What is a computer?

- Everybody uses terms like CPU, RAM, hard drives. We've heard these terms, but in this episode, I wanna talk about what really is a computer? Now, in order to have a computer, remember why we have them in the first place, it's to get stuff done. We do things like generate spreadsheets, or edit videos or control traffic or play games. No matter what it is, the whole function of a computer is to get something done, either work or play. So, the cornerstone of any computer you're gonna have out there are the CPUs. So, here's a couple of CPUs I've got laying in here. Now, a lot of times, we'll say a CPU is like the brain of the computer, and it's really more like a really super high-speed calculator. And what we do is we write programs that the processor can then do stuff with to manipulate peripherals and to create these things called files to get stuff done. So, we have

these huge things called programs. Now, programs can be applications, like say Microsoft Word or Edge Browser, something like that, or it could be a utility that helps us get on a network, but they're all programs. So, when your computer's turned off, those programs have to sit someplace, and that's what these guys are all about. This is mass storage. So, in this particular case, this is called an M.2 SSD, solid state drive, and it stores stuff statically, and this is a older-school, but still very popular hard drive. It's got spinning platters in there. The only thing these guys do is they store data, whether that data is a program or a Microsoft Word document when you're not using it. When you're on your computer and doing whatever you're doing, and suddenly you double-click on an icon to say, start up Microsoft Word, a lot of things start to take place. First of all, your computer, the CPU knows how to talk to the hard drive. It'll say, "Go get that Microsoft Word." Now, hard drives are

fast, but the problem is is they're not nearly as fast as the CPU. So, while we are, ready, running a program, what we're really doing is we're taking a copy of that program off of the mass storage, and we're putting it onto our RAM, our Random Access Memory. So, when a program is running, it is stored in RAM. RAM is super fast, not quite as fast as a CPU, but it's very, very fast. The only downside is, is compared to a hard drive, RAM is very, very expensive, so we only use RAM when we're running a program. So, what will happen is the CPU will then copy, let's say Microsoft Word, copies the Microsoft Word program onto RAM, and then it begins running the Word program. Now, the Word program basically sits there with a nice white screen waiting for you to press buttons on your keyboard. "Hi, mom, I am fine. Love Mike." So, the program itself is monitoring peripherals. In this particular case, it's monitoring the keyboard. And as each letter gets pressed, it starts showing up on the

screen, and you write yourself a letter. And then you go through a process of saving your data, saving a file. So, what'll happen is then the CPU will then instruct the data that you've written on RAM, it doesn't go to the hard drive automatically. The data you've written on RAM, you hit File, Save, and then that's copied also to your mass storage. So, mass storage stores two very different things. Number one, it's storing the programs that aren't in use, and number two, it stores our data. Our pictures, our Word documents, our saved games, whatever it might be. So, we go through this process back and forth and back and forth. In order to be a computer, you have to have a processor, you have to have RAM, and you have to have mass storage. There's no exception to that. The other thing, and we'll cover this in another episode, is you need some kind of program that is the king of all programs, and we call these operating systems. So, even though in this series, I use desktop computers a lot, big

desktops, the reason I use them is because, well, they're big, and it's easy for me to show you RAM and a CPU and a hard drive and stuff like that. But what I need you to get in mind right away is that computers aren't just desktops. Take a look over here. This is a perfectly good little laptop computer. Inside this laptop is a CPU. Inside this laptop is mass storage. Inside this laptop is lots and lots of RAM, and video and a keyboard and all kinds of other stuff too. This is a computer. (laptop plops) Here's my little Apple Tablet. Inside this Apple Tablet is a CPU. Inside this tablet is RAM. Inside this tablet is mass storage. There's even an operating system, all built in to this one little tablet. But don't let it stop there. Here's an Android phone. This Android phone has a CPU, it has RAM, it has mass storage, it has an operating system. It's all built into that. I mean, let's go crazy here. You ever heard the term Internet of Things? Here's a little device called a Raspberry Pi. This is a little, tiny, tiny computer, but

heck, this thing's even got USB ports and a network connection, and all that type of stuff. On this particular guy, we've got CPU, RAM, mass storage, everything. And in fact, on this guy, it's all actually put onto one individual chip, and we use the term System on a Chip to define that type of thing. Computers are everywhere. The washer and dryer in my house as a full computer system. I could actually turn on the dryer using my phone. You use cameras or anything like that? This camera has CPU, RAM, operating system, Everything is all in this camera. How about the routers or the wireless access points that we use in our home? These guys are also complete computers. Inside this computer is going to be a CPU, RAM, storage, an operating system, everything that we need to be a complete computer. So, be careful when you use the word computer. The computers have taken over, happily, 'cause it sure makes our lives convenient, and everything from the big desktops, which we'll

be concentrating on in this series, down to the smallest little camera, or in my case, washer and dryer, is all a computer. All you need is CPU, RAM, mass storage, and so

What is a CPU?

- Odds are good you've talked about CPUs casually like, I've got an Intel CPU or I've got an AMD CPU. In this episode, I want to answer one simple question, what is a CPU? Now, I've got a CPU in my hand right here, and what I want to do is just kind of give you a, these are just, I'm a CPU collector. I literally have hundreds and hundreds of CPUs. So I just thought I'd throw out a few. Some of these CPUs are close to 30 years old. Some of these are quite new. And just give you an idea of what CPUs physically look like. Underneath, they've got little pins or wires, sometimes they've got little pads, and this is the brain of

the computer itself. But I hate the word brain. To me, a CPU, and by the way, it stands for Central Processing Unit, the CPU is more like a really, really incredibly powerful calculator. This calculator can calculate billions of different calculations per second. And the secret is, is we have to learn to talk to this guy in such a way that we can actually make him do things like make games with things blowing up and make spreadsheets appear and make websites do whatever they do. So in order to understand what a CPU is, I want to kind of go with an analogy. An analogy that I call a man in a box. Inside this box is this dude. And this guy can add, subtract, multiply, divide, like incredibly fast, but the problem is he's physically inside the box. There's no way we can get to him. This box is made out of 18 feet of concrete or whatever you need to do to understand that we can't like tap on the box or anything to talk to this guy. So what I want to do, first of all, is come up with a way for us to

communicate with this guy, which I'm going to call the external data bus. And it looks something like this. So here's eight light bulbs. Now, it's not just eight light bulbs. In fact, we're going to make 16 light bulbs and this is going to be the wall of the box, okay? So there's going to be eight light bulbs that are visible inside the box and eight light bulbs that are visible outside the box. In fact, we're going to put a light switch underneath each one of these pairs and we're even going to put a light switch inside. Now with this setup, here's what we can do. If I flip on, say this first switch right here, it's going to turn on the light bulb outside the box and it's also going to turn on the light bulb inside the box and through some magic electricity, it'll even flip the switch up, okay? And so by flipping one switch, we turn on these two lights and even flips the switch up. And let's just say we do this for light number four. If I flip this up, this light is on, this light is on and it even flips the switch up. Now, before we

continue this analogy, there's one thing I'd like to make sure it's okay with you. Instead of saying on and off for these light bulbs all the time, would it be okay if I just, instead of saying on, can I just say the number one, and instead the light bulb being off, can I just say the number zero? Yep, that's binary. Let's go ahead and continue the analogy. And this time, we're going to be using ones and zeros. So if I turn on this first light bulb and the fourth light bulb, I have created a pattern of on and off lights and I've created a set of eight different values. Each one of these values, by the way, is called a binary value 'cause it can only be zero or one. And eight of these across is what we call a byte. So each one of these is a binary value called a bit, and then eight of them across is a byte. So eight bits is equal to a byte. So what I've done here is I've created a pattern, remember, 'cause the same lights are turned on and off inside the box. So what I've done is assuming that the guy inside the box knows what these different

patterns mean, I can actually communicate with him. Sounds good? Well, I agree, but what we're going to have to do first is we're going to have to generate a code book 'cause there's a whole bunch of different patterns in there. And this code book has to say, what does each different pattern of on and off lights mean of these eight bit patterns? So what we're going to do is generate this code book. So this code book is built into every CPU on Earth. In fact, it's called the machine language of the CPU and people who can write different patterns of ones and zeros to get the CPU to do different things are known as programmers. So let's go ahead and go at that analogy. Let's keep going a little bit deeper as we get to CPU running. So what I'm going to do is I'm going to hand this guy a code book. Now he's got the code book. But before we close the lid on this guy, there's one more thing I want to do. What I'm going to do is I'm going to put in four racks of light bulbs. And each one of these

racks of light bulbs are going to have 16 light bulbs on. So we have 16 light bulbs. The function of these light bulbs are going to be storage areas for the guy inside the box to be able to do adding, subtracting, multiplying, dividing. And by the way, we call these registers, okay? And for just this particular example, let's label them. We're going to call this one the AX register, this one the BX register, this one the CX register, and this one is the DX register. Every CPU on Earth has registers like these. In fact, in this particular example, I'm only showing you four different registers, which are known generically as general purpose registers, which means that the guy in the box can use them for anything he wants. On a more modern CPU, you might find 100 different registers, that could be as much as 128 bits across. So we're keeping it simple for you to understand how all this works. Now that we understand that the guy has a code book and these registers, what I want to do right now is pop up an example code

book. Now, the code book I'm about to show you is not real, okay? It's based on one of the oldest CPUs out there that was used on the first PCs called the Intel 8088 processor. So while I'm having a little bit of a creative license in terms of exactly how this works, conceptually, this is exactly how they work. So let's take a look at this example code book. So if we take a look on this code book, it's going to say, this first pattern says, I'm going to put a number up next, put that in the AX register. Now here's a command that says, I'm going to put a number in next, put that in the BX register. And here's another command that says, add together what's ever in the AX and the BX register together and put the result in the AX register. And then last is whatever's on the AX register, put it out on the external data bus so I can see it. So in essence, what we're going to do right now is we're going to add two plus three. And that's really all it boils down to. Now, before we close the lid on this guy, I'm going to

throw one more thing on there. And what we're going to do is we're going to put like a bell inside. You ever gone into a store and you open the door and you hear, ding, ding, ding? Okay, so this is kind of like a bell we're going to put inside the CPU, which we call the clock. Now, the clock's job is to tell the CPU to do something. So with the code book, the four registers, and the clock, we're now going to add two plus three. You guys ready? Woo-hoo, here we go. Okay, now the first thing I'm going to do is I'm going to put in the code in binary that says I'm about to give you a number, whatever the number is next, put it on the AX register. So we're going to flip the wires to get that pattern based on the code book. Got it? Now, the next thing we're going to do is we're going to actually ring the bell. So really what we should do is put like a doorbell on the side and that's going to tell the CPU that he needs to do his thing. So let's make that clock look like a little doorbell. It's actually just a

wire. So we go and put that in and we press the button. Now, the CPU is going to see that, and what he's going to do is he's going to read that and he's going to be ready. So now what I'm going to do, oh, and by the way, when he does this, so that we know that he's ready for the next command, he'll zero all this out. We're not touching anything. We're just hitting that clock wire. Then all of a sudden it goes to zeros. That's telling us the CPU is ready for the next command. So the next command is we're going to put, we're going to add two plus three. So we're going to do the number two in binary. So on this part, you're going to have to do a little bit of a trust me. So that's zero, that's one, that's two. So now what we're going to do is take this value, light up the proper wires and we hit that clock button. Now, the thing you got to be careful here is CPUs aren't like human beings. A lot of times, we're going to have to hit that clock button a few times for it to go ahead and zero everything out. So there's not a perfect one-

to-one correlation of every time I press the doorbell, something happens. It depends on how complicated the command is. In today's modern processors, there are very, very complicated commands that would require me to hit that CPU clock about, I don't know, 100 or 200 times before the CPU actually processes it. So keep that in mind as we move forward. All right, let's keep going. So let's refer to the code book again. And the next command is we got to flip these different wires on and off to put the number three into the BX register. So let's go ahead and get that ready. Press that clock button until it zeros out. That means the CPU's ready for the next value. So let's go ahead and do three in binary. So we'll punch that in, let's hit the clock wire and now all of a sudden it zeros out and we can't see this, but right now, a binary value for the number two is in the AX register and a binary value for the number three is in the BX register. So let's add 'em up. So let's refer to the code book. Here's a pattern of

wires that says add what's ever in the AX register to the BX register and put the result on the AX register. So yes, it's going to wipe out whatever's in the AX register, that happens. So we put that pattern in, we hit the clock wire a few times, boom, it has zeroed out, which now means we can't see it 'cause we can't see inside the box, but now the value for a binary five is now in the AX register. So here's another command on the code book that says whatever is in the AX register, put it on the external data bus so we can see it. So let's go ahead and put that pattern in. And we hit the clock wire and hit the clock wire. Except this time, it doesn't zero out. This time, it now shows us a binary five on the external data bus. So congratulations, you've just added two plus three on a CPU using machine language. So this is really all that the programmer is. A programmer is a person who can use machine language to talk to a CPU to get work done. In this case, we're just adding two plus three, but we can do a lot

more with more modern CPUs. So that sounds great, but I'm going to make it a little bit more complicated or actually more realistic. What we're going to be doing now is that it's really not a single entity inside the CPU. Inside the CPU, you have lots of what we call logic units that are actually doing the math. And some are really good at integer math and some are really good at moving data around. So you have all these different functions inside a CPU. Now, have you ever done your laundry? Now if you do your laundry, do you sit there and go, so I take all my laundry and I go through this big pile of laundry, I pull out all my towels and then I throw my towels in the washer and then I dry 'em and then I fold them and then I grab another load and then I put... No, you don't do that, do you? What you do is you get one load running and as soon as you get that out of the washer, then you go ahead and put that in the dryer. And then while it's drying, you get another load. So you have lots of stuff

running simultaneously. In the CPU world, we call this a pipeline. So let's take a look at a pipeline. Except this time, we won't have a man in a box. We're going to have a bunch of guys in the box. Let me show you how this works. So on the simplest CPU, we're going to have four guys. And these four guys are in front of a conveyor belt. So each time a command comes in from the external data bus, the first guy here, his job is to look at it and decide which one of these guys to give it to because this guy right here, he's really good at integer math. This guy right here, he's really good at, really, well, we'll just call it long numbers to make it easier for right now. And this guy right here, he's really good at holding onto information or to send that information onto other places. It could be registers or it could be outside of the CPU into the stuff called RAM. So I'm just going to call him the cache guy. We'll talk about caching in other episodes. So this guy who we call the prefetch, he looks at that

command and he's going to look at that command to decide to do with it. So on the next click of the clock, he might send it to the integer guy, whereas another command's already come in to handle it for him. So he might, on the next click of the clock, send it over to the long integer guy. The guy in the integer, he's already done with it 'cause he's already added it, so he moves it over to the cache guy. So what happens is all these different commands are moving along in what we call a pipeline or also known as a core. Now, in my simple example here, the pipeline only has four different discrete steps. On modern CPUs that can be 16 steps, even in some cases up to 30 different steps, depending on the make and model of CPU that you buy. The important thing is, is that we don't care necessarily what happens on the inside. We know that if we write using the proper machine language, as long as they can process that machine language on the inside, whether they have four steps, 16 steps,

or 30 steps in their pipeline, it doesn't make any difference to us. It does make a big difference to companies like Intel and AMD as they build their internal architecture for CPUs because they always want to make the best one. So when we're really talking about CPUs, the one big thing we can do right now to speed things up is to be able to hit the clock faster. So making your clock faster and faster can be good, but only up to a certain point. And keep watching some more of these episodes here and we're going to make the CPU lot more sophisticated and a lot faster too.

Modern CPUs

- Whether you call it a processor, the central processing unit, the brains or heart of a computer, a CPU is the primary component of any computer or smart device. A CPU execute

systems and program instructions such as performing calculations and moving data, and coordinates all the actions of a computer. As an A+ certified technician, you need to know the basic characteristics of modern CPUs and the features and capabilities you should consider when choosing a CPU for a given application. Let's look at some of those features and capabilities to gain an understanding of them so you can make the right CPU choice. Let's begin with the two most common instructions set architectures, ISAs, for CPUs, x86 and x64. An ISA defines the interpretation of the low-level machine code instructions, which in turn defines how the CPU reacts to the instructions and its hardware and software interfaces. The x86 is a complex instruction set computer, CISC architecture. This means the processor has a large complex set of built-in instructions, several of which are multiple cycle operations. The x64 and the AMD64 processors are reduced instruction set

computing, RISC architectures, and, you probably guessed it, has a smaller and simpler set of built-in instruction sets. RISC processors are focused on efficiency and speed, while CISC is more versatile for working with hardware. The x86 ISA has been around since 1978 when Intel released the 8086 and the 8088 processors. It began as a 16-bit processor, but was later expanded to 32-bit ISA. The 32-bit design indicates how much data an x86 CPU can produce per clock cycle. Basically, that means a 32-bit CPU can move as much as 32 bits per cycle. The x86 ISA is popular and used in a variety of products, including portable computers, IoT devices, and smartphones even. The x64 ISA is actually based on the x86, but expanded to facilitate 64-bit code and data. Since it was released in 2000, it has evolved into two primary operating modes, 64-bit mode and compatibility mode, which means it also supports 16 and 32-bit software. Because it supports larger addressing, the x64 is able

to store and address larger amounts of data in RAM. The x64 ISA can address up to 16 exabytes of RAM, which is the same as 16 billion gigabytes or approximately 16 quadrillion bytes. Because of its addressability. x64 architecture is commonly used in supercomputers and other high-end systems. In comparison, the x86 ISA only supports four gigabytes of RAM. A couple of things you should know. The x86 was developed by Intel, and the x64 was developed by AMD, and the x64 CPU is also known as the x86_64. The AMD64 and 64 bits, these terms are interchangeable. If you are unsure of which of these two ISAs a computer is using, you can find out in the Windows Settings About App. Okay, here's a summary of the primary differences between the x86 and the x64 ISAs. The x86 supports up to four gigabytes of RAM, and the x64 supports up to 16 exabytes. The x86 uses 32-bit buses, and the x64 uses 64-bit bus. The x86 supports only software written for the 32-bit systems, and the

x64 supports both 32 and 64-bit software. And the x86 is slower than the x64. There is one other CPU architecture that you should know, an Advanced Risk Machine, or ARM. The ARM CPU is based on the RISC architecture and designed for efficiency operations and low power consumption. ARM processors are commonly used in mobile devices like smartphones, IoT devices, and they're actually finding their ways into desktops and servers as well. These, and essentially all other CPUs, are measured by two primary features, cores and clock speed. CPU cores are the result of microprocessing manufacturers figuring out how to put multiple processors on the same chip. So let's take a look on how cores actually affect newer CPUs and let's go ahead and bring out our friend here, the box, and the box can pretty much be anything you want it to be. I'm going to label it CPU, but here you can think of this box as the stadium in your hometown, or the fairgrounds, or any place

that is really busy and that sees a lot of traffic to and from. It'll work great for this, and I'm going to give it something to communicate here with, probably the RAM. Just go ahead and draw a little RAM stick here and there we go. Now, most CPUs are what we know as single-core, so that means that they have one CPU on the system chip, and thus we only have one lane where information can travel back and forth from. Think about this as if you are going to the fairgrounds or the stadium or whatever. If you only have one street to get there, that street's going to get backed up in both ways, right? So instead, we enable in CPUs what's called multi-cores. With multi-cores, we're actually able to give alternate lanes of information where data can flow to and from. If we go from a single core and add one more core to that, we call that a dual-core. So effectively, our traffic, our data going to and from has two brand new lanes to go in. And then if we add another two, because remember,

everything in computing is done in the power of two, and if we add another two lanes of information there, we create what is called a quad-core CPU, and now we have multiple lanes of information that is able to flow freely to and from. CPUs are becoming more advanced, and multiple cores produce increased clock speeds. If you were to take a breath, and immediately exhale, you should take about one second. A CPU speed is clock speed is measured by the number of tasks it can do in one second. How much a CPU can process depends on what it's doing. Typically, a CPU is processing instructions from a variety of applications, services, and device drivers, seemingly at the same time, some of these instructions are relatively simple and some not so simple and may take a bit more processing. This means some instructions may require only a single clock cycle, and others a bit more. A clock cycle is the measurement of a CPU speed represented as the number of clock cycles per

second. One clock cycle in one second is measured as one hertz, after Heinrich Hertz, who discovered that electromagnetic waves exist. Most of today's CPUs operate with gigahertz speeds or billions of cycles per second. So the time it takes you to take a single breath in, a CPU could have processed a billion or more tasks. Wow, right? So CPUs have a variety of architectures, instruction sets and applications, all of which are things that you should know for the A+ certification exam.