Week 1 Module 2 Positional Number System

Digital Data 1.2

Hi there, hoping you're having a great day. Today we're going to talk about position and number systems, and we're all used to working with the decimal number system. We're going to see some other number systems today and see how we can generalize the way we work with a decimal number system to work with some other kinds of number systems. But before we do that, let's talk, let's say a few words about a memory of a presentation.

So we know that our memory basically is made out of bits that are grouped into a bytes, each one of these bits physically is made with electricity flow or a magnetic field. But we like to think of it abstractly as zero and one because it only has two states. We would need to figure out how to represent a lot of data using only this kind of a primitive unit. So we can think that for example numbers can be represented using the binary number system, we'll talk about it in much more detail soon. But the idea is to represent not only numbers, but other kinds of data is as well.

For example text, if we write a document or an email. How is that represented using zeros and ones and not symbols as the alphabet symbols? Images; how do were present images with all the colors and details with the zeros and ones? How do we represent videos, or how do we represent audio? All of that should be represented using only zeros and ones. So the key idea here is that everything would be represented as a number, and in a few minutes we see how we represent numbers, but first let's see that all kinds of data can be transformed into numbers, for example text. We have a mapping for each letter, for each symbol in our alphabet into a number. So if we want to take the text, for example “Hello world!” we can use this mapping. For example, the upper case “H” would be 72. So we start with 72 and then the lowercase, “e” is 101, so we have 101, and then lowercase “l” is 108. So we have two 108’s and so on. Every character has its own mapped number, and we have a sequence of numbers to represent basically a text or a sentence. Not only letters, but also a symbol, such as space or a question mark. Every symbol has its own a number that its mapped to, so text also is mapped into numbers.

Images are a bit more tricky. Let's say we have a beautiful image that we want to represent with numbers. So you know that images is a very dense matrix of pixels: each small pixel has its own unique color and then a color can also be represented using numbers. To represent a color, a common way to do that is using an R.G.B., a representation basically the amount of red, green, and blue. Each one ranges 0 to 255. So for example, this purple here is 120 red, 100 green, and 200 blue. The Pink has other values of R G. and B. and so on. So basically, each pixel has three numbers that represent its color and then the entire picture would be a very, very long sequence of numbers that represent the level of each color. Videos are basically a sequence of images, so if we know how to represent images with numbers, obviously we can represent videos with numbers. Audio also can be represented with numbers. If we sample the audio wave at a very high rate of times, we can see the voltage level in each point of time that is also a number.

So basically, all our data is first transformed into numbers. Numbers can be represented with the zeros and ones, we see that in a few minutes. And therefore, all our data in our memory is then represented with the zeros and ones. Let's go ahead and see how we can represent numbers with zeros and ones.

Counting in Base 10 2.1

In order to get used to working in other number systems, let's start with the most trivial thing we can do. Let's start counting, and in order to be able to count in different number systems, let's take a deeper look at the decimal number system in base ten. How we and what we're doing in that number system and then we'll try to generalize it for other number systems as well.

So let's start counting in the decimal number system in base ten kind of going back to second grade, but let's do that. So, zero after that would come one, after one would come two, three, four, five, six, seven, eight, nine. Then, when we get to the tenth object, instead of having a single symbol to represent the amount (the quantity of ten), we grouped ten ones into a single ten. We have this ten in the second position in our number and then we have zero additional ones. So one zero basically means one group of ten and zero additional ones. Then we have eleven, which is one group of ten and a single addition of one, twelve, thirteen, fourteen, up to nineteen. After that, when comes the next object having a group of ten and ten ones, these ten ones would be grouped into another group of ten, so together we would have two groups of ten and zero ones. And so it continues twenty one, twenty two, and so on, up to, for example ninety nine. And after ninety nine, the next object would have ten ones that would be grouped into another ten, therefore, would have ten tens and these ten tens would be grouped into a single one hundred that is represented in the next position. So we have one of one hundred group, zero of the tens groups, zero of the one. And so on, it continues.

So this is what we do when we are counting in base ten; how we look at the positions of the digits in a decimal representation. The digits we use in order to represent the numbers are zero to nine, to represent all the different values we can have in each position. After nine, we're just not using this position anymore we're grouping it and moving to the next position.

Counting in Base 5 2.2

Let's try to do the same. Now let's try counting in base five.

So it would start zero, one, two, three, four, and now in base five, we are grouping five objects together. In base ten, ten objects (ten ones) were grouped into a ten, ten tens were grouped into a hundred, and so on in base five were grouping more often. We're grouping five objects together so five ones would be grouped into a base five-ten, and five tens would be grouped to one hundred and so on.

So now that we are counting the fifth object, instead of having the single symbol a single digit to represent five objects, we are using one group of five objects and base five-ten and zero additional ones. After that, we have eleven, twelve, thirteen, and fourteen, and the next object would give us, again, five ones, that would group to another group of ten. So, we would have two groups of base five-ten and zero additional ones. And so on: twenty one, twenty two, up to one hundred. Can you think what comes before hundred? I hope you said forty four, because after forty four the next object would be five ones that would be grouped to another ten, and then we would have five tens that would be Group to one base five-hundred. No tens, no ones, and hundred and one, hundred two, and so on. In this case, the digits we needed to use are only zero to four, because we don't need any other symbols to represent numbers in this kind of representation. We are grouping the five objects together so we only need to represent quantities of zero, one, two, three, and four, as a single object.

Counting in Base 8 2.3

The same thing would work when we are counting in other number systems as well. I'll talk about some of the more important ones. One of them is the octal number system, short for base eight. The digits here would be zero to seven; we're going to group eight objects together, so we need symbols zero up to seven.

It would start zero, one, two, three, four, five, six, seven. After this seven, we would have one group of octal ten and zero ones, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen. After seventeen, the next one would be grouped into another ten. So we would have two tens, zero ones and so on. Again, try to think what comes before a hundred. Hope you got it right, it's 77 right? After 77, the next object would be grouped into another ten, the next eight tens would be grouped into one single object of 100 or 100 things.

Counting in Base 2 2.4

You can see that since our computer uses zeros and ones, different number systems use different symbols for the digits. If we want to use only zeros and ones base, two would do exactly that. For base two, the only digits we would need to use is two and one, because two objects are grouped together. Let's start counting in base two, in the binary base system, and so it's zero, one after that would come ten. We're grouping the two ones into a single binary ten and no additional ones, after that we would have eleven. What comes after eleven? 100 right, because we're grouping two ones into a ten then we would have two ten's that would be grouped into one hundred. After that we would have 101, 110, 111 and then 1,000. See how fast we got to 1,000. Actually it makes sense because we group much more often; every two elements are grouped together. Not like in the decimal number system that ten objects would be needed to grouped together to a ten and then only 10 tens would be grouped to a hundred in only 10 hundreds would be grouped to a thousand. Here, much more often we are groupings, our numbers grow along much faster. So, we can keep on counting in binary.

Counting in Base 16 2.5

We'll talk about the binary number system in much more detail in a few minutes. But let's take a look at another very important number system: the hexadecimal number system. which is also called the base sixteen. Let's start counting in base sixteen this time. So zero, one, two, three, four, five, six, seven, eight, nine, after nine. So now we have a problem, because we are still not supposed to group the ones to a ten at this stage, right we are grouping only sixteen ones to hexadecimal ten but we don't have any more digits anymore symbols to use for a single digit number. So, when we worked with the number systems that are less than ten, I don't know, the octave, the binary, we just used a subset of the digits zero to nine. For the octal, we use zero to seven. For binary, we use zero to nine. But, now that we have a bigger number system, in this case base sixteen, we need additional symbols.

So, the common symbols for basics in common digits for basic things is zero to nine, which are the first ten digits we use anyway. And then we need five more, that would be the first five letters in the English alphabet A.B.C. D. E. and F.. So after nine, instead of grouping together and moving to the next position, we still have a single digit number represents the quantity of ten. That is A and then we have B. C. D. E. and F. which represents the amount of fifteen object, see still a single digit number, and then the sixteenth object would be grouped into a single ten and the next position , a decimal ten so we have one and zero additional ones. And then we have eleven, twelve, thirteen, nineteen, after nineteen comes. I hope you didn't say twenty, it's 1a right? Because after one hexadecimal ten and nine ones we have one hexadecimal tens and ten ones that are still not grouped, that is written 1a, after that we have 1b, 1c, up to 1f. After 1f, now comes the twenty, because we are grouping sixteen ones into a single hexadecimal ten. In this case, we would have two tens and no additional ones. Twenty one, twenty two, twenty three, up to 2f, then would come for thirty and so on. Can you think what comes before one hundred? Hope you said f.f. because it represents fifteen tens and fifteen ones. The next one would be grouped into another ten then we would have sixteen ten, grouped into one hundred and so on.

Equivalent Representation 2.6

Okay, now that we know how to count in different number systems, we can see that we can represent the same value in different ways, depending on the number system we're working. Let's take a look at the number 13. So, when I think 13, I have to be more specific because 13 can be a decimal number, can be an octal number, can be a base five number. They all use the digits one and three. So, I have to say what kind of a 13 I'm referring to. Now that we have a lot of number systems, the way to do that is to subscript the number system we're referring this number with the number thirteen. We enclose it in parenthesis and say subscript ten, basically meaning the decimal number system.

Let's try to figure out what's the equivalent value for the decimal thirteen in the octal number system. So, let's try counting up to thirteen. So one, two, three, four, five, six, seven, octal would after seven would group the ones into a ten, so we would have ten, eleven, twelve, thirteen, fourteen, fifteen. So a decimal thirteen is equivalent to a one five, fifteen, octal. Let's try to figure out what's the equivalent value in the base five number system. One, two, three, four, after four would come ten, eleven, twelve, thirteen, fourteen, after fourteen, we have another group so it would be twenty, twenty one, twenty two, twenty three. So thirteen decimal is equivalent to fifteen octal, and to twenty three base five. Let's try to see what this number is equivalent in the binary number system. Again, let's count: one, ten, eleven, one, hundred, one, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101. So the decimal thirteen is equivalent to 1101 binary. Let's do the same for the hexadecimal: one, two, three, four, five, six, seven, eight, nine, then would come A. B. C. D., so it would be a D. hexadecimal.

So, a 13 decimal is equivalent to 15 octal to 23 base five to 1101 binary and D. hexadecimal. Once again, you see that a lower number system, a binary number system, is a longer number and groups more often. So, we get a longer number. And a bigger number system, sixteen for example, is a shorter number, it's one digit in this case. It would be very interesting to have a method to be able to convert a number of presentations in one number system to its equivalent in a number of another number system, to translate N. in base b1 to N., the same amount representing the same value, but in base b2 and that's what we're going to do next.

Base Conversions 3.1

Okay, let's see how we can convert a number in one number system representation into the equivalent number but in a different number systems representation. Actually, I'm not going to show you the general form of taking a number in base b1 and converting it into an equivalent N in b2, I'm going to show you two techniques to do some other kind of a translation.

So, first one would be taking a number "N" and in an arbitrary base B. and converting it into decimal that would be a very important thing to be able to do. Cause a lot of times you'll get numbers in binary hexadecimal and you want to understand what they mean, basically, so you convert it into decimal.

And the other way around is also something important to be able to do. Taking a number in decimal, your natural number system, and converting it into, I don't know, the computer's number system, the binary, or the hexadecimal, or whatever. If you think of it a bit deeper, you see that these two translations actually allow you to convert from any number system to any other number system. Just compose them pmrafter one of the others: start with N and b1, convert it into a decimal, and then take the decimal and convert it into b2. So, basically going from b1 to b2, stepping in the decimal somewhere in the two-step process. But I'll describe each of them separately. Let's start with how to take a number in an arbitrary number system and convert it into decimal.

Base B to Decimal 3.2

So, for that let's start with the decimal number system itself; understand exactly what the positions of each digit mean, and then try to generalize it to other number systems as well. So, for example, number 375, decimal 375. So, we have three digits here: the ones, tens, and hundreds. So, the one basically stands for the five, that is in the one's position. Each value here, represents a single object, right? Each value in the tens position represents a group of ten objects, and each value in the hundreds position represents a group of one hundred objects. We like to write it as powers of ten, so one hundred is ten to the power of two, ten is ten to the power of one, and one is ten to the power of zero. So, now we can look at 375 as a weighted sum of these weights of the digits in their position. So, we have five times one object: ten to the power of zero. Plus seven tens: seven times ten to the power of one. Plus three hundred, which is three times ten to the power of two. If you add that, you get to 375, not surprising at all, but that what we mean when we say 375 decimal.

Now, let's look at 125 octal, for example. Once again, each digit here has its own weight, its own amount of objects that it represents. So, the ones digits represent a single object here as well. The ten digits represent a group of eight, right? And the hundredth digits represent eight eighths or eight octal tens, which means 64 objects. Once again, let's write it as powers of eight. So, sixty four is eight to the power of two, eight is eight to the power of one, and one is eight to the power of zero. So, 125 octal, we can look at it as the sum of: 5 times 1 times 8 to the power of 0, plus 2 times 8 to the power of 1, plus 1 times 8 to the power of 2. If you add them all up, you get 85. Basically, 125 octal represents an amount of 85 objects, decimal eighty five by the way. Same thing would work with the binary number system.

So, again each digit has its own weight, right? The ones digit is 1, the tens digit is 2, the hundreds digit is 4 and the fourth digit, the thousandth digit, represents an amount of eight. Again, as powers of 2 to the 0, 2 to the 1, 2 to the 2, and 2 to the 3. And the sum would then be 1 times 2 to the zero, plus 1 times 2 to the 1, plus 0 times 2 squared, and 1 times 2 to the three. You add them, you'll get eleven. So, 1011 binary is equivalent, represents, 11 objects, 11 decimal, right?

Same thing for hexadecimal 3b2. Each digit has its own weight: 16 to the 0, 16 to the 1, 16 to the 2, and then when you add it up, you get 2 times 16 to the 0, 11 times 16 to the 1, right? B. is one after the A: A is ten, B. is eleven. So, 11 times 6 into the 1 to 3 times 16 squared. All together it adds up to 946. If you look in a general form, you look at the number An to a 0, like you have n. plus 1 digits here. In the number system, base B., each digit would have its own weight: b. to the 0, b. to the power of 1, b. to the power of 2, up to b. to the power of N. And this number, if you add the digits with their corresponding weight, you'll have the formula: A0 times B. to the power of 0, plus A1 times b to the power of 1, and so on up to An times B. to the power of N. If you use this technique, you'll be able to figure out what's the decimal value that is represented, in this case a base B. number.

Decimal to Base B 3.3

Okay, so we know how to take a number in an arbitrary number system in base. B. and figure out it's decimal value. Now, let's try to do the other way around: take a decimal number and try to find its representation in base B, an arbitrary number system.
Actually, I want to demonstrate it only on base 2, on the binary number system, basically to convert the decimal number into a binary number. We can then generalize this method, translating decimal into other number systems as well.

Let's take, for example 75, the decimal 75, and try to find its representation, its binary equivalent representation. So let's think how we can get started here, so obviously the binary number is going to be a sequence of digits, each one is going to be a zero or one. We just have to figure out which digits is a zero

and which one is the one, since it's going to be a binary number, each digit has its own weight. So, we have the 1s, 2 to the zero, 2 to the 1s (the 10s), 2 to the 2s (the hundreds), 2 to the 3, to the 4, and so on.

Just to be more explicit, let's write these powers of two with their decimal value: so, 2 to the 0 is 1, 2 to the 1 is 2, 2 to the 2 is 4, and so on, 8, 16,32, 64, 128, and so on. So, each digit here has this weight and we need to figure out for each position, whether it's going to be a zero or a one.

Let's start with this position here, with the digit in with the weight 256. Should that be a zero or a one? I believe most of you said that it must be a 0. Let's try to argue why this is indeed right, or the only possible value for this digit. Actually, it makes a lot of sense because if we would put a 1 here, only this digit would give us the weight of 256 which is greater than the total weight we're trying to represent, the total value of 75. So, there can't be a one in this position. This position, therefore, must be a 0. Actually, not only this position of 256, but all digits that their weight is greater than 75: the 128, 256, 512, and so on. All of these digits must be a zero, right? The real question is what would be the value with the rest of the digits. So, all of them together should represent or should add up to the value of seventy five. And, let's try to go over them one by one and figure out whether they should be a 1 or a 0. We'll go over them from left to right, starting with this digit here, with a 64.

What do you think, should this digit here be a 0 or a 1? I believe most of you said that it should be a 1, and by the way I agree it would be a 1, but I want to argue why it must be a 1, why it can't be 0? Because my concern is that maybe this digit would be a 0 and the rest of the digits, some way, would add up to 75, even though this digit here is a 0. So, this is not possible but maybe it is. How can we argue that this scenario here is not valid? So, a good argument would say that the rest of the digits, or using the rest of the digits, the largest value that they can represent all together is having them all be ones, right? Taking all of their weights, that would be the greatest value we can have using only these digits. And then, when we add 1 plus 2 plus 4 plus 8 to 16 plus 32, it add up to 63, which is by the way even less than 64. It's obviously less than 75. So, having the 0 at 64 won't allow us to add up to 75 as total, which means that this digit would have to be a 1.

But, before we say that, I just want to you to note a very important thing regarding the sum of geometrical progression. So, if you add up the geometric progression of 2, for example 1 plus 2 plus 4 plus 8, up to for example, 2 to the power of K., you know what this sum adds up to? So, this sum adds up to 2 to the power of K. plus 1 minus 1. Basically, 1 less than the next power of 2 in that sum. So, in this case, 1 plus 2 plus 4 plus 8 up to 32, is one less than the next power of 2, one less than 64, which is exactly 63. So, there is a very important property, we're going to use it a lot in computer science, so just make sure you know the sum and memorize it.

So, in this case we agree that this digit here with the weight of 64, must be a 1. After having a 1 here, out of this 75, we already used 64, we already represented 64, which leaves us with 11. So, the rest of the digits must add up to 11. If we need to add up to 11, 32 must obviously be a 0 and so is 16. The weight of 16 is greater than 11 itself, right? The question again comes when we are in a weight that is less than 11. 8, in this case, should this be a 0 or 1? I believe most of you said 1, but let's give this argument again. If for some odd reason, we have a 0 here, the greatest value we can have with the rest of the digits would be 7, right? 1 less than this 8, which is obviously less than 11. So this bit here, must be a 1. Having the 1 in this position out of the 11, we already represented 8, which leaves us with 3 for the rest of the bits. So, the four must be a 0, and the 2 must be a 1, which leaves us 3 minus 2 with the 1 for the rest of the bits. And that is exactly the value that we're going to put for the 1 digit.

So, you see that this method here figures out the 75. Now that we can copy these bits up there and see that it is 1001011, whatever, and more importantly, this method exposed the bits from left to right. There is an

alternative method that would expose the bits from right to left, and won't talk about it now but you can look it up on Wikipedia.

Binary & Hexadecimal Base Conversations 3.4

Okay, so now we know how to translate a number in an arbitrary number system to decimal, enough to take a decimal number and turn it into a binary. You could generalize it and turn it into any other number system as well. I want to show you, and basically this is enough to convert from one number system to any other number system. If you do these two things as a two step process, but I want to show you another kind of translation and that is to convert a binary number to its equivalent hexadecimal number. It's going to be a direct way, where a direct process that does this translation is going to be a very fast and simple one, I hope. And the question that we ask ourselves is why do we care about the hexadecimal number system? I think I have a better, or I have a good answer for that in one of our future models but for now, let's just see how we can convert a binary to hexadecimal number back and forth.
So, for example, let's take a 3b9, a hexadecimal number, and try to figure out its binary representation. So, first I'll show you the technique, how technically you take a hexadecimal number and turn to binary and binary to hexadecimal. I'll just show you, without explaining why it works just how to do that, and after that, I'll try to justify it in some sense.

So, for example, when you get this number 3b9 hexadecimal and you want to convert it into binary. What do you do is actually, with time you memorize it, but for now let's have a table. That one side would be the hexadecimal digits and the other would be the 4 bit binary representation of these digits, so that the hexadecimal digits are 0, 1, 2, 3, up to F, right? And their 4 bit binary representation goes like that. So 0 would be four 0s, 0. 1 would be 0001, 2 would be 0010, 3 would be a 0011, and so on. If you keep on going, you'll see that basically this 4 bit binary representation gives us all the combinations, all the possible values or options of four bits, and it's not a coincidence. It's part of the reason why this method kind of works. So, assuming we have this table, on a piece of paper or we just memorize it, again with time you memorize it. The process of converting hexadecimal number to a binary one works like that, we work digit by digit individually, independently, and write its 4 bit extension instead of the digit.

So, for example 9 is 1001, and that what will have instead of the 9. b is 1011, that's what we'll have instead of the b. 3 is 0011, that's what would have instead of the 3. So, we have 3b9, each one extended to 4 bits. All of these together gives us the binary equivalent of that number. So, it's kind of simple if we memorize this table, and it's kind of surprising because we are actually translating the hexadecimal number digit by digit, individually and independently. We didn't do that when we wanted to convert 75 decimal to its binary presentation; we didn't convert 7 on its own and 5 on its own and together it was 75, we had to do something that is much more global in order to convert a decimal number into a binary one. But hexadecimal, maybe because it's 2 to the 4, it's a power of two, allows us to have a much easier process to make these conversions.

By the way, the other way around taking a binary number and convert it into hexadecimal, would be very similar. We'll have split the numbers into groups of four and each one to match to its hexadecimal digit. Just one thing to note when we group the numbers by fours: we need to start from right to left, because if one of the groups or if the number of digits doesn't divide evenly by four, it is good to add a 0 to its left without changing the value of the number. And then we would get like a whole number of groups of fours. In this case, the first group is a 3, the second group is a d, the third group is a 6, and all together we get 6d3.

Binary and Hexadecimal 3.5

Okay, so now that we know the technique of how to convert the hexadecimal to binary and backwards binary to hexadecimal. Let's try to say a few words justifying why this magic here works, that we can translate each digit individually, independently without caring with its surrounding. I'll try to give you the intuition of the argument, basically demonstrating it on a specific binary number and how to convert it into hexadecimal. Hopefully you would be able to take this calculation here and generalize it into formal proof, general proof for all cases. So, let's look at this binary number and I'll try to make some mathematical equations, or quantities, that eventually would end up with the hexadecimal equivalent. But, as we go, you'll see how we are relating back to the table and converting each 4 bits independently to its hexadecimal digit.

So, let's get started: so we have this number here, it's a binary number, therefore we can write it equivalently as a weighted sum of powers of 2. So, it is 1 times 2 to the 0, plus 1 times 2 to the 1, plus 0 times 2 to the second, and so on. We can go up to, I think it goes out to 0 times 0 to the power of, raised to the power of 11, and now let's look at this sum here, with 12 errands. And probably, not really surprisingly, we'll divide it into groups of 4 errands. So, let's start with the first 4. In this case, let's take the first 4 errands and just actually copy them. So, we have 1 times 2 to the 0, plus 1 times 2 to the 1, plus 0 times 2 to the square, plus 0 times 2 to the power of 3. So, obviously our first 4 errands in the upper side are equal to the first 4 in the lower one. It won't make any difference if I also multiplied by one, right? Or in other words, instead of one I can just write 2 to the power of 0, exactly the same, right? So that's what we do with the first 4 errands, we kept them equal just wrote it a bit differently.

Let's take the next 4 errands, in this case, instead of just copying them as is, we'll factor them by 2 to the 4. So, instead of 1 times 2 to the 4, we do 1 times 2 to the 0. And later we'll just multiply it by 2 to the 4 and 0 times 2 to the 5 would be 0 times 2 to the 1 and 1 times 2 to the second, and 1 times 2 to the third. So, taking these 4 errands in the parenthesis and multiplying them by 2 to the 4, would give you the second 4 errands up there.

For the last 4 errands, we'll do something very similar. We'll write them but will factor them in this case by 2 to the 8. Right. So, the first one would be 0 times 2 to the 0, plus 1 times 2 to the 1, and so on. You can see that each parenthesis here has some combination of the first 4 powers of 2.

Let's take a look at these ones for starters. Let's go back to the table and if we take a look here, at this line here, this number 0011 actually equals to 1 times 2 to the 0, plus 1 times 2 to the 1, plus 0 times 2 to the second, plus 0 times 2 to the 3, which is equivalent to 3. So, this thing here is equal to 3, same thing. So, we can write it 3 times, to do the same thing with these 4 errands in the second parenthesis. Again, it's a combination of the first 4 powers of 2. Going back to the table, it would match exactly this line there with 1101. Because 1101 basically stands for 1 times 2 to the 0, plus 0 times 2 to the 1, plus 1 times 2 to the second, plus 1 times 2 to 3, which is equivalent to 13, hexadecimal D. but it's equivalent to 13. So, this thing here is 13 times 2 to the power of 4. Same thing with these 4 errands, we'll look at the table. It matches this line and it adds up to 6. So, we can write 6 times 2 to the 8. So, again, as we said it's not a coincidence that our table there gave us all the combinations of 4 bits. Because when we do the math, we would get some combination of the first 4 powers of 2: 2 to the 0 through 2 to the 3.

Now, let's take a look at the powers of 2 that we have there. 2 to the 0, 2 to the 4, and 2 to the 8, so 2 to the 0 is basically 1, which is also 16 to the power of 0. You see like I'm getting us towards a hexadecimal representation, something times powers of 16, so this would be 3 times 6 into the power of 0. Fortunately for us, actually it's not very fortunate, it just works, 2 to the 4 is 16, which is 16 to the power of 1. So, this digit here is also 13 times a power of 16, and 2 to the 8 is 2 to the 4 squared, or in other words, 16 squared. So, we get 6 times 16 squared. So, we can view that as a hexadecimal representation because it's

3 for the ones digit, 13 or D for the tens digit, and 6 for the hexadecimal hundreds digit. So, that very long binary number is equivalent to 6d3 hexidecimal.

Addition 4.1

Okay, so now that we know how to work with other number systems, we know how to a convert from one system to another. Let's see how we can do some math; how we can do some arithmetic in different number systems. Let's start with addition: how we can add numbers to one another. So again, let's start with a decimal number system. And make sure we get exactly what we do there, and then, try to generalize it to other number systems as well.

So, for example, let's try to add 325 to 692, two decimal numbers. We start, I think third grade, our teacher said start with the ones digit. So, we add 5 and 2, we get 7. And then, for the tens digit we had 2 tens and 9 tens, that would give us 11 tens, which would, 11 tens would be grouped into 100 and an additional ten. That's why we write 1 in the tens, and carry over an additional hundred. Then we have 100 plus 300 plus 600, it adds up to 10 hundreds. And we have, therefore, 0 for the ten hundreds and 1 for the thousand because these 10 hundreds were grouped into one group of a thousand. So, we have 0 hundreds and 1 carried to the thousands, which would then be dropped down, and all together we have 1017.

Let's try to do the same when we add, or we try to do some additions, in the Octal number system. For example, a 365 plus 243 Octal. Again, we start with the ones digit, 5 plus 3. When we add 5 plus 3, we get 8. But we don't just write 8, because in the octal number system, 8 ones are grouped into a single ten. So, after grouping these eight ones to a ten, we have 0 additional ones and a carry over of a ten. Then, we add a 10, another 6 ten and another 4 ten, that would give us 11 tens, right? Maybe let's try to count it; count it up starting at 6. So, we kind of have to add: 6 plus 5. So, let's add 5 to 6. Let's count. After six comes seven, after seven we have ten, eleven, twelve, thirteen, so we have thirteen in the tens position. So, we group ten of them to hundreds, we are left with 3 tens and one hundred is carry over. Then, we have 3 plus 1 that's a 4, plus 2 that's a five, a six, that would give us 6. So all together, 365 octal plus 243 octal, gives us 630 octal, right? Same thing as we've done with the decimal, we are working in the octal. We just keep in mind to group items when we have eight individual ones.

Same thing would work when we try to add binary numbers: 0 plus 1 would be a 1, 0 plus 0 would give us 0, 1 plus 0 would be 1, 1 plus 1 that is 2 obviously. That would be 0 plus 1 of a carry over, 1 plus 1 plus 1, that's a 3, that would leave 1 plus 1 of a carry over. 1 plus 0 plus 0 would be a1, 0 plus 1 is 1, and 1 plus 1 is basically 2, or 10. Or a 0 plus a 1 carry over, which then drops down. And that is the result of adding these two binary numbers. Try a few on your own, and make sure you get how to add the numbers in terms of alternative number system.

Subtraction 4.2

Okay, now that we know how to add numbers in different number systems, let's see how we can subtract numbers in a different number system. Again, let's start with decimal for practice, and then try to generalize it to other number systems as well. So, let's try to subtract 427 minus 192 decimal. So, we start with the ones right. 7 minus 2 that would give us a 5, then we try to subtract 2 minus 9 and that's a problem because we can't subtract 9 from 2, so we would need to borrow an additional 10 tens, actually in this case. So, instead of having 2 tens, we'll have 12 tens, and that by splitting the hundred, the 4 hundreds into 3 hundred and 10 tens. That would leave us with 3 hundred and 10 tens additional 2 tens that would give a 312 ten. So, instead of saying 42 tens or 420, we say 312 tens. Then we have these 12 tens minus 9 that would give us a 3 and 300 minus 1, that would give us a 2. So 427 minus 192 is 235.

Let's try doing the same for the octal number system. Let's try to subtract 351 from 536. Same thing, we first subtract 1 from the 6 that would give us a 5, then when we try to subtract 5 from 3, that is not possible. So, we need to split one of the hundreds, one of the 5 hundred into tens. So, instead of having 530, we would have 4 hundred and 13 ten. So therefore, we have 13 tens that we need to take 5 out of them. So, let's count back out of thirteen, so it's twelve, eleven, ten, seven, six. So, we have 6 there, and 4 minus 3 is 1. So, 536 minus 351 is 165 octal.

Signed Numbers 5.1

Okay, so now we know that we can represent numbers using the binary number system, basically base two, so numbers can be represented using zeros and ones. But then, let's see how we can represent not only positive numbers but also negative numbers. So for example, 26. The decimal 26 is equivalent to 11010 binary, right? Because if we add up the weights, 16 plus 8 plus 2, it would add up to 26. Check it yourself.

How can we represent negative 26 decimal in binary? So obviously, we can write negative 11010 but then inside a computer, we don't have, we only have zeros and ones. We don't have the negative sign, we can't have that represented. So, we would need to figure out a different way to represent negative numbers, signed numbers, using only zeros and ones.

The first approach I want to show you is called "Sign and Magnitude." It's a very intuitive approach and it says it's a slight variation of a binary number system, the formal binary number system. It said that not all of the bits have their weight depending on their position; the left most bit is used only to say what the sign of the number is, the rest of the bits would have their weighted value depending on the position. So, for example, a positive number is 0, a negative number is 1 for the sign bit and twenty six would be whatever 26 looks like. 000 and then 11010 so negative 26 in the sign magnitude representation, would start with the 1 as the sign and the rest of the bits would just represent 26. This way using only zeros and ones, we can represent negative numbers as well. Just know that the one in the signs bit position doesn't have a power of two as its weight, it represents something else, its represents the sign of the number whether it's positive or negative. So, that's a very intuitive way to represent signed numbers. But then, that's not the way computers typically represent signed integers. The typical way that computers represent signed integers is called "Two's Complement" and I'm going to explain this representation method right away.

Two's Complement 5.2

Okay, let's see how the 'Two's Complement' representation method works to represent positive and negative numbers. So, when you want to represent the number, assigned number, in the Two's Complement representation method it kind of matters how long, how big you want, how many bits you want in this representation. So, let's let me give you a general definition here: so, for presenting a number in k-bits, it has two major properties that you should know about. The first is that if your number that you want to represent is positive, then the first k1, (k. minus 1) bits are represented as the unsigned binary representation, basically the regular base 2, and then the last bit would just be a 0. So, in order to represent a positive number just take its base 2 representation in (k. minus 1) bits, and the k-th bit would be a 0, on its left. And another important property for the Two's Complement representation method is that if you add a number and its additive inverse, its value with the sign flipped, then the sum would be 2 to the power of k. I suggest you write these two properties down when you look at the next example.

So for example, if we want to translate the decimal value 26 into Two's Complement using 8-bits, okay so our k now is 8. So it's 8-bit Two's Complement representation. Then, since it's a positive number we should look at 26 as in a binary representation, basically in seven bits so the first seven bits would be the

binary representation of 26, like that. And then the left-most bit would be 0. Very, actually, very straight forward.
Let's try to figure out how negative 26 would look like, in once again, an 8-bit Two's Complement representation. So, we said that adding a number and its additive inverse should give us 2 to the k. In this case, adding 26 to negative 26 should give us to 2 in this case, k is 8, 2 to the 8. So 26, we know how it looks like, right? 2 to the 8, if you think about it, it is 1 and eight 0s; just as 10 to the 8 decimal is one and eight zeros, in binary 2 to the 8 would be one and eight zeros. You can figure out the weight of each digit and the ninth digit, basically, is 2 to the 8. But now let's try to figure out what negative 26, would, should look like.
So we know that if we add 26 to negative 26, they should add up to this 1 and eight 0s. So let's try to figure out negative 26, digit by digit. So zero plus what, would give us the zero? Zero, obviously. One plus what, would give us a zero? It could either be zero or one, obviously, it's not a zero. So it must be a one, but not only that it gives us a 0, it also carries another 2 to the next position. So 1 plus 0 plus what, would give us a zero? That's 1 plus 1 would give us a 0 and another carry over of 1. 1 plus 1 plus what, would give us a zero? 1 plus 1 plus 0 would give us a 0, with the carry over; 1 plus 1 plus 0 with the carry over. 1 plus 0 plus what, would give us a zero? 1 plus 0 plus 1 would give us a 0, right? It's basically a ten so it's a 0 and the carryover of another 1; 1 plus 0 plus 1 and 1 plus 0 plus 1. So, taking this number here of 11100110, this thing here, that's the 8-bits Two's Complement representation of negative 26.

Two's Complement 5.3

Okay, let's do another example working with Two's Complement. So let's take this number here, 00101101, represented in a 2-bit Two's Complement, that's the representation method for this number, and try to figure out what's its decimal equivalent, what this number basically represents. So, since it a starts with a zero, we know that it is a positive number. Then, this representation method says that the first (k minus 1), seven bits, basically represent its value. So if we add 32 and 8 and 4 and 1, we get 45 and this number is basically the decimal 45.
Let's take a look at this number here, 11101010, again in an 8-bit Two's Complement representation and try to figure out what its decimal representation. In this case, since it starts with the one, it's not a positive number. But this a presentation method is not 'Sign and Magnitude' that we can just figure out the value of the lower seven bits and say that it is negative of this value. Since it's a Two's Complement representation method, we should figure out the absolute value of this number in a different way.

So let's use the second property that the sum of a number and its additive value, additive inverse, is 2 to the power of 8. In this case, we know that, we don't know what these values are but we know that this number here is some kind of a negative something. Let's say negative X. So 11101010 is some negative X. and we'll try to find what positive X. is, so we can know what the absolute value of this number. So once again, we're trying to figure out what number is missing there, to be the additive inverse of the number that we have no idea what it is. But since the additive inverse would be positive and positive numbers are much easier to figure out their value, we can then figure out what the value of the negative number. So once again, let's try to find what's the additive inverse of our input of this 11101010. So 0 plus what, would give us a zero? 0 plus 0. 1 plus what, would give us a zero? Plus 1 with a carry over. 1 plus 0 plus 1, and then another carry over. 1 plus 1 plus 0 with the carry over. 1 plus 0 plus 1, and 1 plus 1 plus 0. And 1 plus 0 plus 0, and 1 plus 1 plus 0.
So, basically we got that our number is 11101010 but its additive inverse, its absolute value, it's positive corresponding value is 00010110. And this number is very easy to figure out its value because we just have to add the weights of each digit. So we have a 2 plus a 4 plus 16, and that adds up to 22. So this number here is positive 22, which implies that the negative number we had before is negative 22. So this number 11101010 in an 8-bit Two's Complement representation, it represents the number negative 22 decimal.