MATH 2420 Discrete Mathematics
Lecture notes
Numbering Systems

Objectives:

1. Represent a binary (hexadecimal, octal) number as a decimal number.
2. Represent a decimal (hexadecimal, octal) number in binary notation.
3. Represent a binary number in hexadecimal (octal) notation.
4. Add and subtract binary numbers.

Introduction:

We use numbers to represent how many objects we see in a basket. Or how many people are in a theatre. Or how much money the national debt is at the moment. Those numbers are represented by the symbols $0, 1, 2, 3, 4, 5, 6, 7, 8$, and 9. Using those ten symbols we represent any number we can think of. But what are we actually representing?

In actual fact, there is no number '10'. Ten is actually two digits, a one and a zero, which represent a total count. If we separate the digits we see that the one represents a group of items and the zero is a place holder. We say that the one is in the "ten's place" and the zero is in the "one's place." The term, "ten's place" means that the digit in that location represents whole groups of ten items (people, apples, books, dollars). So the number "10" is actually "one group of ten and zero ones."

Written out mathematically we see it as

$$(1 \times 10) + (0 \times 1) = 10$$

Moving one more space to the left we find the "hundred's place". Digits in this position represent whole groups of one-hundred items. If we look at the number 125 mathematically we see it as

$$(1 \times 100) + (2 \times 10) + (5 \times 1) = 125$$

The concept of place to represent whole groups of items (e.g. hundred's place, ten's place, one's place) makes the representation of numbers a sort of code. Note that each position to the left is a power of ten starting with the one's place as "zero" (i.e. $10^0 = 1$). Then the ten's place, the first position to the left, is the first power (i.e. $10^1 = 10$). The hundred's place is the second power (i.e. $10^2 = 100$) and so on. Each position further left represents a whole power of ten, which we call **the base**. That makes this **base ten**, or **decimal**. A move to the left indicates an increase in **magnitude**. It is important to remember that the counting beings at zero, not at one. Anything raised to the power zero is one. Since the first position is the "one's place" then that is represented by the power zero.

10^3	10^2	10^1	10^0
thousands	hundreds	tens	ones

We are most familiar with base ten counting since we have ten fingers and learn to count using our fingers. It's still quite useful at times. Using base ten we can count anything we see, and somethings we can, and there is no limit to our capabilities in that regard. True, the numbers get so large that we can't grasp them in our head or write them, but then we use

a shorthand method known as *scientific notation* and use the power of ten to make things simpler.

One major drawback - computers don't count in base ten. They don't have fingers and they only know about electrical current. On or off - that's all they know. This presents us with a challenge.

To use computers we must understand how they "think" and that means we have to adjust our way of thinking to theirs. They don't have ten fingers, they only have two (on/off), so we must come down to them. This isn't necessarily in our favor.

Using only two states, on and off, to describe something is termed **binary**. The prefix *bi-* means "two". In a binary numbering system there are only two digits - 0 (zero) and 1 (one). Remember I mentioned that there really isn't a number ten, just two digits put together to represent a bunch of items? Well, in binary, or **base two**, there isn't a number two. Already things are getting weird.

> In any number system based on a number n, that number will not appear in the list of available digits.

We've changed the base of the numbering system from ten to two. This means that a move to the left, instead of being a higher power of ten, is now a higher power of two.

2^3	2^2	2^1	2^0
eights	fours	twos	ones

Numbers look rather strange in base two because you only see one and zero. So to represent the number 12 (base 10) in binary (base 2) it looks like so

$$1100$$

which can be expressed mathematically as

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0).$$

To differentiate between base 10 and base 2 numbers, we write the base as a subscript on the number, like so

$$1100_2$$

Converting Between Bases:

Converting a base 2 number to a base 10 number is rather simple. Each position in the base 2 number represents a power of two. Multiplying the binary digit (1 or 0) by the power of two at that position then adding the values together gives you the base 10 number. For example, the binary number 110101_2 can be converted to base 10 by the following:

$$\begin{aligned} 110101_2 &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 32 + 16 + 4 + 1 \\ &= 53_{10} \end{aligned}$$

Converting from base 10 to base 2 is not quite as intuitive. There are two ways to approach this.

One way is to find the largest power of two that will "fit" inside the number (largest value not greater than). Subtract that from the number, then try again. Repeat this until the number is reduced to simply the value one. Any power of two not used in a subtraction is "missing" and represented by a zero in the resulting binary number. As you find powers of two in the number, a one is used to represent that power of two. Writing out the ones and zeros in the order used (largest to smallest) gives you the binary representation.

A second way uses a repetitive division process. Starting with the base 10 number, divide it by two. The remainder (0 or 1) is written to the side. Then divide again, again writing the remainder (0 or 1) to the side. Repeat this until the quotient is zero. Perform the division going down the page in a column. At the end, read the remainders backward (upward) and that is the binary number. For example, convert 209_{10} to binary by the division method:

$$
\begin{array}{lcrcl}
2 & \rightarrow & 209 & & \\
2 & \hookrightarrow & 104 & 1 & \uparrow \\
2 & \hookrightarrow & 52 & 0 & \uparrow \\
2 & \hookrightarrow & 26 & 0 & \uparrow \\
2 & \hookrightarrow & 13 & 0 & \uparrow \\
2 & \hookrightarrow & 6 & 1 & \uparrow \\
2 & \hookrightarrow & 3 & 0 & \uparrow \\
2 & \hookrightarrow & 1 & 1 & \uparrow \\
2 & \hookrightarrow & 0 & 1 & \uparrow
\end{array}
$$

$$209_{10} = 11010001_2$$

Higher Bases:

Notice the number of digits it took to represent the number 209 in base two? Eight digits instead of three. Seems like it takes more digits to represent the same value in base two than it does in base ten. You'd be right in thinking that.

Since there aren't as many digits to use in base two, it takes more positions to represent a larger number. Each position represents a power of the base, but if the base is smaller then each position represents a smaller bundle. Larger bases means we would need fewer digits.

Base ten is a larger base, and one we are familiar with (see fingers), but it isn't a good base for computers. We've already discussed that computers think in binary, and ten isn't a power of two. Why would that matter?

If we change to a base that is a power of two, eight, say, then the conversion from binary (the computer) to the higher base (us) is easy. Let's look at base eight, for example.

Base Eight, or Octal:

Base eight is also known as octal. It didn't take early computer scientists long to get tired of the binary. Too long and takes too much effort to type and think. Better to use a higher base. Eight happens to be a convenient base since it is a power of two, mainly, two to the third (2^3).

The same rules apply to base eight as to base two or base ten - the actual number eight does not appear. The digits available are $0, 1, 2, 3, 4, 5, 6$ and 7. More digits, easier to represent larger values. The base changed so the powers at each position changed.

8^3	8^2	8^1	8^0
five-hundred-twelves	sixty-fours	eights	ones

And that's just up to 8^3. Still not as big as 10^3, but better than 2^3 by a long shot.

We have to remind ourselves that though the number *looks* like a normal base ten number, it isn't. It's base eight, or octal. Each digit past the first position (8^0) represents a power of eight and a bundle of eight.

Converting from base eight to base ten is the same as for base two. Each position is a power of eight, so multiply the octal digit by that power of eight, then add up all the resulting products.

$$\begin{aligned} 1567_8 &= 1 \times 8^3 + 5 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 \\ &= 512 + 320 + 48 + 7 \\ &= 887_{10} \end{aligned}$$

Converting from base ten to base eight can be done using either of the two methods proposed for decimal-to-binary conversion. The division method works the same with octal as with binary, only the remainders are now anything from 0 to 7 instead of just 0 or 1.

$$\begin{array}{lrll} 8 & \rightarrow & 209 & \\ 8 & \hookrightarrow & 26 \quad 1 & \uparrow \\ 8 & \hookrightarrow & 3 \quad 2 & \uparrow \\ 8 & \hookrightarrow & 0 \quad 3 & \uparrow \end{array}$$

$$209_{10} = 321_8$$

Conversion from base eight to base two, or vice versa, is even easier because no division is needed. Eight is a power of two, three to be exact, so we can use that to make the conversion easier. Given a binary (base two) number we can convert it to octal by grouping binary digits together into groups of three, starting at the far right (one's place). If we run out of digits at the far left we simply add zeros in front as place holders. Let's look at our example number, 209_{10}. In binary it is

$$11010001$$

which can be grouped into threes as such

$$\underbrace{11}\,\underbrace{010}\,\underbrace{001}.$$

Now, convert each of those triplets from binary into octal/decimal

$$\begin{aligned} 11_2 &= 1 \times 2^1 + 1 \times 2^0 = 3 \\ 010_2 &= 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2 \\ 001_2 &= 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1 \end{aligned}$$

So, that means

$$\underbrace{11}_{3}\,\underbrace{010}_{2}\,\underbrace{001}_{1}.$$

Wasn't that convenient? Now, if only we didn't have to do that math. Well, we don't. If we think beforehand and make a table of binary and octal digits we can see a pattern emerge:

Octal	Binary
7	111
6	110
5	101
4	100
3	011
2	010
1	001
0	000

So now it's a matter of grouping the binary digits into groups of three (starting from the right) and then replacing each group of three by the corresponding octal digit. Each binary triplet represents a pattern which corresponds to a single octal digit, like so:

$$011 = 3 \quad 010 = 2 \quad 001 = 1$$

Base Sixteen, Hexadecimal:

Octal was fine and a great advance over binary. It's easier to see what the number really is and it takes fewer digits, which means less writing or typing. But in these days of megabytes, gigabytes, and terabytes even octal is stressed. We need a bigger base. Continuing with our logic of powers of two we move on to 2^4, base sixteen, or **hexadecimal**.

Remember what was said about a higher base being able to represent large numbers with fewer digits? Well, base sixteen can accomodate a really large number, more than base ten. It converts from binary the same as octal, just in groups of four instead of eight. But it does present us with a slight problem. We only have symbols for numbers from 0 to 9, but hexadecimal goes past 9. Now what? We borrow from the alphabet and use letters **A** through **F**. Now a number can be comprised of digits and letters, which makes for interesting reading sometimes.

Hexadecimal	Binary
F	1111
E	1110
D	1101
C	1100
B	1011
A	1010
9	1001
8	1000
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

Just like in base ten, two, and eight, there is no symbol for 16 in hexadecimal. The "digits" only go up to 15. This means that 10_{16} is actually one group of 16 and zero in the one's place.

Following the same theme as we did with octal, we can convert from binary to hexadecimal by grouping binary digits together. But now, instead of groups of three ($8 = 2^3$) we use groups of four ($16 = 2^4$). So that binary number we were working with earlier, 11010001 can be converted into hexadecimal like this:

$$1101 = D \qquad 0001 = 1$$

So our number is $D1$ in hexadecimal.

Addition with Binary Numbers:

Adding numbers together in binary is really quite similar to adding numbers in decimal (base ten), only you have to remember that $1 + 1$ does not equal 2 but 10_2. Just like adding five and five results in a carry, adding one and one results in a carry in base two.