Binary Numbers and Machine Representation of Data

Garrett Dancik, PhD

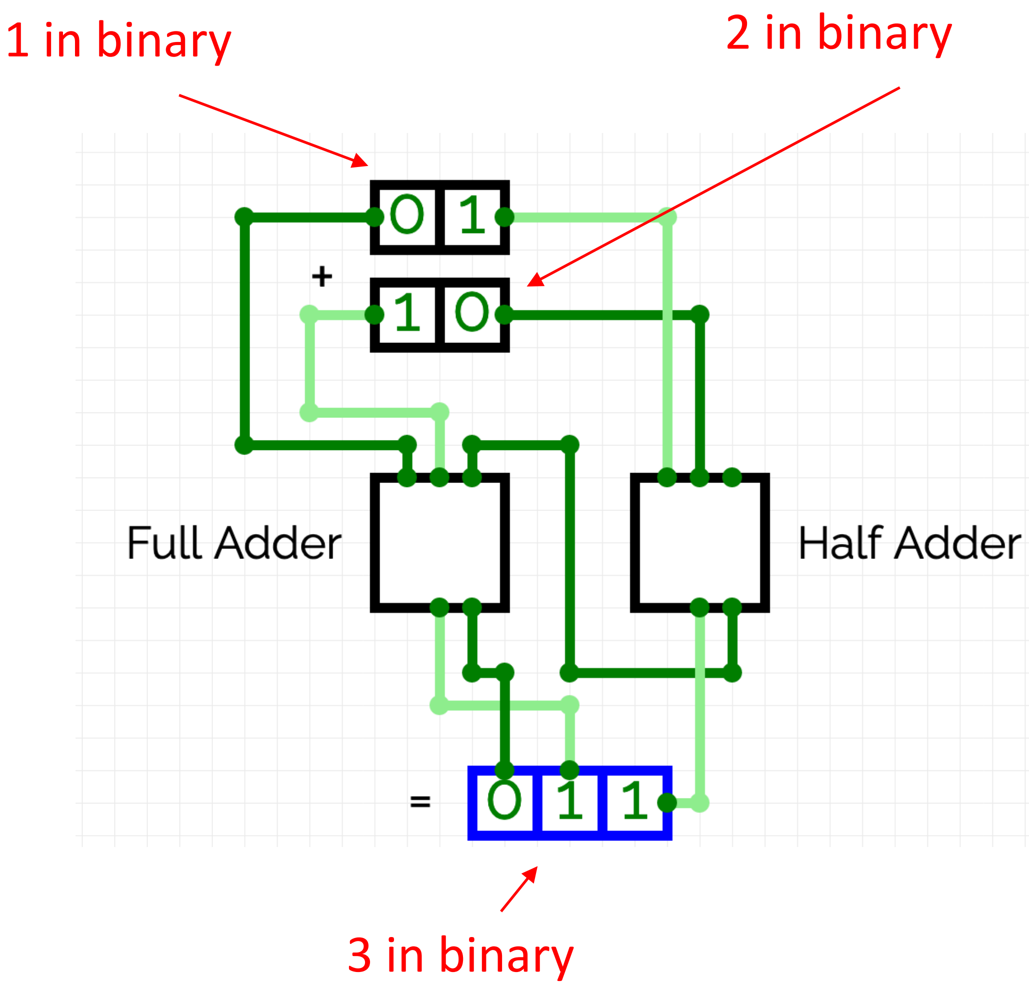
Fall 2021

Course Notes: https://gdancik.github.io

Bigger Picture

```
In [1]: x = 1 + 2
        x
Out[1]: 3
```

Computer code



Digital Logic

The decimal number system

- Decimal number system:
  - Base 10
  - 10 digits (0 – 9), each digit represents a power of 10
  - Example: 524

5 2 4

$5 \times 10^2 +$

$2 \times 10^1 +$

4×10^0

$$
\begin{array}{r}
500 \\
+\ 20 \\
+\ 4 \\
\hline
524
\end{array}
$$

The binary number system

- Binary number system:
  - Base 2
  - 2 digits (0 – 1), each digit represents a power of 2
  - Example: 101

1 0 1

$1 \times 2^2 +$

$0 \times 2^1 +$

1×2^0

$$\begin{array}{r} 4 \\ +\ 0 \\ +\ 1 \\ \hline 5 \end{array}$$

Powers of 2

n	2^n
0	$2^0 = 1$
1	$2^0 = 2$
2	$2^0 = 4$
3	$2^0 = 8$
4	$2^0 = 16$
5	$2^0 = 32$
6	$2^0 = 64$
7	$2^0 = 128$
8	$2^0 = 256$

Binary conversion example:

Binary	1	0	1	1	0	0	1	1	0
Calculation	1×2^8	0×2^7	1×2^6	1×2^5	0×2^4	0×2^3	1×2^2	1×2^1	0×2^0
	256	0	64	32	0	0	4	2	0
Sum	358								

What is the following binary number in decimal:
1011

The hexadecimal number system

- Hexadecimal number system:
  - Base 16
  - 16 digits (0 – 9, A - F), each digit represents a power of 16
  - Example: 101

1 A 9

$1 \times 16^2 +$

$10 \times 16^1 +$

9×16^0

$$\begin{array}{r} 256 \\ +\ 160 \\ +\ 9 \\ \hline 425 \end{array}$$

Addition and Counting

Decimal:

```
     1 ← carry
    18
+    3
  ----
    21
```

Binary Addition Rules:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (carry a 1)
```

Binary:

```
  11  ← carry
1011
  11
----
1110
```

What is 1011 + 11 in decimal notation?

Counting:

Decimal	Binary	Binary (Padded)
0	0	000
1	1	001
2	10	010
3	11	011
4	100	100

Relationship between binary and hexadecimal

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

We can represent a binary number as hex by translating each group of 4 binary digits to its single hex value, moving from right to left

<table>
<tr><th>Binary</th><th>1</th><th>0</th><th>1</th><th>1</th><th>0</th><th>0</th><th>1</th><th>1</th><th>0</th></tr>
<tr><td>Hex</td><td>1</td><td colspan="4">6</td><td colspan="4">6</td></tr>
</table>

We can therefore write the binary number 101100110 in hex as 166.

What is the following binary number in hex:
1010011110

Decimal to binary conversion

- Because a binary digit specifies a number as the sum of powers of 2, we can continually divide by 2, and use the remainder to determine the binary digit from right to left, stopping when the quotient is 0.
- Intuition: If a decimal number *N* is even, then N % 2 is 0, and the last binary digit is 0; if N is odd, then N %2 is 1, and the last binary digit is 1.
- Example: Find the binary representation of the number 13

Calculation	Quotient	Remainder
13 / 2	6	1
6 / 2	3	0
3 / 2	1	1
1 / 2	0	1
Stop		

Read reminders from last to first to get the binary number

In binary, the decimal number 13 is:

1101

Characters are represented on machines using binary

- A character (A, B @, 9, \, etc) value is displayed by interpreting the binary value using the specified *encoding* standard
  - ASCII codes: https://www.rapidtables.com/code/text/ascii-table.html
    - Uses 7 bits for each character (2^7 = 128 possible characters)
    - https://mothereff.in/binary-ascii
  - Unicode: https://www.rapidtables.com/code/text/unicode-characters.html
    - Represents over 140,000 characters in many languages, and is encoded in different ways (the most common is UTF-8); encoding determines how characters are stored in binary
- Even though characters are stored as binary values on a computer, we often use unicode, hexadecimal or decimal values to specify them in a more human-readable way.

Prefixes for bits and bytes

- 1 bit can store 2 values (0 and 1)
- 1 byte = 8 bits, and can store 2^8 = 256 values (in general, n bits can store 2^n values)

Prefix (SI)	Decimal (SI)
kilo (k)	1000^1 = 1,000
mega(M)	1000^2 = 1,000,000 (1 million)
giga (G)	1000^3 = 1,000,000,000 (1 billion)
tera (T)	1000^4 = 1,000,000,000,000 (1 trillion)

Prefix (IEC)	Binary	Prefix (Memory)
kibi (Ki)	1024^1 = 1,000	kilo (K)
mebi (Mi)	1024^2 = 1,048,576	mega (M)
gibi (Gi)	1024^3 = 1,073,741,824	giga (G)
tibi (Ti)	1024^4 = 1,099,512,000,000	tera (T)

- Storage considerations
  - An audio file requires about 1 MB per minute of sound
  - 1 full length movie is about 2 GB
  - 1 GB can hold about 16 hours of music
  - A photo (from an iphone) requires around 2 MB
  - Over 1,000 photos are uploaded to Instagram every second (https://www.internetlivestats.com/one-second/#instagram-band)
    - This requires roughly 2 GB every second. If your computer has 1TB of space, at this rate you would run out of memory in a little over 8 minutes.

Signed numbers (signed/magnitude notation)

- The left-most bit is used to represent the sign as positive (0) or negative (1).
- Suppose we use 4 bits for the signed number (1 sign bit, and 3 bits for the number). How many numbers can we store?

	Sign bit			
Binary	**1**	**1**	**0**	**1**
Calculation	-	1×2^2	0×2^1	1×2^0
	-	4	0	1
Sum		- 5		

	Sign bit			
Binary	**0**	**1**	**0**	**1**
Calculation	+	1×2^2	0×2^1	1×2^0
	+	4	0	1
Sum		+ 5		

Problems with signed/magnitude notation

- There are two zeros, +0 and –0 (e.g., 1000 and 0000)
  - This can cause problems if you need to check whether a value is 0.
- Addition is more complicated, because it depends on the signs of the numbers
  - For numbers with the same sign, add the numbers, ignoring the sign bit, then prepend the sign bit to the answer.
  - But this does not work if numbers have different signs

Carry bit

Sign bit	1	1		Decimal
1	0	1	1	- 3
1	0	0	1	-1
1	1	0	0	-4

Carry bit

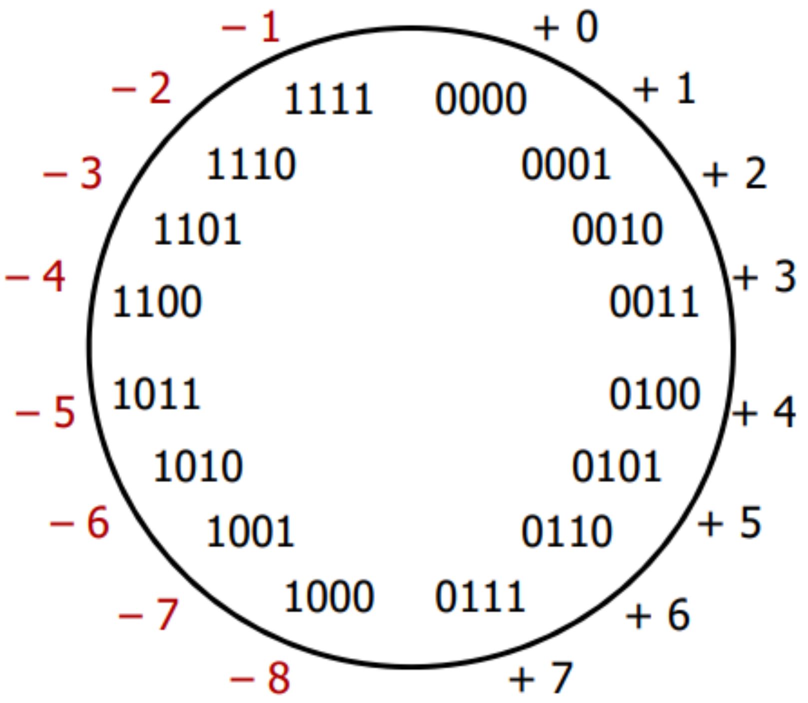
Sign bit	1	1		Decimal
1	0	1	1	- 3
0	0	1	1	+3
?	1	1	0	0

This is not equal to 0!

Two's complement

There is only 1 zero!

- Most common representation of signed integers
- The sum of a number and its two's complement is 2^N, where N = number of bits
- Assuming 4 bits, the two's complement of 0011 is 1101 because they sum to 10000 (which is 2^4)



https://cs.carleton.edu/faculty/awb/cs208/s20/topics/topic5.html

Carry bit

1	1	1	1	
0	0	0	1	1
0	1	1	0	1
1	0	0	0	0

- For positive numbers, use standard binary representation
- For negative numbers, use the two's complement of the positive value, which can be found by inverting the number (flipping from 0s to 1s and vice-versa), and adding 1

Two's complement

- Find the two's complement of +3 (0011)

Number	0	0	1	1
Invert	1	1	0	0
Add One	**1**	**1**	**0**	**1**

The two's complement of 0011 (+3) is 1101 (-3)

- Addition of bits works for positive and negative numbers!

Extra carry bit is ignored

Carry bit

1	1	1	1		Decimal
	0	0	1	1	+3
	1	1	0	1	-3
1	0	0	0	0	0

- For N bits, two's complement provides a range of numbers between -2^{N-1} and $2^N - 1$
- In Java, an int (integer) type uses 4 bytes (32 bits) of memory and stores integers in the range $-(2^{31})$ and $2^{31} - 1$, which is between

  -2,147,483,648 and 2,147,483,647