RAM vs. Hard Drive Memory

Both RAM and hard drive memory are referred to as memory, which often causes confusion. RAM stands for Random Access Memory. Physically, it is a series of chips in your computer. When your computer is turned on, it loads data into RAM. Programs that are currently running, and open files, are stored in RAM; anything you are using is running in RAM somewhere. As soon as the electricity to the RAM is cut, it forgets everything; that's why an unsaved document is lost if the computer locks up or there is a power failure. When you save a document, it goes on a hard drive, or another type of media storage device. Typically, this type of storage is magnetic, and does not depend on electricity to remember what is written on it. However, it's much slower than RAM. The computer can access anything stored in RAM nearly instantly. Things on the hard drive need to be located, read and sent to RAM before they can be processed. If your computer says you are low on disk space you have too many programs or files on your computer. To correct this, you will need a new hard drive, or will need to uninstall unused programs or delete unneeded files off the computer. If your computer says you are low on memory, you have too many programs running, or your computer does not have the RAM needed to run the software you want to. Restarting your computer will clear the RAM and usually clears up 90% of low memory errors. If your computer has more RAM than the programs minimum requirements, and restarting the computer does not solve the problem, the error could be caused by a buffer overflow, or another technical issue that is outside the scope of this tutorial.

ROM and Firmware:

Programs stored in ROM are called Firmware.
Firmware is held in non-volatile memory devices such as ROM, EPROM, or flash memory. Changing the firmware of a device may rarely or never be done during its lifetime; some firmware memory devices are permanently installed and cannot be changed after manufacture. Common reasons for updating firmware include fixing bugs or adding features to the device. This may require ROM integrated circuits to be physically replaced, or flash memory to be reprogrammed through a special procedure. Firmware such as the ROM BIOS of a personal computer may contain only elementary basic functions of a device and may only provide services to higher-level software. Firmware such as the program of an embedded system may be the only program that will run on the system and provide all of its functions.
ROM stands for read-only memory. It's used to store the start-up instructions for a computer, also known as the firmware. Most modern computers use flash-based ROM. It is part of the BIOS chip, which is located on the motherboard.
In computing, firmware is a computer program that is "embedded" in a hardware device and is an essential part of the hardware. It is sometimes called embedded software. An example is a microcontroller, a part of the microprocessor that tells the microprocessor what actions to take.

BIOS

Basic Input/Output System and also known as the **System BIOS**, **ROM BIOS** or **PC BIOS**) is firmware used to perform hardware initialization during the booting process (power-on startup), and to provide runtime services for operating systems and programs.

The BIOS software has a number of different roles, but its most important role is to load the operating system. When you turn on your computer and the microprocessor try to execute its first instruction, it has to get that instruction from somewhere. It cannot get it from the operating system because the operating system is located on a hard disk, and the microprocessor cannot get to it without some instructions that tell it how. The BIOS provides those **instructions**. Some of the other common tasks that the BIOS performs include:

- A power-on self-test (POST) for all of the different hardware components in the system to make sure everything is working properly
- Activating other BIOS chips on different cards installed in the computer - For example, SCSI and graphics cards often have their own BIOS chips.
- Providing a set of low-level routines that the operating system uses to interface to different hardware devices - It is these routines that give the BIOS its name. They manage things like the keyboard, the screen, and the serial and parallel ports, especially when the computer is booting.
- Managing a collection of settings for the hard disks, clock, etc.

The BIOS is special software that interfaces the major hardware components of your computer with the operating system. It is usually stored on a Flash memory chip on the motherboard, but sometimes the chip is another type of ROM.

When you turn on your computer, the BIOS does several things. This is its usual sequence:

1. Check the CMOS Setup for custom settings
2. Load the interrupt handlers and device drivers
3. Initialize registers and power management
4. Perform the power-on self-test (POST)
5. Display system settings
6. Determine which devices are bootable
7. Initiate the bootstrap sequence

The first thing the BIOS does is check the information stored in a tiny (64 bytes) amount of RAM located on a **complementary metal oxide semiconductor** (CMOS) chip. The CMOS Setup provides detailed information particular to your system and can be altered as your system

changes. The BIOS uses this information to modify or supplement its default programming as needed. We will talk more about these settings later.

Interrupt handlers are small pieces of software that act as translators between the hardware components and the operating system. For example, when you press a key on your keyboard, the signal is sent to the keyboard interrupt handler, which tells the CPU what it is and passes it on to the operating system. The **device drivers** are other pieces of software that identify the base hardware components such as keyboard, mouse, hard drive and floppy drive. Since the BIOS is constantly intercepting signals to and from the hardware, it is usually copied, or **shadowed**, into RAM to run faster.

Driver Circuit:

A driver circuit allows a controller of some kind to operate a device that requires different or more powerful control signals than the controller provides.

Take for example an Arduino processor. The pins are capable of putting out a few volts amd a few milli Amps. You can connect an LED with a series resistor directly to the pin, and with the arduino software turn the LED on and off. But for if example you want to control a robot's drive motor, the pins of the Arduino do not have enough power to make the motor turn if connected directly; you need a driver circuit.

In this case, the driver circuit will accept the signals from the Arduino, and create the signals required by the motor. The circuit will likely also require a power supply of its own with enough voltage and current to turn the motor. The driver circuit would also include the capability of adjusting the speed and direction of the motor based on commands from the Arduino.

A device driver is a special kind of software program that controls a specific hardware device attached to a computer. Device drivers are essential for a computer to work properly. These programs may be compact, but they provide the all-important means for a computer to interact with hardware, for everything from mouse, keyboard and display (user input/output) to working with networks, storage and graphics.

How Do Device Drivers Work?

Device drivers generally run at a high level of privilege within the operating system runtime environment. Some device drivers, in fact, may be linked directly to the operating system kernel,

a portion of an OS such as Windows, Linux or Mac OS, that remains memory resident and handles execution for all other code, including device drivers. Device drivers relay requests for device access and actions from the operating system and its active applications to their respective hardware devices. They also deliver outputs or status/messages from the hardware devices to the operating system (and thence, to applications).