Binary, Octal and Hexadecimal Numbers

CMSC 201 Section 60
October 28, 2024

Administrative Notes

Project 1 is due Friday - November 1

Next week is exam week, which means:

- No discussion NEXT week
- Sample exam 2 will be out later this week

Note: material from today's lecture WILL be included on this exam

Material from Wednesday's lecture WILL NOT be included on this exam

- Wednesday's lecture will be a recording on my YouTube channel
- Class will NOT meet in person Wednesday, October 30 - I'll be out of town

Number bases

As humans, we tend to count things by powers of 10: 10, 100,1000,..

We have only ten symbols, or digits, used to count

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Why? Because most humans have ten digits - fingers, or toes - and that made it easy to keep track of things

While that's the best way for humans to count, it's not the best way for machines to count

Why not?

Most computers (and similar machines) are made up of electrical switches.

Electrical switches have two states: ***ON*** and ***OFF***

We can say that if something is ON it has the value 1 and if it is OFF it has the value 0.

This is called “binary” numbering. We only need two digits, 0 and 1, to represent all numbers, using powers of 2. So, there are only two binary digits

“***Bi***nary digi***t***” was a pain to say all the time, so it got shortened to … “bit”

There are 10 types of people in the world - those who understand binary and those who don't.

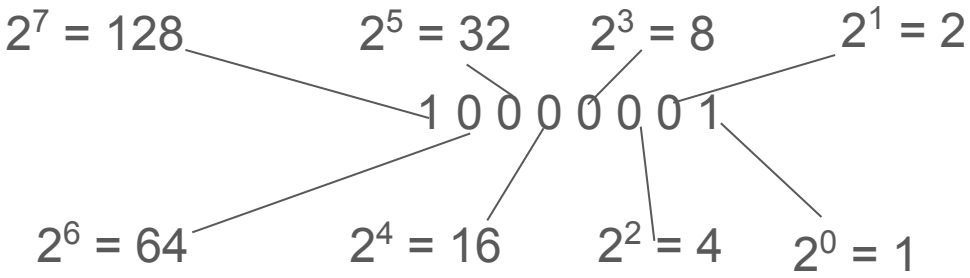
```
147355  - 5 * 10**0 + 5 * 10**1 + 3 * 10**2 + 7 * 10**3
+ 4 * 10**4 + 1 *10**5
```

A binary number

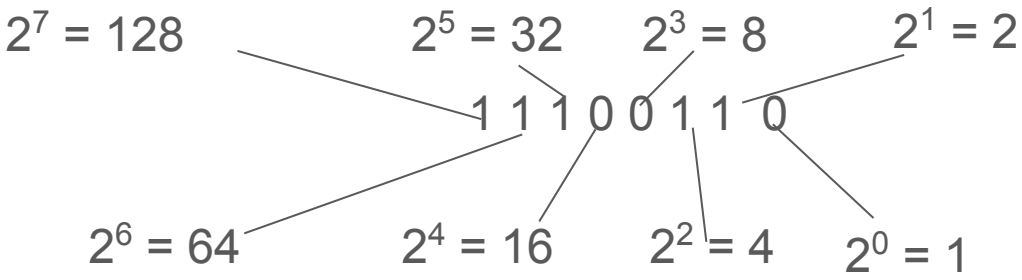
1 0 0 0 0 0 0 1 in base 10 = 129

What's it's value in human-speak i.e. decimal?

Places mean the same as in decimal - the "least significant" is on the right and each place to the left is one power of two greater



Another example



2 +4 + 32 +64 + 128 = 230

Binary Math - just like decimal math (mostly)

Addition: start on the right; add the numbers in the column and carry when needed

$$\begin{array}{rr} & 1010 \\ + & 110 \\ \hline & 10000 \end{array}$$

Subtraction: start on the right; borrow from the next place over when needed

1010

- 101

101

Multiplication: go column by column; add products

So how do you convert from decimal to binary?

Suppose I gave you the decimal number 10,000,001 and asked you to tell me what it is in binary?

1001

You have to subtract powers of two from it until you get zero.

Huh?

10,000,001 - 8,388,608 = 1,611,393 - 1,048,576 =562,817

What is the largest power of two that is less than 10,000,001?

n	2^n	n	2^n	n	2^n
1	2	6	64	22	4,194,304
2	4	10	1024	23	8,388,608
3	8	15	32768	24	16,777,216
4	16	20	1,048,576		
5	32	21	2,097,152		

Converting decimal to binary:

10,000,001 decimal: write down 1 in the 2^{23} column, subtract 8,388,608, from 10,000,001 = 1,611,393. The largest power of 2 that's less than that is 2^{20}. Write down zeroes in the next two columns, and then put a 1 in the 2^{20} column. Subtract 1,611,393 - 1,048,576 = 562,817.

Keep doing this, and you eventually wind up with:

100110001001011010000001

Fortunately, Python makes this somewhat easier

To indicate that a number represents a binary number, preface it with “0b”

```
x = 0b110111011
```

Tells python you mean this to be the binary number, NOT the decimal number 110,111,011

To convert decimal numbers to Binary, use the built-in Python function “bin”.

```
i = 54

bin_i = bin(i)
```

As before, we can convert (almost) anything to an integer - base 10 - by using int()

Awkward! That's big number and hard to deal with

Luckily, there's another numbering system: base 16, or "hexadecimal"

- "hex" for short

We need 16 different symbols to represent hexadecimal "digits"

We have 0,1,2,3,4,5,6,7,8,9 - that's 10. We need six more. So we use the first six letters of the alphabet.

A in hex = 10 in decimal; B = 11; C = 12; D = 13; E = 14 and F = 15. That's all we need, because 16 in decimal is 10 in hex. We go by powers of 16

Converting Decimal to Hex

The algorithm is similar to binary

Fortunately for us, there's a built-in function in Python3 that does the conversion for us

hex(int) takes a decimal integer value and returns its hexadecimal value

0x means a hexadecimal number

0x1234 to decimal = 4 * 1 + 3 * 16 + 2 * 16**2 + 1 * 16**3
4 *1 + 3 *16 + 2 * 256 + 1 * 4096

Ox ab123

3 * 16 to the 0 power (3 * 1)

2 * 16 to the 1 power (3 * 16 = 48)

1 * 16 to the 2nd power (2 * 256 = 512)

B or 11 * 16 to the 3rd power (11 * 4096 = 45056)

A or 10 * 16 to the 4th power (10 * 65,536 = 655,360)

3 + 48 + 512 + 45,056 + 655,360 =

Converting binary to hexadecimal

This is actually easier - and more important

Four binary digits - four bits - together represent one hexadecimal digit, because four bits can represent exactly 16 different values

So you simply replace four bits with their hex equivalent

1001 1000 1001 0110 1000 0001

Start from the left - 1001 = 9 hex; 1000 = 8 hex; 1001 = 9 hex; 0110 = 6 hex; 1000 = 8 hex; 0001 = 1 hex. So the binary string above, converted into hex, is:

0x989681

A Handy Conversion Table

Binary	Hex	Decimal	Binary	Hex	Decimal
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

Octal is base 8

Octal digits are 0,1,2, 3, 4, 5, 6, and 7

Octal numbers are represented in Python as 0o(numbers)

That’s a 0, then a lower-case o

Representation and examples

In python - and most other languages - hex numbers are prefaced with “0x” to indicate that they are NOT decimal numbers

Let’s use the built-in hex() function to do some conversions

Built in functions

hex() - convert from current form to hexadecimal

- Can take input as decimal, binary or octal

bin() - convert from current form to binary

- Can take input as decimal, octal or hexadecimal

There is no built-in function to convert to Octal

If you had to write a program to do this yourself, how would you do it?