to the store, you can take 4 minutes to drive only 2 miles at 30 mph. You might drive a whole lot faster on the highway, but you'll get to the store faster on the straight-line route. In this example, the store is the memory controller and bus.

**Continuity Modules**

Because Rambus memory works with serial transfers, all memory slots in the motherboard must have an installed module. Even if all the memory is contained in a single module, the unused sockets must have a PCB, known as a continuity module, to complete the circuit. This is similar to old strings of Christmas tree lights, wired in series, where every socket required a bulb. Since the current moved through each bulb in order to get to the next, a missing or burnt-out bulb would stop the flow of current to the entire string.

---

**Note** - One of the interesting things about very high-speed memory is that as signals move across the circuits in different ways, the modules begin to look more like microscopic networks. When we speak about network architectures in Chapter 8, we'll refer to signal interference as packets collide. This is similar to what's taking place in modern memory modules. At such high speeds, the physics of impedance and capacitance have more of an impact on the way data signals arrive at their destinations. Configuring a serial pathway has one set of problems, while parallel pathways have different problems. We won't go into all the low-level details, but you may find it interesting to do some research on the technical specifications of Rambus and DDR memory.

---

**Double Data Rate SDRAM (DDR SDRAM)**
DDR SDRAM probably won't be on the exam, but we'll mention it because technology is rapidly abandoning the previous types of SDRAM, along with RDRAM. Intel has stopped developing chipsets for RDRAM technology, and has shifted its resources to DDR SDRAM and DDR-II, using a newer version of the i845E chipset. Double Data Rate (DDR) came about as a response to RDRAM specifications put together by Intel and Rambus. Intel eventually bought Rambus and began licensing the technology for a fee. In much the way that IBM created Micro Channel Architecture (MCA) and charged a fee, both situations led to the development of separate consortiums for a different standard. DDR is an open architecture, and comes from a consortium of non-Intel manufacturers.

Note - DDR and Rambus memory are not backward compatible with SDRAM.

In our discussion of motherboards in Chapter 2, we mentioned that AMD developed faster processing by using a double-speed bus. Instead of using a full clock tick to run an event, they use a "half-tick" cycle, which is the voltage change during a clock cycle. In other words, as the clock begins a tick, the voltage goes up (an up tick) and an event takes place. When the clock ends the tick, the voltage goes down (a down tick) and a second event takes place. Therefore, every single clock cycle can have two memory cycle events. The AMD Athlon and Duron use the DDR specification with the double-speed bus.

DDR (double data rate) memory is the next generation SDRAM. Like SDRAM, DDR is synchronous with the system clock. The big difference between DDR and SDRAM memory is that DDR reads data on both the rising and falling edges of the clock tick. SDRAM only carries information on the rising edge of a signal. Basically, this allows the DDR module to transfer data twice as fast as SDRAM. For example, instead of a data rate of 133MHz, DDR memory transfers data at 266MHz.

DDR modules, like their SDRAM predecessors, are called *DIMMs*. They use motherboard system designs similar to those used by SDRAM; however, DDR is not backward compatible with SDRAM-designed motherboards. DDR memory supports both ECC (error correction code, typically used in servers) and nonparity (used on desktops/laptops.)

Aside from the royalty fee to Intel, Rambus memory is more expensive to produce than DDR SDRAM. DDR is designed around an open architecture, meaning no royalties. Rambus die (chips) are much larger than SDRAM or DDR die, which means that fewer parts can be produced on a wafer. Even so, Silicon Integrated Systems (SiS), the manufacturer of the first DDR400 chipset for the P4, has launched a PC1066 RDRAM system. Intel is still validating PC1066 support for the i850E chipset. DDR is the most popular DRAM at the moment, but RDRAM is still being used in a few high-end desktops and workstations.
RDRAM uses a 16-bit bus for the data signals, and this narrow 16-bit path is the main reason why RDRAM is able to run at speeds up to 533 MHz (with double data rate, an effective 1066 MHz). However, one of the problems with parallel transfers at high speeds is something called skew. The longer and faster the bus gets, the more likely it is that some data signals will arrive too soon or too late, based on the clock signal. This is where the very high speeds in today's systems are rekindling interest in serial transfers. The parallel port used to be the fastest port, but fast serial USB ports and serial ATA transfers (IDE drives, discussed in Chapter 6) are making parallel transfers seem out-of-date.

RDRAM also developed a different type of chip packaging, called Fine Pitch Ball Grid Array (FPBGA). Most DDR SDRAM uses a Thin Small Outline Package (TSOP). TSOP chips have fairly long contact pins on each side, as opposed to FPBGA chips, which have tiny ball contacts on the underside. The very small soldered balls have a much lower capacitive load than the TSOP pins. DDR SDRAM using the FPBGA packaging is able to run at 200-266MHz, whereas the same chips in a TSOP package are limited to 150-180MHz.

**DDR-II and "Yellowstone"

DDR-II may be the end of Rambus memory, although people have speculated that RDRAM wouldn't last for quite some time. DDR-II extends the original DDR concept, taking on some of the advantages developed by Rambus. DDR-II uses FPBGA packaging for faster connection to the system, and reduces some of the signal reflection problems (stubs) of the original DDR. However, as bus speeds continue to increase, so too does latency. DDR-II is expected to enter the consumer market sometime in 2003. RDRAM is expected to continue to exist in 2003, but with very little chipset support. This makes it likely that DDR-II will take over from both DDR SDRAM and RDRAM.

The current PC1066 RDRAM can reach 667 MHz speeds (PC1333), so Samsung and Elpida have announced that they are studying 1333MHz RDRAM and even 800MHz memory (PC1600). These systems would most likely be used in high-end network systems, but that doesn't mean that RDRAM would be completely removed from the home consumer market. Rambus has already developed a new technology, codenamed "Yellowstone," which should reach 400MHz, using octal data rates. This may lead to 3.2GHz memory, with a 12.4GB/s throughput. With a 128 bit interface Rambus promises to achieve 100GB/s throughput. Yellowstone technology is expected to arrive in game boxes first, with PC memory scheduled for sometime around 2005.

**Summary—Memory Modules**

Everything about the packaging of memory chips rests on the concept of modules. These modules are vaguely like tiny motherboards within a motherboard, in that they, too, are integrated circuit boards. The big difference between DRAM and SDRAM is the synchronization feature. Be sure you understand how SDRAM uses timing cycles to
more efficiently interrupt the CPU. Remember, SRAM is extremely fast and is used in secondary caches; SDRAM is a type of main memory.

It’s all well and good to know how SDRAM differs from Rambus RAM, but you’re also going to have to be able to differentiate between SIMMs and DIMMs. Inline memory modules are the small IC cards you install in your machine when you upgrade your memory. You won’t have to remember clock speeds and the exact number of pins, but you’ll definitely be tested on the different types of modules.

**Memory Diagnostics—Parity**

Parity is the state of either oddness or evenness assigned to a given byte (not bit) of data. Parity checking is the way that the computer uses a special set of logical rules and chips to make decisions based on the parity (state) of a particular byte. When a PC runs through the POST (Power On Self-Test), commonly called a cold boot, memory integrity is one of the first things tested. On many machines you can see this taking place as a rapidly-increasing number displays on the screen before the operating system begins to load. This checking is designed to verify the accuracy (integrity) of the bits contained in memory, and it happens after every read/write operation.

---

**Caution** - Remember that computers think in binary bits and bytes. The binary computer language uses "words" composed of 1s and 0s.

---

Originally, parity checking was a major development in data protection. At the time, memory chips were nowhere near as reliable as they are today, and the process went a long way towards keeping data accurate. The parity circuit allowed the CPU to compare what it sent to main memory with what it read from main memory. Parity checking is still the most common (and least expensive) way to check whether or not a memory cell can accurately hold data. A more sophisticated (and expensive) method uses Error Correcting Code (ECC).

**Even and Odd Parity**

Parity checking puts a byte into memory along with its parity bit and then reads the byte from memory. If the byte plus the parity bit matches what was sent, the memory is okay. Otherwise, we have a parity error. Parity can be set to odd, even, or off. The parity circuit works by adding one bit to every byte of data, resulting in nine bits. (Remember that a byte is already eight bits.) The value of any given bit (1 or 0) is determined at the time that data is written to memory. Then, a 1 or 0 is assigned to the parity bit, depending on whether the overall byte is made up of an odd or even number of 1s. **Figure 3.2** shows
various bytes of data with their additional parity bit.

**Figure 3.2**
Odd and even parity.

In plain odd and even parity, every byte gets 1 parity bit attached, making a combined 9-bit byte. Therefore, a 16-bit byte has 2 parity bits, a 32-bit byte has 4 parity bits, and so forth. This produces extra pins on the memory module, and this is one of the reasons why various DIMMs and SIMMs have a different number of pins.

Even parity checking, is where the total of all the 1 bits in a byte must equal an even number. If five of the bits are set to 1, the parity bit will also be set to 1 to total six (an even number). If 6 bits were set to 1, the parity bit would be set to 0 to maintain the even number six. If the parity is set to "even" and a returning byte contains an odd number of "ones," the system knows the data is corrupted. This is called a *parity error*.

---

**Caution** - In even parity, the overall number of 1s must come out to an even number.

---

Odd parity works in the reverse of even parity, but the concept is the same. The total number of 1s in any given byte must come out to an odd number. Once again, this is done by using either a 1 or a 0 for the parity bit, as you can see in **Figure 3.2**.

---

**Caution** - You’ll receive a parity error if the parity is odd and the parity circuit gets an even number, or if the parity is even and the parity circuit gets an odd number. The circuit can’t correct the error, but it can detect that the data is wrong.

---

**Fake or Disabled Parity** - Some computer manufacturers install a less expensive "fake" parity chip that simply sends a 1 or a 0 to the parity circuit to supply parity on the basis of which parity state is expected. Regardless of whether the parity is valid, the computer is fooled into thinking that everything is valid. This method means no connection whatsoever exists between the parity bit being sent and the associated byte of data.

A more common way for manufacturers to reduce the cost of SIMMs is to
simply disable the parity completely, or to build a computer without any parity checking capability installed. Some of today’s PCs are being shipped this way, and they make no reference to the disabled or missing parity. The purchaser must ensure that the SIMMs have parity capabilities, and must configure the motherboard to turn parity on.

Error Correction Code (ECC)

Parity checking is limited in the sense that it can only detect an error—it can't repair or correct the error. This is because the circuit can't tell which one of the eight bits is invalid. Additionally, if multiple bits are wrong but the result according to the parity is correct, the circuit will pass the invalid data as okay.

Error correction code (ECC) uses a special algorithm to work with the memory controller, and it adds an error correction code bit to each data bit when it's sent to memory. When the CPU calls for data, the memory controller decodes each error correction bit and determines the validity of its attached data bit.

The system requires twice the number of bits, but the benefit is that ECC can correct a single-bit error. Because approximately 90 percent of data errors are single-bit errors, ECC does a very good job. On the other hand, ECC costs a lot more, due to the additional number of bits.

Caution - Remember that ECC can correct only single-bit errors, but it can also detect multi-bit errors. Parity checking understands only that the overall byte coming out of memory doesn't match what was sent into memory. Parity checking cannot correct anything.

Usually, whoever is buying the computer will decide which type of data integrity checking they want, depending mainly on cost benefits. They can choose ECC, parity checking, or nothing. High-end computers (e.g., file servers) typically use an ECC-capable memory controller. Midrange desktop business computers typically are configured with parity checking. Low-cost home computers often have nonparity memory (no parity checking or "fake" parity).

Practice Questions

Question 1
Part of a computer’s RAM chip is dedicated to storing key system settings required for boot-up.

a. True

b. False

Answer b, false, is correct. Random access memory (RAM) is volatile and loses all of its data without a source of power. RAM comes in modules, and is almost never referred to as a chip. System boards commonly use nonvolatile CMOS to store system settings. CMOS memory uses very little current (a trickle charge) and continues to be powered for extended periods of inactivity by a small battery on the system board.

**Question 2**

Over-clocking allows a microprocessor to run considerably faster than motherboard components. What type of memory structure was developed to minimize the delay of accessing RAM on the motherboard?

a. Processor resident pipeline

b. L-2 cache

c. CMOS memory

d. Duplex memory

Answer b is correct. Intel 486 and Pentium processors have a small amount of memory integrated in the chip called an L-1 cache. However, as processor speeds increased, additional high-speed memory was needed. This second block of memory was called an L-2 cache and was located on a special high-speed bus. Later, some designs included an L-2 cache on the processor cartridge or the die itself. Duplex memory and a processor resident pipeline do not exist.

**Question 3**

DIMMs and SIMMs are interchangeable, provided speed and capacity requirements are observed.

a. True

b. False

Answer b, false, is correct. SIMMs and DIMMs look similar, and both use edge connectors. However, DIMMs use both sides of the connector to support a 64-bit or
wider memory bus, and they have two separate connector pins, one on each side of the module board.

**Question 4**

Parity chips on SIMMs no longer provide a useful purpose and have been largely removed.

a. True

b. False

Answer b, false, is correct. Parity chips allow memory to be tested during the POST, and they also monitor memory during computer operations. Some manufacturers have eliminated them or bypassed their function to cut costs. This allows less expensive SIMMs to be used, but at the expense of reliability.

**Question 5**

Which of the following choices best describes what is meant by cache memory?

a. A place where instructions are stored about the operations of a device or application

b. Extended memory that can be made accessible with the SMARTDRV /ON command

c. Memory that holds applications and data that the CPU isn’t running

d. Memory that holds data that the CPU will search first

Answer d is correct. The CPU will look in cache memory first. If it fails to find the necessary data, it will look in main memory. If it fails to find what it needs in main memory, the CPU will look on the disk.

**Question 6**

A 64-bit EEPROM is an inline memory module with 32 connecting pins located on either side of the memory bank.

a. True

b. False

Answer B, false, is correct. EEPROM refers to programmable read-only memory chip
that is used to store the BIOS. This chip is not a part of the main memory, and it is not a "module."

**Need to Know More?**


Mueller, Scott. *Upgrading and Repairing PCs, 12th Edition*. Indianapolis, IN: Que, 2000. ISBN 0-7897-2303-4. This is one of our favorites. If you are only going to have one reference book, give this one serious consideration.


[http://www.aceshardware.com](http://www.aceshardware.com)

Ace’s Hardware—Technical Information for the Masters and the Novices
A comprehensive site covering everything you would ever want to know about memory.
Brian Neal - Publisher, Webmaster
All Content is Copyright (c) 1998-2002 Ace's Hardware. All Rights Reserved.

© Copyright Pearson Education. All rights reserved.