Hexadecimal Numbers

CS106E, Young

Hexadecimal Numbers are often used in Computer Science to represent binary numbers. We'll run into them in all sorts of places including Memory Addresses (this lecture), Network Addresses (next week), and HTML Colors (in two weeks). So let's take a quick look at how they work, and why Computer Scientists use them so much.

What is a Hexadecimal Number

Hexadecimal is based on 16, whereas as we've previously seen Binary is based on 2 and Decimal is based on 10. As we don't actually have 16 digits to use for Hexadecimal, we use the regular Decimal digits 0-9 and then add the letters A, B, C, D, E, and F; this gives us 16 total digits. Colloquially computer scientists refer to Hexadecimal as just "Hex".

Converting from Binary to Hexadecimal (and Back)

Because 16 is a power of 2, converting from Binary to Hexadecimal and back is very easy, whereas converting from Binary to Decimal and back is kind of a pain (especially for very large numbers). Every four bits correspond to a Hexadecimal Digit, and a Byte corresponds to two Hexadecimal Digits.

Let's take a look at how to convert from binary to hex. ***The important point for our class purposes is not that you learn how to actually do the conversion, rather it's to help you see that this is pretty straightforward and to have an understanding of why Computer Scientists use it all the time***, and why they often prefer to use hexadecimal to decimal.

When converting sets of 4-bits to Hexadecimal, you can either work it out manually, remembering that each binary digit is a power of 2:

0111_2 for example is $1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 4 + 2 + 1 = 7_{10}$

For 0-9 the conversion will be the same for both decimal and hexadecimal, but for numbers between 10-15, you'll either need to remember that in Hex and A = 10, B = 11, C = 12, D = 13, E = 14, and F = 15, or you can use a table like the one shown at right.

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

The real benefit to hex becomes apparent when working with larger binary numbers.

Suppose I have the binary number:

10110110

I break this into sets of 4 bits:

1011 0110

and then convert these individually either doing it manually or using the table and I get

B6

So 10110110 in binary can be rewritten as B6 in hexadecimal.

What about that MIPS machine instruction we saw last lecture?

00000001010010110100100000100000

Converting it to decimal would be a real mess. It would be:

$1x2^{24} + 1x2^{22} + 1x2^{19} + 1x2^{17} + 1x2^{16} + 1x2^{14} + 1x2^{11} + 1 \times 2^{5}$

Yuck. Converting to Hex on the other hand is easy. Just break the number up into groups of 4 and look them up in the table or work through each set of four.

0000 0001 0100 1011 0100 1000 0010 0000

0 1 4 B 4 8 2 0

014B4820

So 00000001010010110100100000100000 is 014B4820 in hexadecimal.

Because I never have to work with a group of bits larger than four, it's actually fairly easy to work them out manually without the table. There's no way I would do the decimal conversion of a binary number this large manually.

For conversion from hexadecimal back to binary you just reverse the process. Each hexadecimal digit gets converted to its equivalent set of four bits, write down all the sets of four bits and you have your binary number. In contrast converting from decimal-to-binary is even more painful than going from binary-to-decimal, so I'm not going to show it here.

Hexadecimal is preferred to decimal because there's a direct correspondence between the hexadecimal digits and their matching binary digits that just isn't there with decimal digits.

Hexadecimal is preferred to binary, because a hexadecimal number is much more compact than the binary equivalent, and because it's very easy to get a long sequence of 0s and 1s messed up.

Other Hexadecimal Conventions

There are several ways to indicate that something is hexadecimal.

- I might write a Hexadecimal number simply as 2F5C trusting my reader will recognize it as a hexadecimal number.
- Alternatively I could use the previous mathematical notation we saw, indicating the base of a number using a subscript: $2F5C_{16}$ However, writing subscripts isn't something I can do when writing source code for a computer program.
- Within a program, many computer languages allow a programmer to indicate that a value is hex instead of decimal by preceding it with an "0x" – for example: 0x2F5C.

For a casual computer user, you'll most often run into this when the computer crashes and you see an error message showing 0x followed by a sequence of hexadecimal digits. This is probably the memory address at which the program crashed (although it could also be a hexadecimal error code, indicating the type of error).