



Datatypes I



Outline

- Binary-Decimal Conversion
- Arithmetic Operations on Bits
 - Addition & Subtraction
- Trouble In Paradise (Don't Be Negative...)
- Representations & Numbers
- A New Hope... (2's Complement)
- Binary-Decimal Conversion (revisited)
- Sign Extension and Overflow

Representations

- Modern computers effectively store patterns of switches that are set On and Off.
- How do we do work within that limitation?

“When I use a word,’ Humpty Dumpty said in rather a scornful tone, ‘it means just what I choose it to mean — neither more nor less.’

‘The question is,’ said Alice, ‘whether you can make words mean so many different things.’

‘The question is,’ said Humpty Dumpty, ‘which is to be master — that’s all.’”

— Lewis Carroll, *Through the Looking Glass*

- In that vein, we **choose** representations for numbers, letters, and many other things that are “easy” to work with.

Representations & Numbers

What does this mean to you?

5

The number 5?

An Arabic numeral (digit) that **represents**
the **number** 5?

Could there be other representations?

0101_2 , V, ⋮

Representations & Numbers

And what about

$$\sqrt{2}$$

Can you write that number accurately using Arabic numerals?

No? Then how are you going to store it in a computer memory?

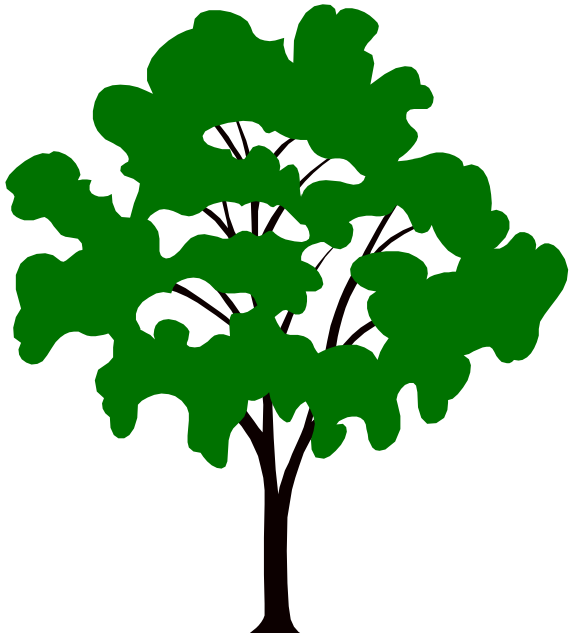
Data Types

- A **data representation** is the set of values from which a variable, constant, function, or other expression may take its value and includes the meaning of those values. A representation tells the compiler or interpreter how the programmer intends to use it. For example, the process and result of adding two variables differs greatly according to whether they are integers, floating point numbers, or strings.
- Patt: "We say a particular representation is a **data type** if there are operations in the computer that can operate on information that is encoded in that representation."

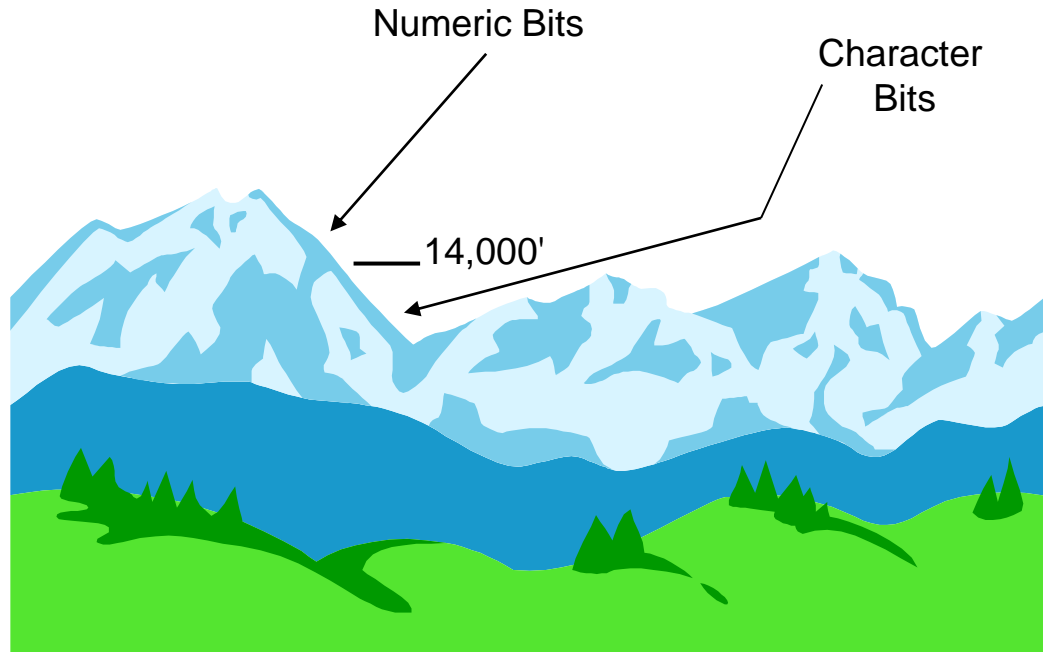
Question

- What is a bit?
- “Bit” is short for “binary digit”
- A bit can take on exactly two values
- We often refer to those values as 0 and 1

Where do bits come from?



Bitaba Tree



Andes Mountains



NOT!



Where do bits come from?

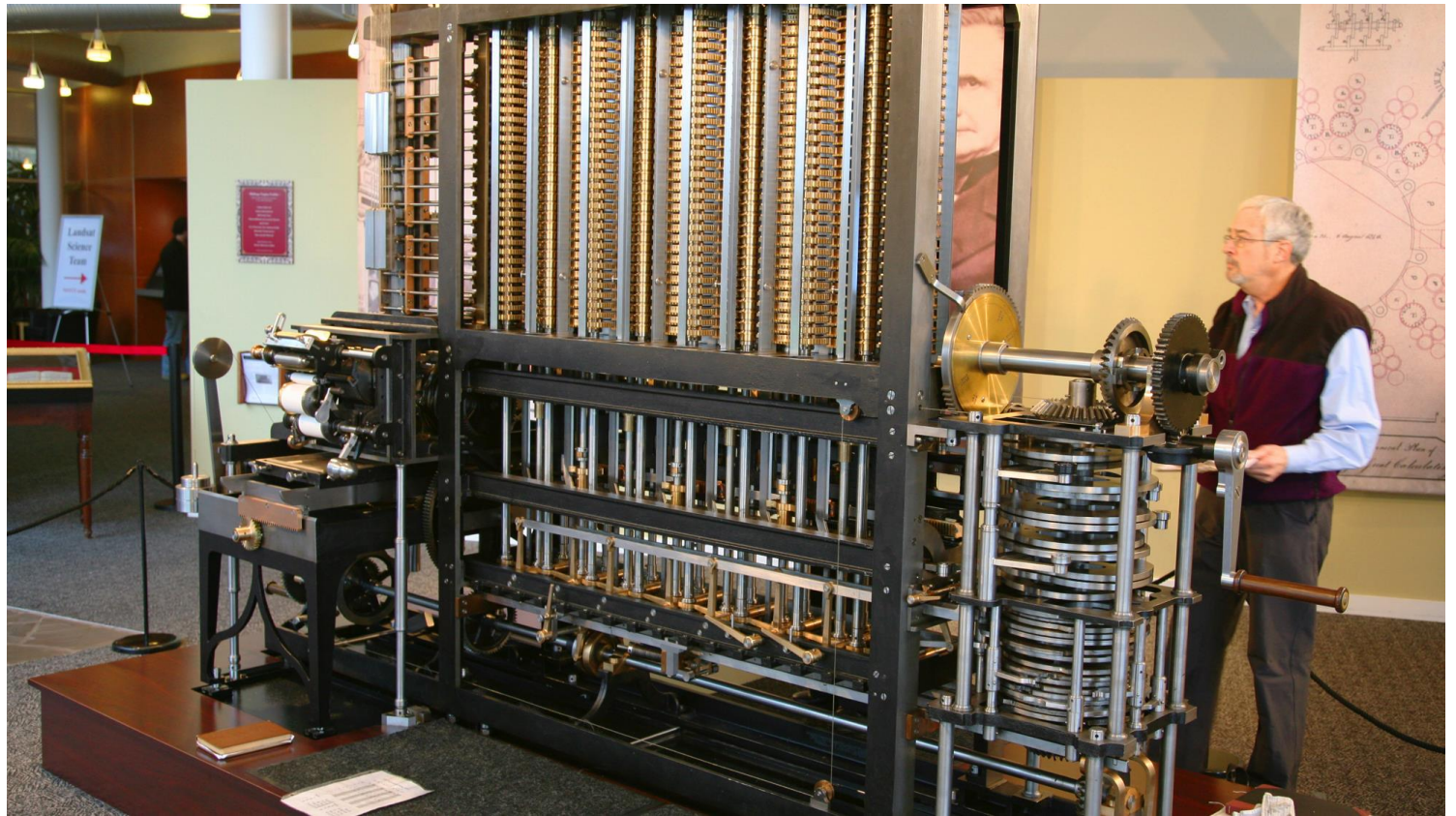
- A. Bitaba Tree
- B. Andes Mountains
- C. Above 14000 feet
- D. None of the above


Today's number is 21100

Question

- The first computers did decimal arithmetic. Couldn't we just skip binary?

Babbage Difference Engine #2



- 
- Difference Engine #2 Video
 - Method of Finite Differences Video
(Math starts at 1:35)

Decimal or Binary?

- History has pretty well determined that binary is the the most effective way to store and calculate using today's electronics.
- Only computers in the 40s and 50s attempted to actually store data in decimal.
- Up through the 70s, computers binary-coded decimal instructions (store in coded binary, calculate in decimal).
- Decimal storage is very rare in modern computers.

Question

- What is the smallest number of bits we would need to designate 5 different items or states?
- 3 ($2^2 \leq 5 < 2^3$)
- How about 16 states?

Unsigned Integers

- How many different numbers can be represented by 4 bits?
- 2^4
- How many different numbers can be represented by 7 bits?
- 2^7
- How many different numbers can be represented by n bits?
- 2^n

Let's Represent Non-Negative Numbers Using Bits

0	0 1 0 0
1	1 0 0 1
2	1 1 0 0
3	0 0 0 1
4	1 0 1 1
5	0 0 1 1
6	1 1 1 0
7	0 1 0 1
8	0 1 1 0
9	0 0 0 0
10	1 1 0 1
11	0 1 1 1
12	1 0 0 0
13	1 1 1 1
14	1 0 1 0
15	0 0 1 0

Is this a good choice?

How many other choices are there?

$$16! = 20,922,789,888,000$$

Maybe we can pick a better representation?

Perhaps We Should Take Advantage of a Pattern?

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Quick Review of Base 2 Representation

- There are only 2 values in each place: 0 and 1.
- We assign place-values just like base 10, except they are powers of 2 instead of powers of 10

128	64	32	16	8	4	2	1
-----	----	----	----	---	---	---	---

0	1	0	0	0	1	0	0
---	---	---	---	---	---	---	---

$$0100\ 0100_2 = 68_{10}$$

128	64	32	16	8	4	2	1
-----	----	----	----	---	---	---	---

0	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---

$$0011\ 1111_2 = 63_{10}$$

Memorize Powers of 2

2^0 1

2^1 2

2^2 4

2^3 8

2^4 16

2^5 32

2^6 64

2^7 128

2^8 256

2^9 512

2^{10} 1,024

2^{11} 2,048

2^{12} 4,096

2^{13} 8,192

2^{14} 16,384

2^{15} 32,768

2^{16} 65,536

Hmm...

Looks Like Place Value in Binary Arithmetic

0	0 0 0 0	0+0+0+0
1	0 0 0 1	0+0+0+1
2	0 0 1 0	0+0+2+0
3	0 0 1 1	0+0+2+1
4	0 1 0 0	0+4+0+0
5	0 1 0 1	0+4+0+1
6	0 1 1 0	0+4+0+1
7	0 1 1 1	0+4+2+1
8	1 0 0 0	8+0+0+0
9	1 0 0 1	8+0+0+1
10	1 0 1 0	8+0+2+0
11	1 0 1 1	8+0+2+1
12	1 1 0 0	8+4+0+0
13	1 1 0 1	8+4+0+1
14	1 1 1 0	8+4+2+0
15	1 1 1 1	8+4+2+1

Binary to Decimal Conversion

➤ Here's a place to practice!

➤ <https://games.penjee.com/binary-numbers-game/>

N = 4		Number Represented	
Binary	Unsigned		
0000	0		
0001	1		
0010	2		
0011	3		
0100	4		
0101	5		
0110	6		
0111	7		
1000	8		
1001	9		
1010	10		
1011	11		
1100	12		
1101	13		
1110	14		
1111	15		

Addition and Subtraction

- Work the same way as base 10. Carry if the bit value is > 1 ; borrow if less than 0.

$$\begin{array}{r} 0000\ 1111_2 \\ +0001\ 0001_2 \\ \hline 0010\ 0000_2 \end{array}$$

$$15 + 17 = 32$$

$$\begin{array}{r} 11\ 0000_2 \\ - 00\ 0011_2 \\ \hline 10\ 1101_2 \end{array}$$

$$48 - 3 = 45$$

Practice Problems

➤ What is $5 + 2$?

```

0101
+0010
-----
0111 = 7
    
```

➤ What is $9 + 4$?

```

1001
+0100
-----
1101 = 13
    
```

➤ What is $10 - 3$?

	010	10
1010	1010	1010
-0011	-0011	-0011
-----	-----	-----
	1	0111 = 7

➤ What is $2 - 6$?

0010	0110
-0110	-0010
-----	-----
	-0100 = -4

Trouble In Paradise

- We are used to dealing with positive and negative integers
- Then we'd better find a way to represent signed integers
- What are our options?
- One possible approach: reserve one bit to represent the sign of a number

Signed Integers

- Signed magnitude
 - Easy to understand
 - Fun to be with
 - Not so good with hardware

N = 4		Number Represented	
Binary	Unsigned	Signed Mag	
0000	0	0	
0001	1	1	
0010	2	2	
0011	3	3	
0100	4	4	
0101	5	5	
0110	6	6	
0111	7	7	
1000	8	-0	
1001	9	-1	
1010	10	-2	
1011	11	-3	
1100	12	-4	
1101	13	-5	
1110	14	-6	
1111	15	-7	

Signed integers

- 1's complement
 - Used in some early computers
 - To make a negative just flip all the bits

N = 4	Number Represented		
Binary	Unsigned	Signed Mag	1's Comp
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	-0	-7
1001	9	-1	-6
1010	10	-2	-5
1011	11	-3	-4
1100	12	-4	-3
1101	13	-5	-2
1110	14	-6	-1
1111	15	-7	-0

Nagging Concerns...

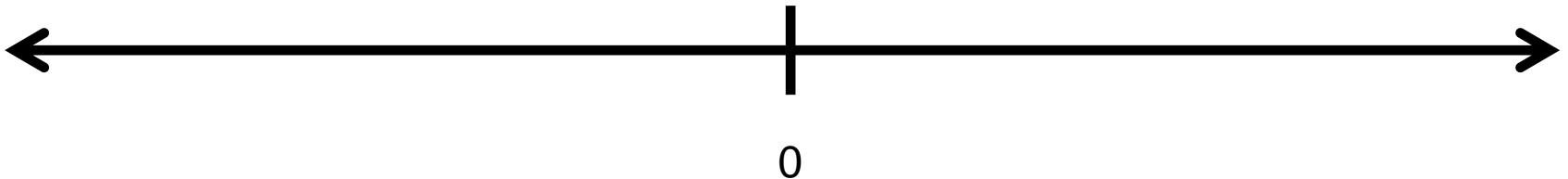
- Why do I have to think about two different operations: addition and subtraction?
- How much time will I spend examining the signs and magnitudes of each operand to determine how to find the answer?
- And what about plus zero and minus zero representing the same number?
- Is there a better way?...

Reality Sets In

- Any arbitrary scheme of allowing different bit patterns to represent different numbers could be made to work, but there is an especially clever choice...

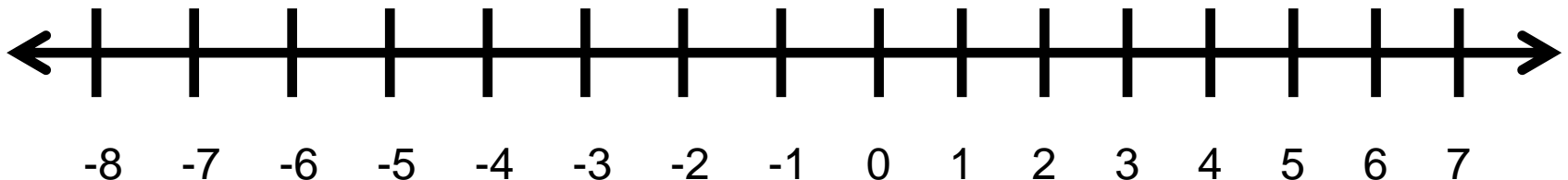
A New Hope...

➤ Start with the number line



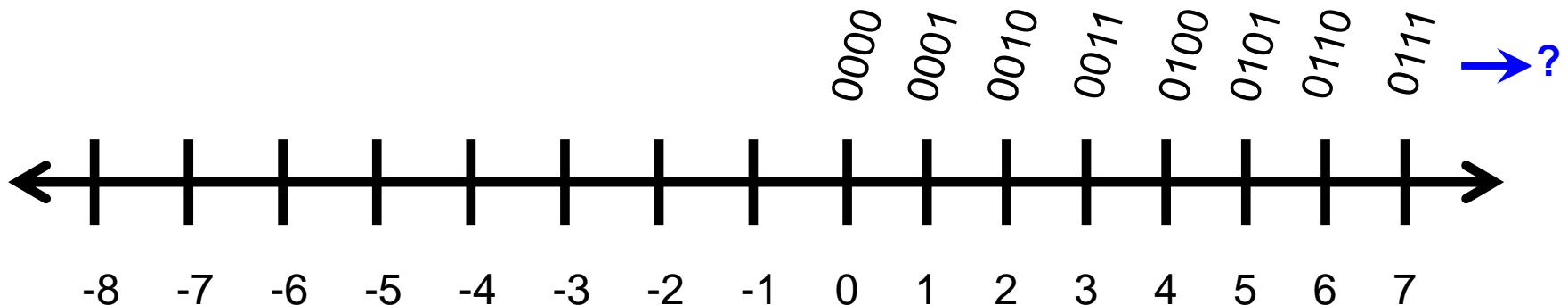
A New Hope...

- Let use 4 bits for our example representation
- This gives us $2^4 = 16$ different values that we can represent
- Let's split them as evenly as possible between positive and negative values



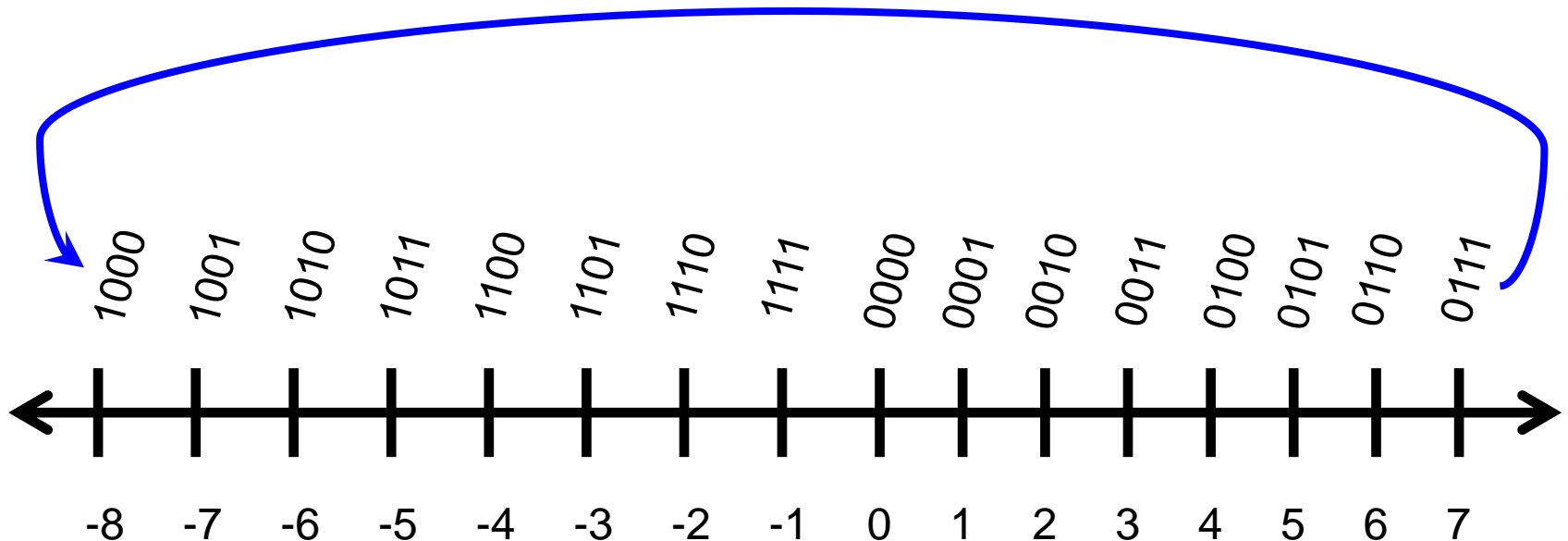
A New Hope...

- Now, start numbering from 0 using our trusty unsigned binary representation
- But... what happens when we reach the “end of the line” ?
(hint: think odometer)



A New Hope...

- Wrap around and keep going!
- Remember: We're stuck with modular arithmetic – we can't represent all numbers



N = 4	Number Represented				
Binary	Unsigned	Signed Mag	1's Comp	2's Comp	
0000	0	0	0	0	
0001	1	1	1	1	
0010	2	2	2	2	
0011	3	3	3	3	
0100	4	4	4	4	
0101	5	5	5	5	
0110	6	6	6	6	
0111	7	7	7	7	
1000	8	-0	-7	-8	
1001	9	-1	-6	-7	
1010	10	-2	-5	-6	
1011	11	-3	-4	-5	
1100	12	-4	-3	-4	
1101	13	-5	-2	-3	
1110	14	-6	-1	-2	
1111	15	-7	-0	-1	

...for Signed integers

- 2's complement
 - Used in almost all computers manufactured today
 - Allows for low cost, high performance circuitry to do math operations
 - There's no need for a hardware subtractor, just a bitwise complement
 - Why does it work?

Useful Properties of 2's Comp


- There is only one zero
- Addition still works as expected from binary arithmetic (if you don't wrap)
- The first bit still indicates the sign (as long as you will tolerate zero being positive)
- Finding the complement (additive inverse) of a number is easy
- Since finding the complement is easy, there's no need for subtraction: just complement and add

Two's Complement Representation

- Note there's a useful new pattern:
 - You can find the two's complement of any number by flipping the bits and adding 1.
 - And a two's complement conveniently represents the additive inverse of the originally represented number!
- Repeat the Mantra (negating a number):
 - Flip the bits
 - Then add one

Question

How does one compute the additive inverse of a two's complement number?

- A. Take the Boolean NOT of each bit
- B. Take the Boolean NOT of the high-order bit
- C. Take the Boolean NOT of each bit and subtract 1
-  D. Take the Boolean NOT of each bit and add 1

Today's number is 33,777

Signed Binary to Decimal Conversion

- Given a negative number in two's complement notation how do we convert it to decimal?
 - Flip the bits
 - Then add one
 - Convert to decimal
 - The number is the negative of that decimal num

1. Consider the four-bit two's complement number: 1011
2. Flip the bits: 0100
3. Add 1: 0101
4. Convert to decimal: +5
5. So the binary two-s complement 1011 represents -5 in decimal

Which ones to know

- Be able to do math with:
 - Unsigned
 - Two's Complement
- Only understand the concept of:
 - Signed magnitude
 - One's complement
 - (You don't have to do math with these.)

More Practice Problems

➤ What is $5 + 1$?

➤ What is $5 - 2$?

➤ What is $5 - (-2)$?

➤ What is $5 + -5$?

More Practice Problems

$$5 + 1$$

	0001
5	0101
1	0001
--	----
6	0110

$$5 - 2$$

Convert to $5 + (-2)$

	1100
5	0101
-2	1110
--	----
3	0011

More Practice Problems

$$5 - (-2)$$

Convert to $5 + 2$

	0000
5	0101
2	0010
--	----
7	0111

$$5 + (-5)$$

	1111
5	0101
-5	1011
--	----
0	0000

Questions?



Question

What is the sum of 101111_2 and 001010_2 ?

(Interpret as unsigned whole numbers.)

A. 56



B. 111001_2

C. 100101_2

D. 47

010100

101111

001010

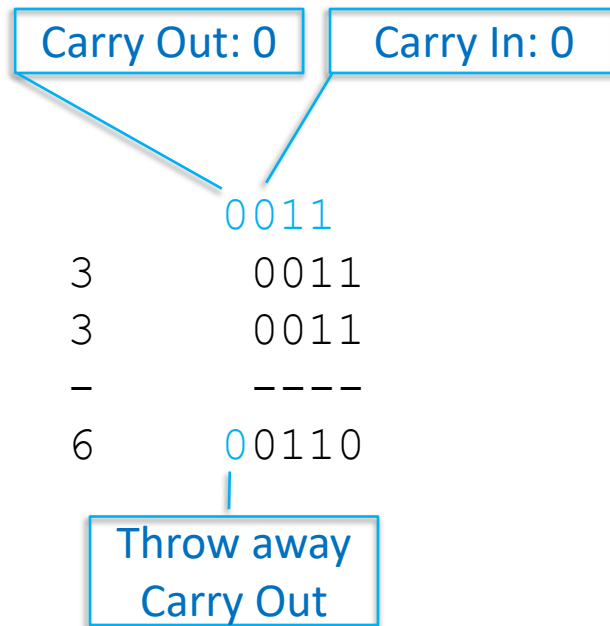
$111001 = 32+16+8+1 = 57$

Overflow (2s comp addition)

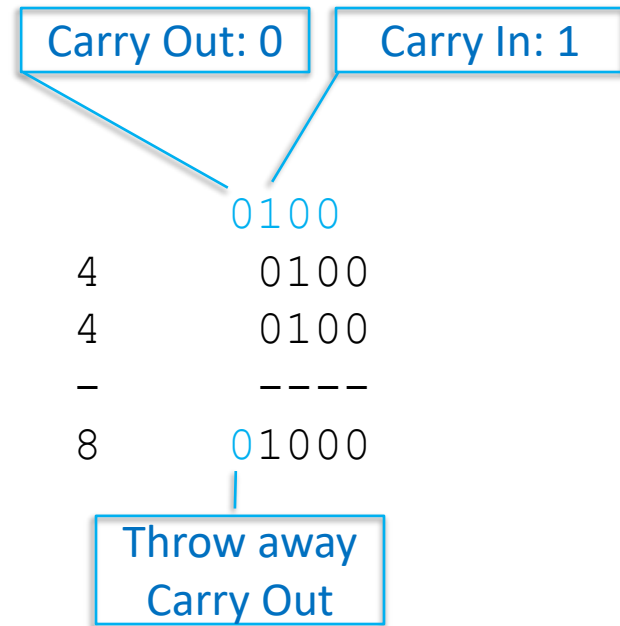
- Check carry in and out of the leading digit
 - (most significant digit, on the left)
- Add two numbers; we have problems if
 - We get a carry into the sign bit but no carry out
 - We get a carry out of the sign bit but no carry in
- If we add a positive and a negative number we won't have a problem
- Assume 4 bit numbers (-8 : +7) for some examples

Overflow

- If we add two positive numbers and we get a carry into the sign we have a problem



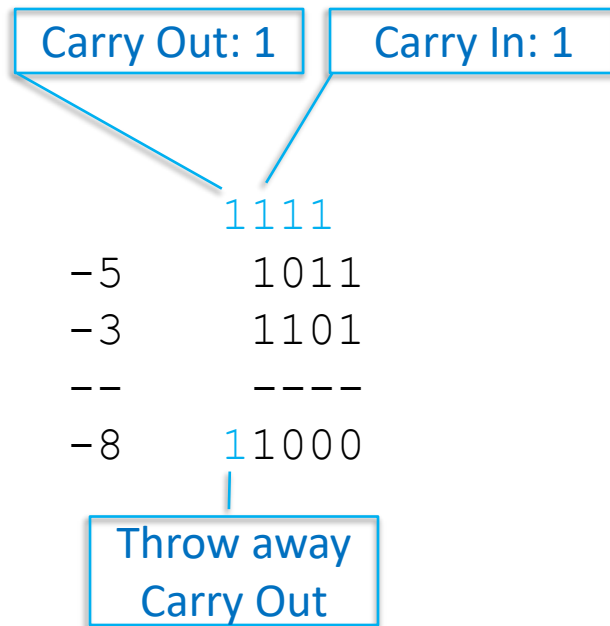
No Overflow



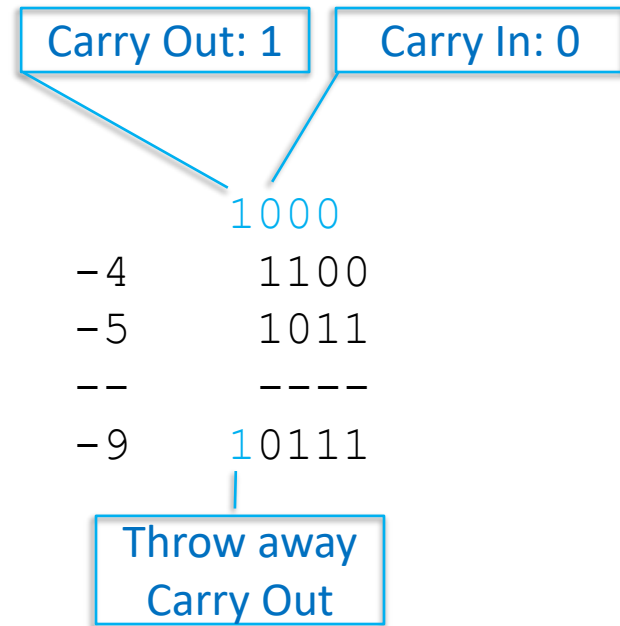
Overflow

Overflow

- If we add two negative numbers and we don't get a carries into and out of the sign bit we have a problem



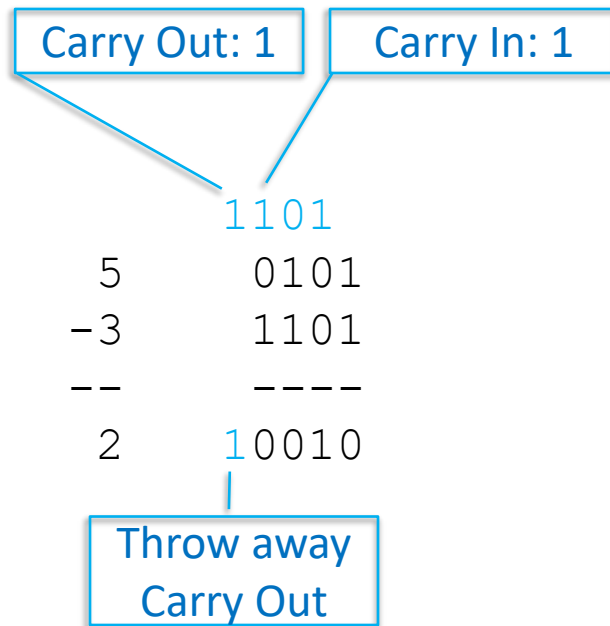
No Overflow



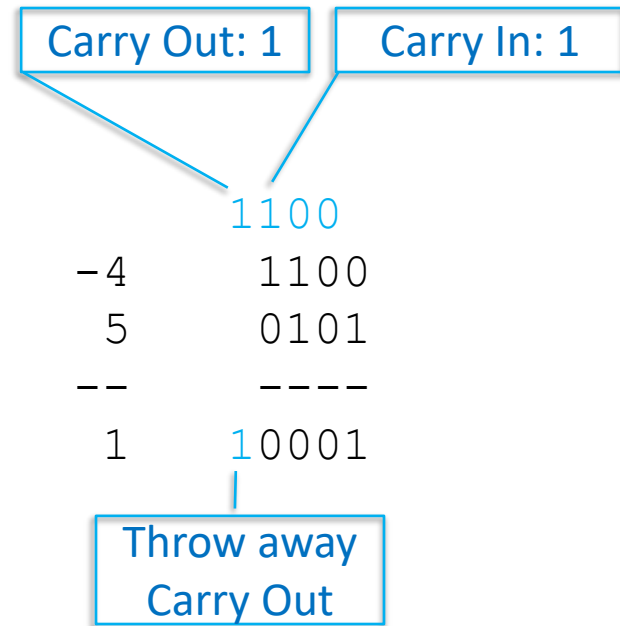
Overflow

Overflow

- If we add a positive and a negative number we won't ever have a problem



No Overflow



No Overflow

Sign Extension

- Suppose we have a number which is stored in a four bit register
- We wish to add this number to a number stored in an eight bit register
- We have a device which will do the addition and it is designed to add two 8 bit numbers
- What issues do we need to deal with?

Sign Extension

- To add bits to the left of a two's complement number (i.e. make the number of bits bigger while representing the same numeric value)...
- Fill the new bits on the left with the value of the sign bit!
- This is a characteristic of two's complement that you have to remember!

Sign Extension – First Attempt

➤ Add 3 and 64 (0011 and 01000000)

$$\begin{array}{r} 0011 \\ +0100\ 0000 \\ \hline 0100\ 0011 \end{array}$$

Correct!

➤ Add -3 and 64 (1101 and 01000000)

$$\begin{array}{r} 1101 \\ +0100\ 0000 \\ \hline 0100\ 1101 \end{array}$$

WRONG!

Sign Extension Example

➤ Add 3 and 64 (0011 and 01000000)

```
      0011
+0100 0000
-----
0100 0011
```

```
  0000 0011
+0100 0000
-----
0100 0011
```

➤ Add -3 and 64 (1101 and 01000000)

```
      1101
+0100 0000
-----
0100 1101
```

```
  1111 1101
+0100 0000
-----
10011 1101
```

Throw away the
carried out 1

What happens when we add a number to itself?

```
00010000
00010000
-----
00100000
```

```
00011111
00011111
-----
00111110
```

```
00010001
00010001
-----
00100010
```

```
00000101
00000101
-----
00001010
```

```
01110011
01110011
-----
11100110
```

```
01010101
01010101
-----
10101010
```

```
00011100
00011100
-----
00111000
```

```
00000001
00000001
-----
00000010
```

$A + A$ is the same as $2 * A$ is the same as $A \ll 1$

Fractional Binary Numbers

What is the place value of the first digit to the right of a decimal point?

$$10^{-1}$$

What is the place value of the first bit to the right of a binary point?

$$2^{-1}$$

So, what is the decimal value of the following binary number?

$$\begin{array}{r} 1 \wedge 0 \ 1 \\ = 1.25_{10} \end{array}$$

8	4	2	1	^	.5	.25	.125	.0625
1	0	1	0	1	1	0	0	

$$1010 \wedge 1100 = 10.75_{10}$$

What base 10 number is represented by

$$101.011_2$$

A. 5.11

$$1 \ 0 \ 1 \ . \ 0 \ 1 \ 1$$

B. $4 \frac{5}{8}$

$$4+0+1+0+1/4+1/8$$



C. $5 \frac{3}{8}$

$$5 \frac{3}{8}$$

D. 5.08