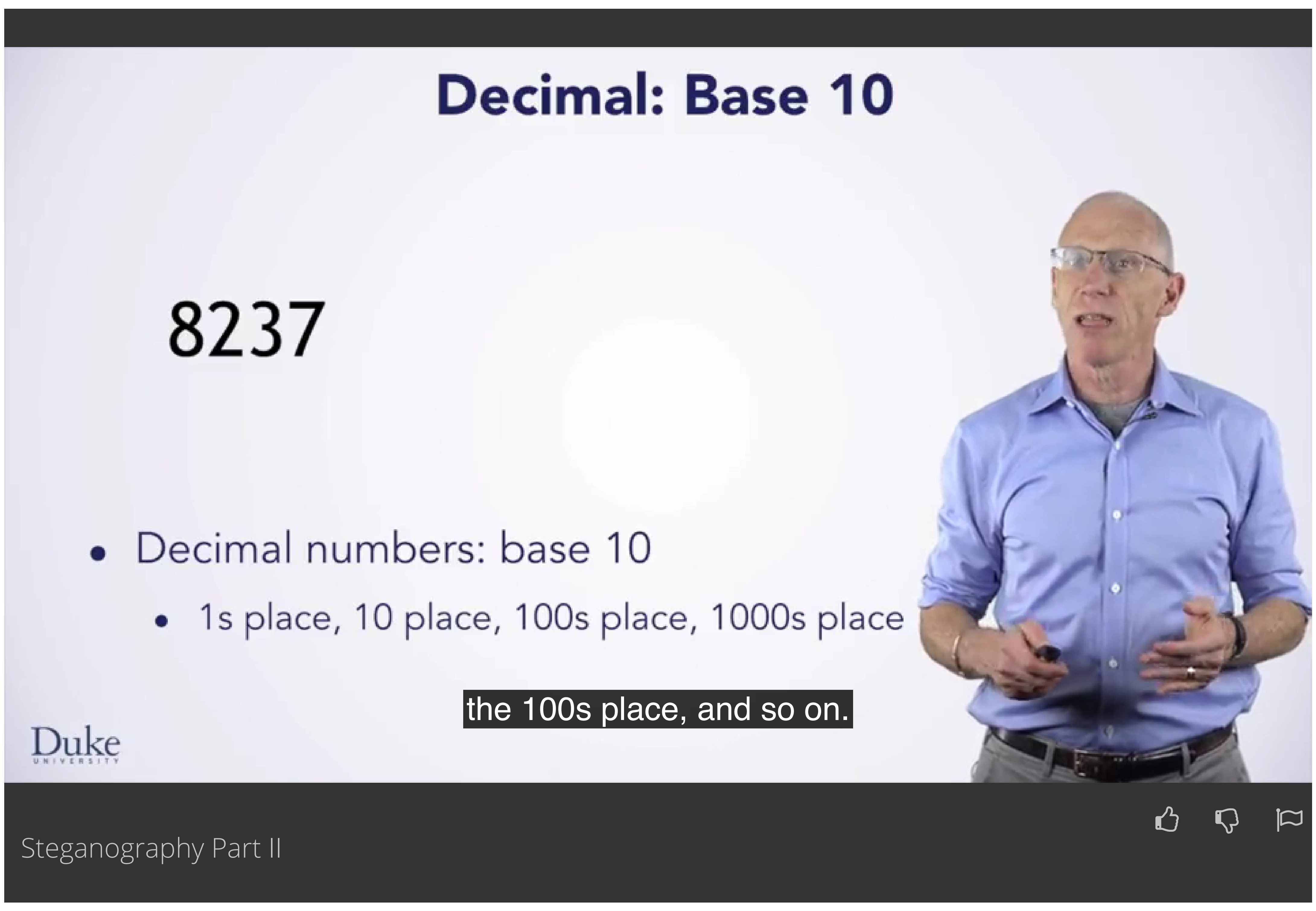

Decimal: Base 10

8237

- Decimal numbers: base 10
  - 1s place, 10 place, 100s place, 1000s place

the 100s place, and so on.

Duke UNIVERSITY

Steganography Part II

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript   English

0:03
Now that you have the basic idea of steganography, it's time to learn a bit about binary. So you can do the math on the numeric values of pixels. Let's start with a refresher on base ten numbers, the normal decimal system you use everyday. In a base ten number, the digits are in the 1s place, the 10s place, the 100s place, and so on. This means that a number like 8,237 is 7*1 plus 3*10 plus 2*100 plus 8*1000. The same principles apply in binary but the places are powers of two instead of powers of ten. Computers do this because they fundamentally work with electric circuits and transistors which are easiest to implement with two voltage levels. So, computers work naturally in a number system where you have only two values, ones and zeros. So the binary number 10111, one is 1 * 1 plus 1 * 2 plus 1 * 4 plus 0 * 8 plus 1 * 16, which, if you wrote in decimal, would be 23. The red, green, and blue components of pixels range from 0 to 255 because they are stored with eight binary digits called bits. Note that the b in bit comes from binary, and the it in bit comes from digit. With eight digits, or eight bits, a number can range from 0000 0000, which is the decimal number 0, to 1111 1111, which is decimal 255. These eight bits are in the 1's place, the 2's place, and so on up through the 128's place. If you were to add up all these place values, you would get 255. Now that you know that each color is an eightbit binary number, let's revisit this steganography problem. You want to hide the eight-bit number on the left, 10110010 in the number on the right, 01110101. As before, you'll hide the most significant digits. In this case, you use the four digits from each number, since that's half of the total digits. The resulting number would be this eight bit binary number, where we've colored the bits based on which of the two original numbers they came from. The most significant bits from the number you want to hide shown in blue are now hidden as the least significant bits of the color that will be displayed. That's how you do it conceptually, but how do you pull these numbers apart and then put them back together using mathematical operations that you can write down in code?

3:00
Let's again revisit the problem in the more familiar context of decimal numbers. Working similar problems in more familiar contexts can be a great strategy in general whenever you're approaching problems that have unfamiliar concepts. The first question we might ask is, how do you extract or get 82, the blue digits out of 8, 274? The number on the left, or the 82, is underlined in blue. You could take 8274 and divide by 100. That would give you 82.74. And then throw away the 0.74. The mathematical operation called taking the floor of a number, which means rounding it down to the whole number, less than or equal to it. In JavaScript, you would do this with Math.floor(8274 / 100), and that would give you 82. The same principle works to get the most significant two digits of 3,568. Math.floor(3568 / 100) is 35. In fact, you can get rid of the least significant digits of any base ten number by dividing by the appropriate power of ten and discarding the fractional part.

4:15
Now that you have 82 and 35, you want to put them together to get 3,582, which you can do simply by multiplying 35 by 100 and adding 82. In fact, this strategy works to combine any two digit numbers into a four digit number. If you wanted to combine numbers with more or fewer digits, you would multiply by a different power of 10. Now, let's go back to the numbers represented in binary. You can apply the same principle but now you'll need to use powers of two instead of powers of 10. And since we want to work with four digits at a time, we'll use 2 to the 4th, which is 16. So if you took Math.floor(10110010 / 16) you would get 1011. Note that there's nothing really special about binary math. Binary numbers are still just numbers, in fact, this really is 178 divided by 16 which is equal to 11. We just wrote the numbers down in binary rather than decimal. Similarly, to get the most significant four bits of the other number, you can divide by 16, and take Math.floor of the result. In this case, you're taking 117 dividing by 16 to get 7. Now, you can combine the two different four bit numbers into one eight bit number with the same principle we saw before with base 10. But this time, instead of multiplying by a 100, you multiply by 16 and then add.

5:51
That's the math required to hide one image in another.

5:55
But what about the math to extract the hidden image?. How do you get the least significant four digits out of the number on the right, shown in blue, to make them into the most significant four digits of the number on the left?

6:09
Let's again go back to the more familiar base 10 scenario. When you take 3582 and divide by 100, you get 35, with a remainder of 82. That 82, the remainder when you divide, is exactly what you want. So how to you ask JavaScript to give you the remainder that's left over when you use division? You use the percent sign operator, show here. 3582 % 100 is 82. The percent sign operator is called mod, short for modulus, the formal mathematical name for the remainder when you divide. So, we'd actually read this statement as 3582 mod 100 is 82. Once you have these two digits, you can just multiply by 100 to get the number you want, making them the two most significant digits of the resulting four digit number.

7:05
The same principle will work for binary digits. But again, you use powers of two instead of powers of ten.

7:14
Since you want four digits, you would use 2 to the 4th, which is 16, instead of 100. Taking a number mod 16 gives you the four least significant or right most bits, the binary digits. In this case, 1011.

7:31
If you take that number and multiply by 16, you end up with these bits as the four most significant bits of an eight bit number. You see that the leftmost, or most significant bits, are now 1011. That's exactly the math you need to extract a hidden image. So now you've seen how numbers are represented in binary and how you can extract the digits you want from numbers in base 10 or in the base 2 using either division or mod. You've seen how you can combine this digits back together in either based using multiplication and addition. And you've seen that you can use Math.floor to discard the fractional parts after you've divided. We've summarized these ideas on this slide. Remember, think about numbers in base 10 but do the math in base 2 by using powers of 2 instead of powers of 10. Enjoy your programming.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

71.20_H.1.20_SteganographyPartII.pdf
pdf

Would you like to help us translate the transcript and subtitles into additional languages?