Week 1 Module 1 Fundamentals of System Hardware

Introduction 1.2

In this first module we just want to give everybody a start off with a basis on what a computer is and how it works. In the future modules we're going to be doing a lot of programming. We're going to be doing a lot of operating systems concerns.
We're going to be doing a lot of things that you need to know about what goes on inside the computer and so in this module we just like to start you off with the very basis of what's inside the computer and how things work. We don't know what your background is we don't know where you come from or what your history is so if some of this is a little bit basic, then we apologize but for everybody we'd like to start off on the same foot.

Overview 1.3

So just an overview for what we're going to be covering in this module we'll cover the definition of a computer, types of computers, what's inside a computer (just an overview of what's in what the components are), what each component does, the commonalities between the components, how the components communicate with each other, how the CPU works, the idea of a memory hierarchy, hard disks, and some computer networking concerns.

Definition 1.4

To give an easy definition of what a computer is a very basic definition the computer is an electromechanical device which takes input, does processing and produces output. And we know that what we define as a computer today is a heck of a lot more, but this is just a basic definition for what we think should be in a computer in the most basic form: taking input, do some processing, and producing output. It meets the very basis of the definition and it doesn't have a lot of superfluous items that we are not necessarily concerned with.

Types of Computers 1.5

We have a lot of different types of computers in use in the environments today as well as and past history. So we have this concept of a mainframe which is a very, very large computer which used to act for lack of a better term, used to act as a central point for all the computing done on a campus. So for example, one university might have just one mainframe for each campus in the entire university and that's where all the processing was done. Imagine only having one computer for the entire university. It gets very very busy and the time on that computer is very much in demand. And then we move to a more common definition today which is that of a server, and a server would be something that probably exists in a data center or in a room dedicated with air conditioning and power controls. This would not be used

by most people, so it would have a lot of capability a lot of RAM a lot of CPU (we will talk about these definitions later) but the server serves one purpose. And a desktop would be used by an end user at one physical location (obviously) because you don't want to carry a desktop around too much. A laptop we know we just carry it around and use it just as a regular computer. A tablet is pretty much the same as a regular laptop except without a keyboard, and that's basically what we've come to these days. And then we could also add that a portable phone is by definition a computer because it does take input, do some processing on that input (your voice), and then encoding your voice, and then it produces output in the form of a signal it gets transmitted to your cell provider.
So all of these are types of computers and we're going to explore the commonalities between these computers in just a minute.

Inside a Computer 1.6

So in this image, we've got what looks like the inside of a desktop computer and you're seeing hopefully what you would expect to see when you open up a regular desktop computer. In the upper right there we've got the power supply, which has the big label that says Dell on it. And below that is what we call the main board or motherboard of the computer where we have most of the main components related to the computer. Down below it you'll see the CPU. The memory is off to the left. A little bit further than that you have the secondary storage device which in this case looks like a spinning hard drive. All the way up in the upper left we have the tertiary storage device which looks like a CD-ROM and then run about the middle we have the video controller card. So these are the major macro components of what's inside of a computer and each of them communicate with each other through the main board which has components known as the system bus, which we're going to talk about in a little bit.

What's Common Between Them 1.7

So all computers every computer will have at least one central processing unit and the central processing unit is the brain of the computer. That's where all the real work is done. Main memory – main memory is where the code and the data are stored, but main memory has got to be temporary storage because once we turn the computer off all of that is lost. And then because we're going to need some sort of a permanent storage system, there's got to be a secondary storage system where the information is stored permanently. Of course all computers are going to have something that relates all those, so that they can communicate. Most computers are also going to have a video graphics controller where images can be rendered and we can put them on display on a screen.
They'll also probably have a network interface so that we can communicate on the Internet or in the local area network. And then of course they're probably going to have a peripheral interface which is a USB or a Thunderbolt, or a Firewire, or a SCSI. And all of these components meet the requirements so that we can bring data into the computer, we can process that data, and then we can produce output. And if you

look back at each one of these individual items that I've mentioned you're going to see that they either do one thing or two things: input processing or output and that's the basic idea.
Most devices inside the computer are either IO devices or they're processing.

Communications Between the Devices 1.8

So all the communications inside of the machine the communications between the individual components are done via a bus. And a bus is, actually the way that I usually think about it is a big yellow school bus moving from one component to the next and carrying, not children in this case, but information and the information can move either from one component to the other or vice versa.
So it's a bidirectional communications pathway in which we can send information and the bus is actually something that exists physically in the system. So if we put the system under a microscope, we could see the little copper lines that go on the mainboard or on the component to act as the physical bus. So this is something that we can see physically on the system if we probably put it under a microscope because it's rather small, but the system bus is the main pathway between the CPU and main memory.
Now IO devices often have access to the system bus as well for communications both to main memory and to the CPU but the system bus is primarily used priority for CPU and main memory communications back and forth, and that's what has to be done very very quickly inside the system.

The CPU 1.9

So the brain of the computer is what's known as the central processing unit or the CPU and this is where all of the processing, all of the data processing, all the commands that we're going to run inside of a computer. This is where all that is done. It is a single piece of silicon in the form of a chip designed and manufactured by a company like Intel or AMD. or there's a few others that produce these chips, and this is the only location in the entire system where code can actually be run.
Now the code that's going to run here is not the code that we're going to write, at least not in the first parts of this course. The code that we're going to write in the first part of this course is what's known as a high level language and the CPU only runs machine language code. Machine language code is not easy to write and we will do something a little bit like it but it will then convert to machine language code.
In short machine language codes for machines and human code is for… higher level code is for humans. The CPU operates on a fetch, decode, execute cycle which we'll talk about in just a little bit, but it runs this cycle very very quickly and when it's running this cycle each CPU has its own set of instructions which it understands.
So the CPU has a language for itself if you will. Each CPU really only has a small amount of memory, there's not a lot of memory inside the CPU and those are called registers. And the registers are where we store information temporarily, in order to do the immediate processing.

But we only have a few of these registers so we have to be… When we're writing machine language code, we have to be very careful to use them without wasting the memory. And the CPU may have a cache memory which can perform more quickly than main memory, but it's not really much of a concern for us because as programmers we don't have a lot of capability for working with cache memory, in terms of what's inside the CPU.

Machine Language 1.10

Computers only understand very basic commands and I've listed some of the commands here, so you can see them: move, add, subtract, multiply, compare, and jump. You're not seeing any of the higher level commands that you might see in a high level language. So there's really no function calls; there's not a lot of not a lot of commands that we would see except for very basic math operations. There are a lot of machine language instructions, but this set of machine language instructions is limited and we can't really add to it. The machine language instructions are designed physically into the chip.
So the chip is programmed by its manufacture and it's written in silicone in the chip. So we can't expand on that we have no capability to add instructions, but we have the capability to build higher level languages and make the higher level languages convert to the machine language instructions.

Instruction Set 1.11

When the desires of the CPU create the instructions what they're doing is creating what's called a set of instructions or an instruction set and these are the set of instructions that the CPU can perform. And it's usually a very small set just about one hundred instructions or so each one represented by a numeric value and when the CPU receives a particular instruction it performs that task that's associated with the instruction.
Now one instruction might actually have multiple codes associated with it if there are different variables that are associated with that instruction but fundamentally we only have a very small set of instructions and they do limited operations. And we have to work with the higher level languages to bring the higher level languages code down into the machine language code so that it can actually run on the CPU.

Fetch-Execute Cycle 1.12

Probably the most important consideration here is what the CPU does when it has to execute instructions. And it performs what's called the fetch-execute cycle, or the fetch-decode-execute cycle. What it does is move one instruction from main memory, which we haven't talked about but we will in just a minute, it moves it from main memory into a register in the CPU, which is known as the instruction register. And then it decodes the instruction and if necessary moves in any necessary parameters, so it copies those parameters in from main memory. And

once everything is complied in the CPU, once everything is loaded in the CPU, then the CPU can actually execute that instruction and do whatever the task is that is asked. For example, if we had an add instruction which added two numbers together, the fetch would be for a particular memory location. And we would bring that memory location to the register, and we'd add that to a value, maybe that's already in the CPU or maybe its in main memory. Once the operation is complete, then we repeat the process with the next instruction in the sequence. So this operation happens very very very quickly in the system and it has to happen literally millions of times per second in the CPU. Which means each operation could take as little as a bout 10 nanoseconds to perform this operation which is unbelievably small. Nano is one times ten to the negative nine, so we're talking about something happening incredibly quickly inside the system. So while it happens very frequently, literally millions of times a second, it happens very very quickly.

Memory 1.13

The instructions and all the data has to come from somewhere. And in order for the code to be executed it has to be in a register built into the CPU, that one that we spoke about called the instruction register. So why don't we just load all of the code directly into the CPU and then run it directly from there.
The unfortunate fact is that the amount of memory inside of a CPU is really measured in bytes. It's a very small amount of memory. So it doesn't fit a lot of code in fact we really only want to store one instruction directly inside the CPU and the reason that is because each additional byte of memory costs a really significant amount of money to add to each CPU and the manufacturers of these CPU's like Intel and A.M.D. want to keep the costs down as much as possible so we can store everything in registers because it's just too expensive so what we do is we create a memory hierarchy whereby each layer of the hierarchy adds a little bit more space and costs a little bit less but in addition it runs a little bit slower. And we'll see that.

The Hierarchy 1.14

So at the top of the memory hierarchy you can see that we have registers and registers a built regularly into the CPU; there's only a few of them, the size is measured in bytes. But the time to access each of those bytes is measured in nanoseconds and registers are the only place that instructions can be executed.
So we need to use the registers very sparingly. We need to be very cautious about how we use the registers, and in fact in the higher level languages we're not going to concern ourselves with how to use the registers. But when we get down to machine level instructions we have to be cautious about that. The next level of the hierarchy is cache, and in cache we can measure. There's actually two layers of cache but we can measure the total in megabytes. And each of them have nanosecond access times. Although it is slower than a register it can help speed up the processing and the CPU does this automatically in terms of things like pipelining and bringing in instructions before they're really needed. The processor designers take care of the cache and it's not terribly useful for programmers there's not a lot that we can do

inside the cache of a CPU. The next level is probably one of the most important ones that we have to take care of and that's RAM. RAM is measured, of course, in gigabytes, but it does have a lot slower access time; this is over ten nanoseconds access time. So it's a factor of ten slower than the registers.
But it's the only place where we can really store code and store our data. We do have to take into account, as I said before, that RAM is volatile memory. So once the computer is shut down all the contents of RAM are erased completely and that's automatic, it's just by design - the way the RAM is designed. But RAM is incredibly useful because that's where the data is going to come from that's where the code is going to come from to load the code into the registers one by one.
Of course if we have temporary storage, we're going to need some permanent storage because we don't lose everything every time the computers shut down. So the secondary storage devices which have sizes that may be measured in terabytes have much slower access times on the order of ten milliseconds, which if you remember we talked about RAM is ten nanoseconds that's one times ten to the negative nine, ten milliseconds is one times ten to the negative third, so secondary storage is about a million times slower than RAM but it's the only permanent storage device that we have inside the system.
So if we want to save something so that it's not erased when the power is lost on the computer, the secondary storage is where we have to store. Secondary storage is usually in the form of a hard drive or a more modern solid state disk drive, but we use it for the permanent storage of the system.
Thankfully this is where the operating system comes from this is where your code comes from when it starts off and then it's loaded into RAM, of course so that it can execute. But this is where we store just about everything on the system. Tertiary storage is a term that we use for offline storage and these are things like your USB, flash drives, a CD-ROM drives, if anybody still uses those, possibly even tape drives, if we're talking about a server. These have sizes that can be left to your imagination and we can talk about offline storage in the form of petabytes and possibly even exabytes. This is used for backup and it's used for information that doesn't need to be immediately accessible.
You can imagine that the time it takes to load a tape would certainly be significant if we were talking about it in comparison to in ten millisecond access time but that's the idea behind the memory hierarchy as we're working at the top things are faster but significantly smaller and much more costly as we're working at the bottom things are much much slower but more larger and much less cost much cheaper.

Ram 1.15

RAM is another term that we use for main memory and that's because of it's the way it's constructed it's known as random access memory because any place in the memory can be accessed in the same amount of time, which in computer science is known as random time.
So we can access any byte of main memory at the same amount of time. The areas of main memory inside RAM are broken down into bytes and each byte is access independently, so we can access all of the bytes of main memory as individual bytes

or we can group them together in terms of known as a word or D word or cue word double word or quad word.
Usually they're accessed in terms of the data types which we're accessing might have a particular size. So for example in C++ it's very common to have an integer which is a size of four bytes which is equivalent to a D word or double word.
Unfortunately was RAM, as we said when the computers turned off everything in RAM is lost, all of it is completely erased and that's by virtue of its design.
So once the electrical power is shut down to RAM, then the RAM no longer contains any information. When running a program all the machine language instructions are brought into RAM and then there one by one pulled by the CPU during the fetch and execute cycle into that instruction register and then process. So RAM is an absolutely critical portion of the system and we have to know how it works in order to use.

RAM 1.16

As you can see from this, this is what looks like a physical RAM stick or it's known as a dual inline memory module. What we have here is a printed circuit board or a PCB with a number of chips. This one is made by Samsung Corp. Each of the individual chips store a certain amount of RAM. Regardless you can't really peer into it and look at what's going on and as soon as you pull the power out everything's lost anyhow. So the basic idea here is that you have a physical device in which we can temporarily store some information.

Secondary Storage 1.17

So the secondary storage can be broken down into two types, and this is kind of representing a change in the industry as we are today we're moving from hard disk drives to solid state disk drives for a lot of computers. Now hardest drives are not going away by any means. Hard disk drives are also known as spinning drives and the reason for that is that they contain magnetic material which is in the form of a disk and the disk rotates at a constant velocity, and it's measured in a number of revolutions per minute.
And that's published by the manufacturer usually is fifty four hundred RPM or seventy two hundred RPM or possibly even ten thousand RPM. The disk rotates and we have a head that moves in and out along the disk for different radii on the disk and this allows us to define, by polar coordinates, a particular radius with a particular rotation angle and a height (the head) to indicate a specific point on the disk where we want to either read or write. So we can access the entire magnetic material which is where the real storage happens. Unfortunately with this system if we move the head, which has some time that it takes to move, if we move the head to the innermost radius and then do a read operation and then try and move the head to the outermost radius, it really takes a significant amount of time in comparison to move in the head to an adjacent radius. So what we call the each radius is known as a track, and it looks like a track if you were running around in a in a gym. Each track can hold a certain amount of information and the tracks are all

adjacent to one another but moving from one track to the next adjacent track takes a very insignificant amount of time, I shouldn't say insignificant, it still takes an amount of time, but moving from an inner most track to a very far distant track will take quite a significant amount of time. In addition to that we have this rotation that happens and on a drive with, for example a fifty four hundred RPM, we can recognize that it takes eleven milliseconds to do one full rotation of the disc. So if we move the head to the appropriate location and happen to notice that the sector that we want to read, the location where we wanted to read just went past, it's going to take eleven milliseconds for that to come around again. So that could take some significant time, also. The benefit to having a hard disk drive is that the size can be enormous, sizes of four terabytes six terabytes are not usual today. So these sizes are really significant amount of storage space.

The unfortunate downside is that accessing that storage does take a little bit of extra time with a solid state disk drive, which is where a lot of the industry is moving to for most end user machines. This solid state destroy don't have anything that moves, hence the term solid state. They contain a number of chips which look just like your USB flash drives, if you ever happen to accidentally crack one of those open. All the data is stored electrically in the chips and it looks very much like RAM chips, but the chips are in a very different design, the chips are designed very differently.

These are store data permanently, so that even if the power is removed from the chip the data is still stored which is a great thing because this is secondary storage and we don't want to lose the data if the power is removed. Thankfully with solid state disk drives, all the data can be accessed in random time and they can all be accessed in the same amount of time. So accessing the first byte on the drive and then accessing the last byte on the drive doesn't take any additional amount of time other than the first or the second bite.

Unfortunately due to the cost and design of these drives they're much smaller, so solid state drive are significantly smaller, they're a little bit more expensive but hard drives are larger, and they're cheaper but they perform slower. So again we have the memory hierarchy working here just inside of the term of secondary storage. So even inside secondary storage we still have this concept of a memory hierarchy working.

HDD vs. SDD 1.18

As you can see from the images, what we have here on the top is a spinning hard drive and you can see that magnetic disk that I was talking about there. The head is the component that looks like a straight line and that one is in what's known as a parking location right now once the disk spins up.

It would come off that parking location and move to different radii along the disk and that this would just continue spinning along until the read operation was complete or until the drive was shut down completely. So one of the issues that come up with a hard disk drive is that it does waste a lot of energy to move that disk physically, so there is quite a bit of energy savings in in going with the solid state disk drive, which is along the bottom, and there's a performance benefit also for

solid state disks. Unfortunately the downside is with a solid state disk you get significantly less storage than with a hard disk drive.

Networking 1.19

Since we now live in a globally connected world. We now understand the data can come from a variety of sources literally around the world or even part of it, right? The ISS has connections to the Internet, so we might be talking with somebody on the Internet and they might not even be on Earth. Networks are connected via the Internet and we can access data anywhere on our network, so now we can really access data from anywhere.
And it's important to understand how that data comes in and how that data moves around the Internet and around networks. So that's where we're going to talk about.

Physical Connections 1.20

So there's a lot of different ways that computers can be connected physically. And we want to take a look at a few of those in the most popular ones that have come up with are copper, fiber, and wireless. The copper is your standard Ethernet cable or any sort of physical cable that involves a metallic connection, and that's usually called copper because it's usually made up of copper. Very commonly inside of a copper cable there might be eight wires, four pairs of cables and they're twisted together in a form of an unshielded twisted pair or you might have heard UTP cable so that's the idea behind a copper physical connection. Fiber is an Advent that's come out in the past thirty years and it's actually where we transmit information via light on a piece of glass.
That's a cable but it can transmit with a lot less attenuation, so we lose a lot less information in that data transfer via light and it can go a lot farther distances. So fiber cables go faster and they travel longer distances, unfortunately they cost a lot more too because it's glass. And then the latest one is wireless, we've had wireless only for about the past twenty years and it's certainly taken off.
We see wireless virtually everywhere usually in the form of a wifi connection which we'll talk about. But wifi is only one type of wireless and we have to understand that there's quite a few types of wireless including microwave and a lot of others. Protocols are in use for a lot of situations, for example we want to know when we need to start sending data, we want to know who it's coming from, we want to know who it's going to. In terms of physical connections, the idea of a protocol is very important because it's a language, it's a language that two computers are going to be talking and each of them have to understand the complete language. So we need to know, we need to have a very well defined protocol and thankfully we do. As far as physical connections are concerned we can talk about ethernet and ethernet physical connection would define both the protocol and the type of connection so we might have for example one hundred base T, which is a fast ether net connection better known as Fast ethernet, that's one hundred megabit connection of ethernet over copper and it defines how we would communicate; it defines when we start sending data, it defines when we stop sending data, and it defines the format for

information that's being passed over that network. So we have a lot of these different protocols including wifi which you've heard of as the 802.11 group, the 802.11 group is defined by the IEEE and it has a lot of different sub working groups for example 802.11B was the original one that produced eleven megabit wifi and we've 802.11N, and AC, and there's a lot of different letters that go associated with that and then we have ATM which is asynchronous transfer mode which is used in a lot of backbone networks for transmission of high capacity links.
So all of these are our physical types of connections that we would have to physically connect to multiple computers to a network.

Packets 1.21

When we used to make a call on the telephone, we'd have a physical connection. A physical connection between your telephone and the destination telephone and you can go back to the very, very old days, your parents and grandparents days, of making telephone calls. They used to call the local operator and ask for a connection to the national operator and the national operator would connect to another local operator another local operator would connect to the destination. Along this line they would create a physical channel so one wire that went from the source phone, your phone, to the destination phone.
That's absolutely not what happens at all today. Now we're sending packets. So today we send small amounts of information, very, very small amounts of information usually of around a thousand bytes or fifteen hundred bytes and the data gets sent from one program running on a machine to another program but unfortunately it can't be sent directly, it has to be sent via protocols which each one will add a small amount of information to this packet.
So the way that we send it is buying capsulated the packet and it's given to the lower level protocols, so we take the data and we give it to the next lower level, and then the next lower level gives it to another lower level, and the next lower level gives it's another lower level, so we have a higher key that's created whereby the data stands at the top. And then we have a small amount of information for application protocols added to the front of that data, and then a network, for example, layer will add a small amount of data to the front of that. Now the purpose of all of this is so that when it's received on the destination the destination can decide which for example application is supposed to receive this information so we might have for just to use a good example we might have two tabs of chrome open, or even we might have chrome and Internet Explorer open and if we have chrome an Internet Explorer open and both are browsing the exact same website at the exact same time when we receive data back, it's really important for the computer to be able to route appropriately the response from the website either to chrome or to Internet Explorer and we do that via quite a few protocols which will talk about in just a minute, but it's important to know that path back. When the packet moves to the logical protocol we add a header and so on and so forth until the data is actually sent on the physical network.

Layers We Commonly Use 1.22

So we have a lot of layers that we often use, and I've chosen some ones that are probably familiar with here. Right now you're watching this video via a website and the way that you've done that is by connecting via they hypertext transport protocol or HTTP. And that's an application level protocol which is used to send data to and from websites.

We also have the Simple Mail Transport Protocol (SMTP) which is used to pass mail between mail servers or the Internet Message Access Protocol, the Internet Mail Access Protocol (IMAP) which is used to obtain our mail when it finally is on our mail server we use IMAP to get that down into our machines.

So those are the application level protocols and then if we move down the scope we can look at logical level protocols which is actually more formally called network level protocols and that's usually broken down into two layers and it ties back to the idea that we have to simulate the concept of a connection like we used to have on the telephone networks.

So if we're talking about simulating a connection. There's going to be a little bit of overhead involved, so there's going to be some extra data and there's going to be some extra processing time involved in creating a connection, oriented connection if we don't need that, if we don't need to have guaranteed delivery of their packets or if we don't need care that they get out of order, for example, we can use what's called a connection less connection and the most common ones that are in use today are UDP and TCP. UDP being the connection less and TCP being the connection oriented.

So right now you're watching this video via HTTP over TCP. We also need to know how to route the packets appropriately from a global perspective. So now each machine in the world is required to have what's known as an Internet Protocol address, an IP address, and that IP address has to be unique for every computer in the world. And I know I'm being overly simplistic, and those people that know will know that there's things called network address translation and we deal with this because of the exhaust of the IPV4 protocol addressing scheme, but let's be simple and say that every computer in the world has to have a globally unique IP address. What that means is that your IP address is unique to your computer and if I need to send information directly to your computer I can do so by sending it directly to your IP address. And IP addresses are grouped together so that they form networks and it creates a larger whole which is great, but the IP protocol will add an additional header so that the information can be routed to your machine and you know where it's coming from. And then we have, lastly, the physical layer which often adds both the header end of footer to the end which is mostly concerned with local addressing and how to deal with whatever eccentricities there are in the physical medium.

So we can talk about ethernet which is very common protocol for use on a physical layer it's in fact probably what you're using and then 802.11 has some physical concerns as well as we move do HTTP header we would add a TCP header we would add an IP header and then we would add some data link or physical header like ethernet So as you're watching this right now we've probably gone through a couple

of million packets of information that have been sent via HTTP, TCP, IP, and ethernet altogether.

Conclusion 1.23

So in this module we looked at the definition of computer. We looked at types of computers we saw the insides of the computers. We talked about what each component of the computer does and the commonalities between those components we looked at system bus and how all those components communicate with each other we talked about how the CPU work specifically with the fetch and execute cycle.
We talked about the memory higher key how things on the top are much faster but much more expensive and things on the bottom are much slower but much cheaper. We talked about hard disks and the difference between solid state this drives and hard disk drives and then we got into computer networking and how data travels around the world.
So I hope you followed and I hope you enjoyed that see you next week.