Firmware and Middleware are simple industry terms to describe a certain set of 'some code'.

Firmware (as wiki states) 'usually' represents software 'usually' written in assembly or C that 'usually' runs **on/the** hardware.

The 'on/the' is to indicate that this software resides on a permanent storage medium (like some sort of small flash) that is directly connected to the hardware the software is to control. It's called 'firmware' because it is software specifically designed to run the hardware.

I also quote the word usually because the tech industry has a horrible habit of coining terms and then warping them so that no one can know what the true intention of the word is. 'Embedded programming' is a good current example; while the term typically referred to programmers who developed what is essentially firmware (i.e. low-level ASM/C hardware developers), the term has since morphed to include Android developers (i.e. Linux/kernel developers who can operate at both the Java and C level). I'm not bashing any sort of developer or development practice (I do a lot of those types of developing myself), merely pointing out that the tech industry likes to misuse words.

Middleware is another example of such a word; while I've yet to hear or see 'firmware' used for anything but hardware level code, I've seen middleware used on everything from firmware to .NET/Java. Middleware is a generic term that (as a developer) you usually can decipher from context what is intended as I've seen it misused too many times; so to answer you directly:

**I still don't understand the differences in their roles, relative to hardware, high-level software, operating system, etc.** Firmware is software (code turned to binary format) that resides on a certain piece of hardware and runs said hardware. Middleware can be used to mean an API/library that interacts with hardware (or another piece of software for that matter).

**Is BIOS firmware not middleware?** Depending on your view of this, it could be both. Technically the BIOS is firmware (it's software that is specifically used to run the hardware it's on, i.e. the motherboard), but if your intent is to interact directly with a piece of hardware in some fashion (as an OS would), the BIOS 'could' be middleware, as you don't have to 'write' your own BIOS and hardware routines and this is where the misuse of 'middleware' can come in, but in a classical since, no the BIOS is not middleware.

**Is a bootloader for an OS a firmware, middleware or someware?** A bootloader for an OS is a piece of code that resides between the BIOS (or other basic hardware subsystem) and the higher level systems (usually the OS) that tells the lower systems (i.e. BIOS) where the OS resides on the boot medium. Typically speaking, a bootloader usually sits on a specific location of a storage medium (like the first few hundred bytes or more) and the BIOS knows (thanks to industry standards) where to 'look' for a bootloader at which point the bootloader takes over to do what it was written to do (usually boot the OS).

A bootloader is not firmware but could technically reside in firmware and as far as it being 'middleware', that could be yes or no depending on 'your' view of what 'middleware' is/should be, though in the classical since of middleware, I would not consider a bootloader middleware as it doesn't provide me any easier context (unless I'm writing an OS).

**Is the instruction set of a CPU firmware and not middleware?** An instruction set of a CPU would be considered more of an application programming interface (API) (which 'could' be considered a form of middleware), it is not firmware as the CPU (the central processing unit itself) doesn't have anything 'to run'; the CPU instruction set is what the software gets 'compiled' to (assembly) that the CPU then 'understands' how to run.

**Is there some middleware not firmware?** Yes; just about any software framework you can think of (the .NET or Java libraries for example) could be considered a form of middleware as they give a programmer an 'easier' way to interact with various aspects of a computer. Using Java, for instance, you could open a file and write to it and have that code 'work' on any Java supported system. Since certain Java API's let you have a few lines of code that interact with files across disparate systems, it can be considered a form of 'middleware' since the developer doesn't have to write the code for each different system (Linux/Windows/Apple,etc) to interact with the filesystem of the hard drive and OS him/herself.

**What are the relations and differences between device drivers and firmware and middleware?** A device driver is a piece of software that sits at the OS level (usually as an 'installed' library) that tells the OS 'how' to interact with said device. For instance, when you install the latest video drivers, you are installing software that the OS 'uses' to communicate with the actual video card itself. The video card itself has firmware on it that knows how to interpret the information given it to by the OS (because of the driver) and does what it will with it (draw a window or game sprite for instance).

Device drivers could be considered middleware (again depending on your view of such) since it sits between the OS/hardware and anyone wanting to use the hardware through the OS.

Anecdotal: in my industry experiences, I have found 'middleware' to 'mean' 'web services' or something of the like (at least what the person using the term is meaning anyways), though I have also seen/heard it (mis)used in many job postings and interviews to mean a variety of things from API's and libraries (like Boost or Spring) to the C++ STL and even C# itself (the language itself, not .NET).

I hope that can help.

[DIfferences between firmware and middleware and device driver? - Super User](https://superuser.com/questions/707405/differences-between-firmware-and-middleware-and-device-driver)

BIOS (basic input/output system) is the program a computer's [microprocessor](https://www.techtarget.com/whatis/definition/microprocessor-logic-chip) uses to start the computer system after it is powered on. It also manages data flow between the computer's operating system (OS) and attached devices, such as the hard disk, video adapter, keyboard, mouse and printer.

### History of BIOS

The term BIOS was first coined in 1975 by American computer scientist Gary Kildall. It was incorporated into IBM's first personal computer in 1981 and, in the years to come, gained popularity within other PCs, becoming an integral part of computers for some time. However, BIOS' popularity has waned in favor of a newer technology: Unified Extensible Firmware Interface ([UEFI](https://www.techtarget.com/whatis/definition/Unified-Extensible-Firmware-Interface-UEFI)). Intel announced a plan in 2017 to retire support for legacy BIOS systems by 2020, replacing them with UEFI.

### Uses of BIOS

The main use of BIOS is to act as a middleman between OSes and the hardware they run on. BIOS is theoretically always the intermediary between the microprocessor and I/O device control information and data flow. Although, in some cases, BIOS can arrange for data to flow directly to memory from devices, such as video cards, that require faster data flow to be effective.

### How does BIOS work?

BIOS comes included with computers, as [firmware](https://www.techtarget.com/whatis/definition/firmware) on a chip on the [motherboard](https://www.techtarget.com/whatis/definition/motherboard). In contrast, an OS like Windows or iOS can either be pre-installed by the manufacturer or vendor or installed by the user. BIOS is a program that is made accessible to the microprocessor on an erasable programmable read-only memory (EPROM) chip. When users turn on their computer, the microprocessor passes control to the BIOS program, which is always located at the same place on EPROM.

When BIOS boots up a computer, it first determines whether all of the necessary attachments are in place and operational. Any piece of hardware containing files the computer needs to start is called a boot device. After testing and ensuring boot devices are functioning, BIOS loads the OS -- or key parts of it -- into the computer's random access memory ([RAM](https://www.techtarget.com/searchstorage/definition/RAM-random-access-memory)) from a hard disk or diskette drive (the boot device).

### The 4 functions of BIOS

BIOS identifies, configures, tests and connects computer hardware to the OS immediately after a computer is turned on. The combination of these steps is called the boot process.

These tasks are each carried out by BIOS' four main functions:

1. **Power-on self-test (**[**POST**](https://www.techtarget.com/whatis/definition/POST-Power-On-Self-Test)**).** This tests the hardware of the computer before loading the OS.
2. [**Bootstrap loader**](https://www.techtarget.com/searchdatacenter/definition/boot-loader-boot-manager)**.** This locates the OS.
3. **Software/drivers.** This locates the software and drivers that interface with the OS once running.
4. **Complementary metal-oxide semiconductor (**[**CMOS**](https://www.techtarget.com/whatis/definition/CMOS-complementary-metal-oxide-semiconductor)**) setup.** This is a configuration program that enable users to alter hardware and system settings. CMOS is the name of BIOS' non-volatile memory.

### Accessing BIOS

With BIOS, the OS and its applications are freed from having to understand exact details, such as computer hardware addresses, about the attached I/O devices. When device details change, only the BIOS program needs to be changed. Sometimes, this change can be made during system setup.

Users can access BIOS and configure it through BIOS Setup Utility. Accessing BIOS Setup Utility varies somewhat depending on the computer being used. However, the following steps generally enable users to access and configure BIOS through Setup Utility:

* Reset or power off the computer.
* When the computer turns back on, look for a message that says "entering setup" or something similar. Accompanying that message will be a key that the user should press to enter system configuration. Here's an example message a user might see: "Press [key] to enter BIOS setup." Some keys often used as prompts are Del, Tab, Esc and any of the function keys (F1-F12).
* Upon seeing the prompt, quickly press the key specified.

Once in BIOS Setup Utility, users can change hardware settings, manage memory settings, change the boot order or boot device, and reset the BIOS password, among other configuration tasks.

### BIOS security

BIOS security is a somewhat overlooked component of cybersecurity; however, it should still be managed to prevent hackers from executing malicious code on the OS. Security group Cylance, in 2017, showed how modern BIOS security flaws could enable ransomware programs inside a motherboard's UEFI and exploit other PC BIOS vulnerabilities.

Another unique exploit involving the manipulation of BIOS was [Plundervolt](https://www.techtarget.com/searchsecurity/definition/Plundervolt). Plundervolt could be used to mess with a computer's power supply at the time data was being written to memory, causing errors that lead to security gaps. Intel released a BIOS patch to defend against it.

### BIOS manufacturers

BIOS, in its beginnings, was originally owned by IBM. However, some companies, such as Phoenix Technologies, have reverse-engineered IBM's original version to create their own. Phoenix, in doing this, allowed other companies to create clones of the IBM PC and, more importantly, create non-IBM computers that work with BIOS. One company that did this was Compaq.

Today, many manufacturers produce motherboards with BIOS chips in them. Some examples are the following:

* AMI
* Asus
* Foxconn
* Hewlett Packard (HP)
* Ricoh

Knowing the motherboard manufacturer is important because users may want to update their BIOS and chipset drivers -- the drivers that enable the OS to work with other devices in the computer, such as a video card -- to the most recent versions. Driver updates may improve computer performance or patch recent BIOS-level security vulnerabilities. Each manufacturer has a unique way of updating these drivers.

This was last updated in November 2020

[What is BIOS (Basic Input/Output System)? (techtarget.com)](https://www.techtarget.com/whatis/definition/BIOS-basic-input-output-system#:~:text=BIOS%20(basic%20input%2Foutput%20system)%20is%20the%20program%20a,%2C%20keyboard%2C%20mouse%20and%20printer.)

I am learning about operating systems but there is a small concept I cannot grasp. Say a process 1 is running on the CPU and then it issues an I/O request to read from a disk. For efficiency, the CPU begins executing process 2 as this request is handled. That all makes sense but doesn't the I/O need to use the CPU?

enter image description here

**My Question:** Why isn't the CPU needed to service process 1's request?

It would help to understand the role of 3 important aspects of I/O in computer architecture: Interrupts, DMA, and Hardware Controllers.

When the CPU issues an I/O request to the hard disk, the hard disk has its own specialized chip called a device (or hardware) controller designed solely for processing commands from the CPU, such as reading from the disk. Originally these were simple chips that performed specific operations for the CPU, but modern hardware controllers are basically their own microprocessors with firmware and everything, so they are capable of very complex operations without the main CPU's help. While the hard drive's controller is busy performing the request, the main CPU is free to do whatever it wishes, such as execute process 2 in your example. The controller is able to read and write directly to and from system RAM using what is called a Direct Memory Access (DMA) controller, a special unit that transfers data from the hardware controller to main RAM without the CPU needing to do anything.

When the hard drive is done with the request and the relevant data has been loaded into RAM through DMA, it issues an interrupt request which informs the CPU that the data has been loaded into RAM. At this point the CPU can transfer control back to process 1. Thus, the CPU does not need to micromanage all tasks involved with I/O. At one time this used to be the case, but these tricks (interrupts, DMA, special controllers) were invented in order to improve CPU performance and make things more efficient.

[io - Why is the CPU not needed to service I/O requests? - Stack Overflow](https://stackoverflow.com/questions/13596997/why-is-the-cpu-not-needed-to-service-i-o-requests?rq=1)

## **What Does Zombie Process Mean?**

A zombie process is a process in its terminated state. This usually happens in a program that has parent-child functions. After a child function has finished execution, it sends an exit status to its parent function. Until the parent function receives and acknowledges the message, the child function remains in a “zombie” state, meaning it has executed but not exited.

A zombie process is also known as a defunct process.

[What is a Zombie Process? - Definition from Techopedia](https://www.techopedia.com/definition/18788/zombie-process)