Data Representation II

Lecture 3

Data Types

- Common Data Types
  - Integer
  - Real Numbers
  - Strings
  - Boolean
  - Memory Address

Integers

- Whole numbers
- E.g 12345
- Formed by
  - Natural numbers
    - Including zero
  - Negatives of non-zero nature number
- Subset of Real numbers

$\mathbb{Z}$

Real Number

- Represents a quantity along a continuous line
- Formed by
  - Rational numbers
    - -5, 5, ⅖
  - Irrational numbers
    - $\sqrt{2}$
    - π

$$\mathbb{R}$$

Strings & Boolean

- String
  - Sequence of characters
  - “hello!”

- Boolean
  - True / False
  - Also equal to 1 / 0

Memory Address

- **0xA09E1223**
- Fixed-length sequences of bits
- Unsigned integers
- Memory sizes tend to be power of 2
- Therefore hex
  - $16 = 2^4$
- Shorter and easier to convert from hex to binary and vice versa

Range of Data Types

Data Type	Size	Range
Integer types		
Boolean	1 bit (though often stored as 1 byte)	0 to 1
Byte	8 bits	0 to 255
Word	2 bytes	0 to 65535
Double Word	4 bytes	0 to 4,294,967,295
Integer	4 bytes	−2,147,483,648 to 2,147,483,647
Double Integer	8 bytes	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Real types		
Real	4 bytes	1E-37 to 1E+37 (6 decimal digits)
Double Float	8 bytes	1E-307 to 1E+308 (15 decimal digits)

Mathematical Prefixes

SI Prefixes										
Name	yotta	zetta	exa	peta	tera	giga	mega	kilo	hecto	deca
Symbol	Y	Z	E	P	T	G	M	k	h	da
Factor	10^{24}	10^{21}	10^{18}	10^{15}	10^{12}	10^{9}	10^{6}	10^{3}	10^{2}	10^{1}
Name	deci	centi	milli	micro	nano	pico	femto	atto	zepto	yocto
Symbol	d	c	m	μ	n	p	f	a	z	y
Factor	10^{-1}	10^{-2}	10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}	10^{-21}	10^{-24}

Signed and Unsigned

- Integral types
  - E.g. Short, Integer, Word
- Unsigned
  - Capable of representing only non-negative integers
- Signed
  - Capable of representing negative integers as well

Why unsigned?

- Range is higher
- E.g
  - 8 bit signed
    - Minimum : -128
    - Maximum : +127
  - 8 bit unsigned
    - Minimum : 0
    - Maximum : 255
- Reason:
  - No matter signed or unsigned, it still goes down to 0s and 1s. With 8 binary bits, signed will result in losing 1 bit to represent positive/negative.

Binary Representations

- It's more than simply 10101101
- Excess notation
- Ones' complement
- Two's complement

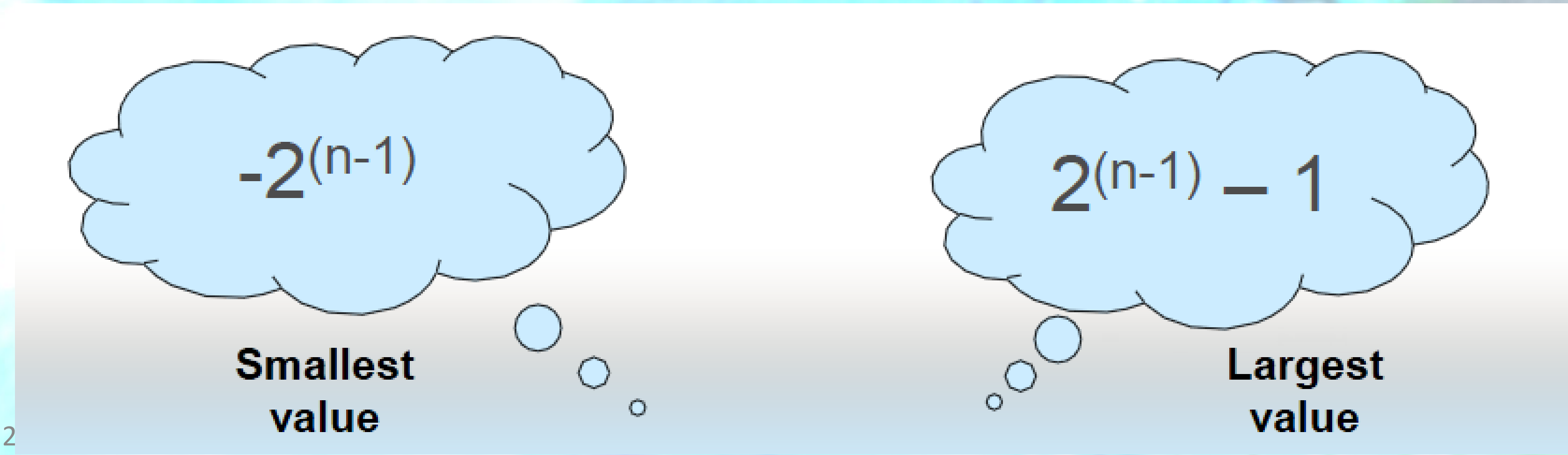
Lecture 4

Data Representation II

EXCESS NOTATION

Excess Notation

- Represent the sign of the number using the leftmost bit
- Must know how many storage bits are to be used



Excess Notation

- Has 2 parts
  - Most significant Bit, MSB
    - 1 for positive
    - 0 for negative
  - The "Rest"

- E.g **1** **0101010**
  - MSB is **1**
  - Rest is 0101010 which is the magnitude

Excess Notation Example

- Example 4 bits. Notice the Negative Values

Non-Negative Values			Negative Values		
Excess No	Bit Rep	Actual	Excess No	Bit Rep	Actual
15	<u>**1**</u>111	7	7	<u>**0**</u>111	-1
14	<u>**1**</u>110	6	6	<u>**0**</u>110	-2
13	<u>**1**</u>101	5	5	<u>**0**</u>101	-3
12	<u>**1**</u>100	4	4	<u>**0**</u>100	-4
11	<u>**1**</u>011	3	3	<u>**0**</u>011	-5
10	<u>**1**</u>010	2	2	<u>**0**</u>010	-6
9	<u>**1**</u>001	1	1	<u>**0**</u>001	-7
8	<u>**1**</u>000	0	0	<u>**0**</u>000	-8

Deriving Excess Notation

- Step 1:
  - Decide number of bits used to represent the number
    - E.g 10 bits
- Step 2:
  - For any number within the range, add excess value to it
    - E.g 10 + 2^{10-1}= 522
- Step 3:
  - Convert it to binary
    - E.g 1000001010

Lecture 4

Data Representation II

ONES' COMPLEMENT NOTATION

Ones' Complement Notation

- Negative value is represented by inverting all the bits in the binary representation of the number
  - E.g 7 → 0000 0111

    -7 → 1111 1000
- Range is slightly reduced
  - E.g 8 bits can represent -127 to 127

Problems of Ones’ Complement

- There are 2 representation for 0
  - 0000 0000
  - 1111 1111

- End-around borrow
  - Occurs during subtraction

```
  0000 0110      6
- 0001 0011     19
===========   ====
1 1111 0011    -12    —An end-around borrow is produced, and the sign bit of the intermediate result is 1.
- 0000 0001      1    —Subtract the end-around borrow from the result.
===========   ====
  1111 0010    -13    —The correct result (6 - 19 = -13)
```

Lecture 4

Data Representation II

TWOS' COMPLEMENT NOTATION

Two's Complement Notation

- Leftmost bit → sign
  - Similar to excess notation
  - E.g. **1** 001 0100

Unsigned Example:

1	**0**	**0**	**1**	**0**	**1**	**0**	**0**
1×2^7	0×2^6	0×2^5	1×2^4	0×2^3	1×2^2	0×2^1	0×2^0
128	0	0	16	0	4	0	0

= 148

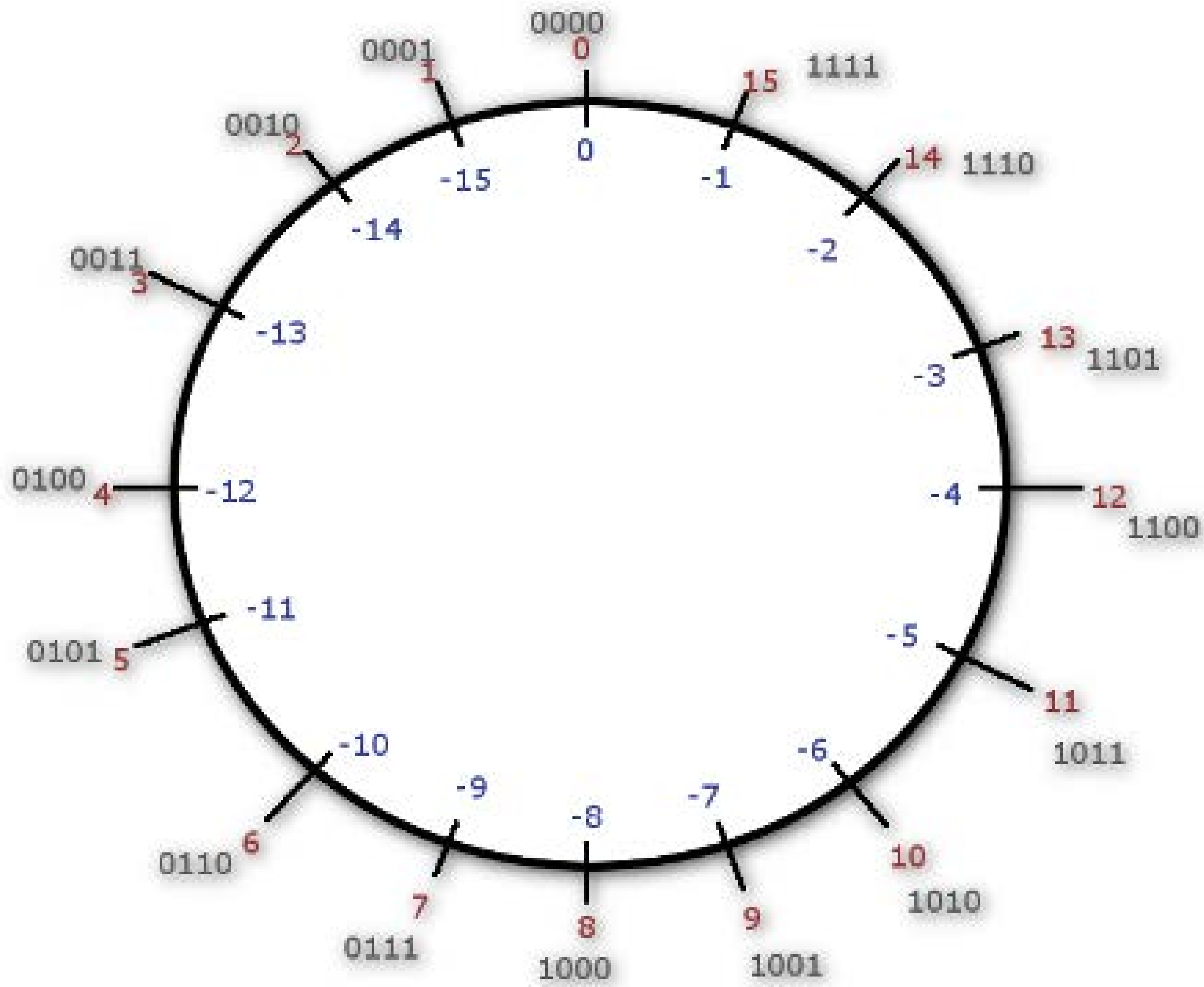
Signed 2's Complement Example:

1	**0**	**0**	**1**	**0**	**1**	**0**	**0**
-1×2^7	0×2^6	0×2^5	1×2^4	0×2^3	1×2^2	0×2^1	0×2^0
-128	0	0	16	0	4	0	0

= -108

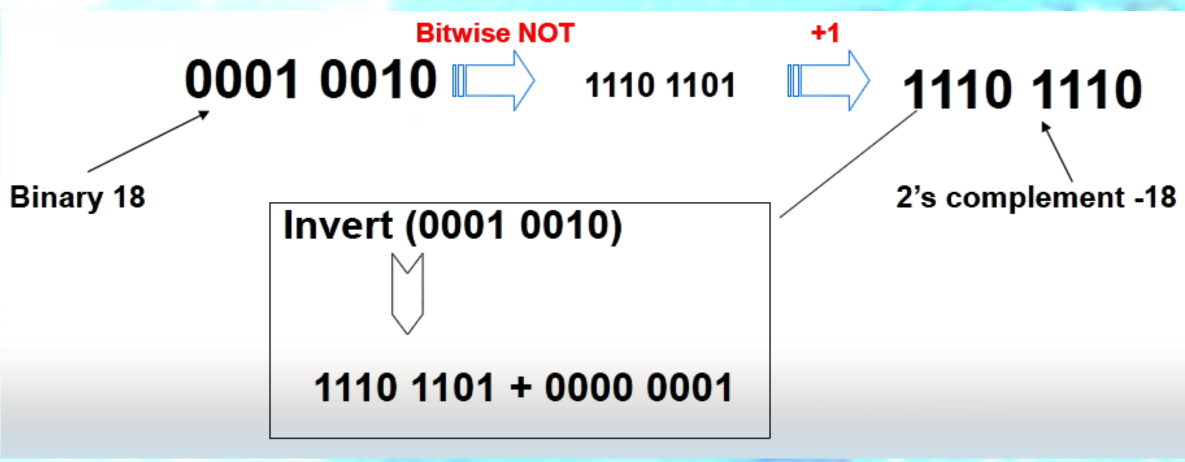
Two's Complement Notation

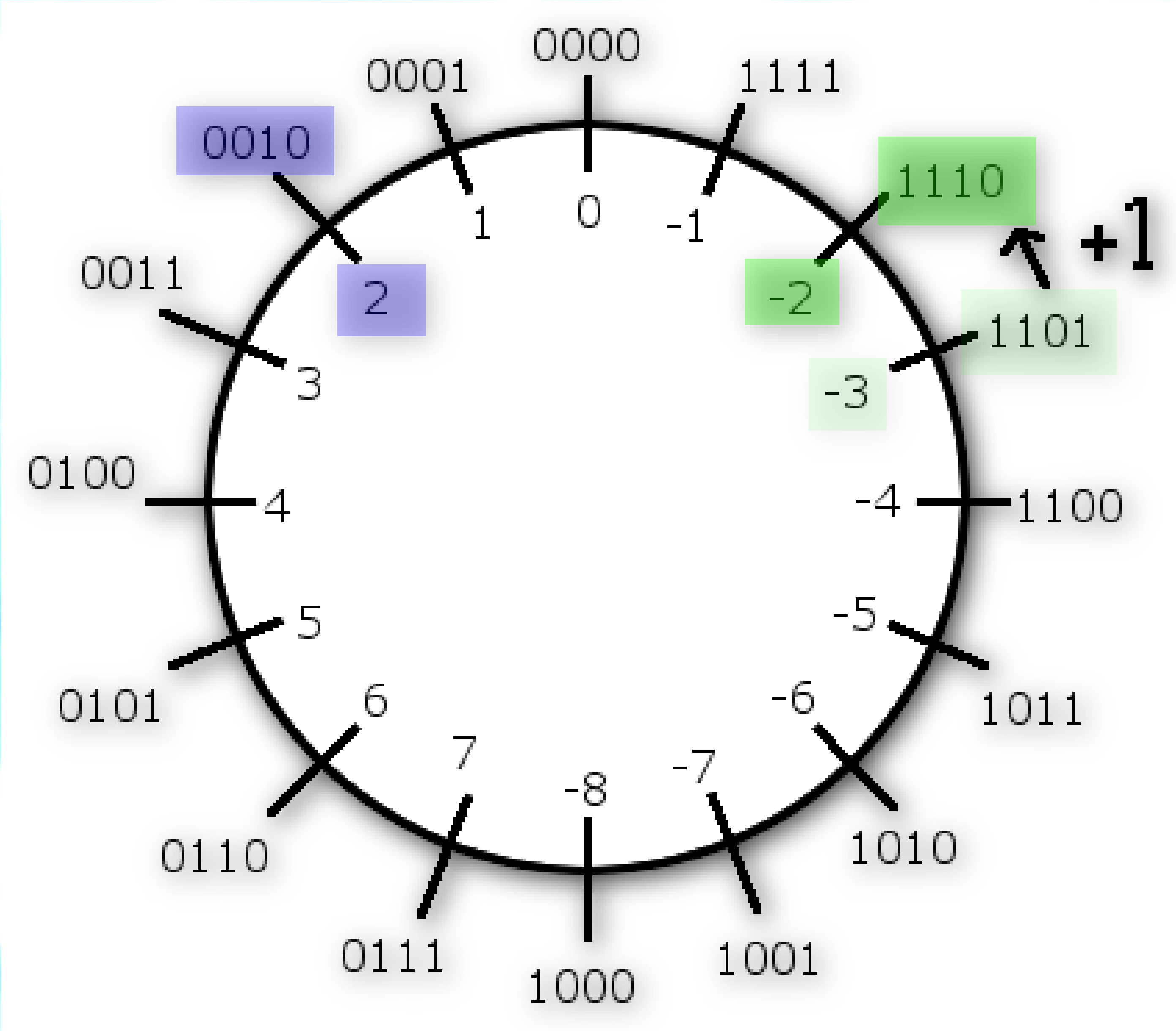
- Subtraction can be performed as addition of negative values
  - Saves on constructing additional hardware logic


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

Two's Complement of Negative Numbers

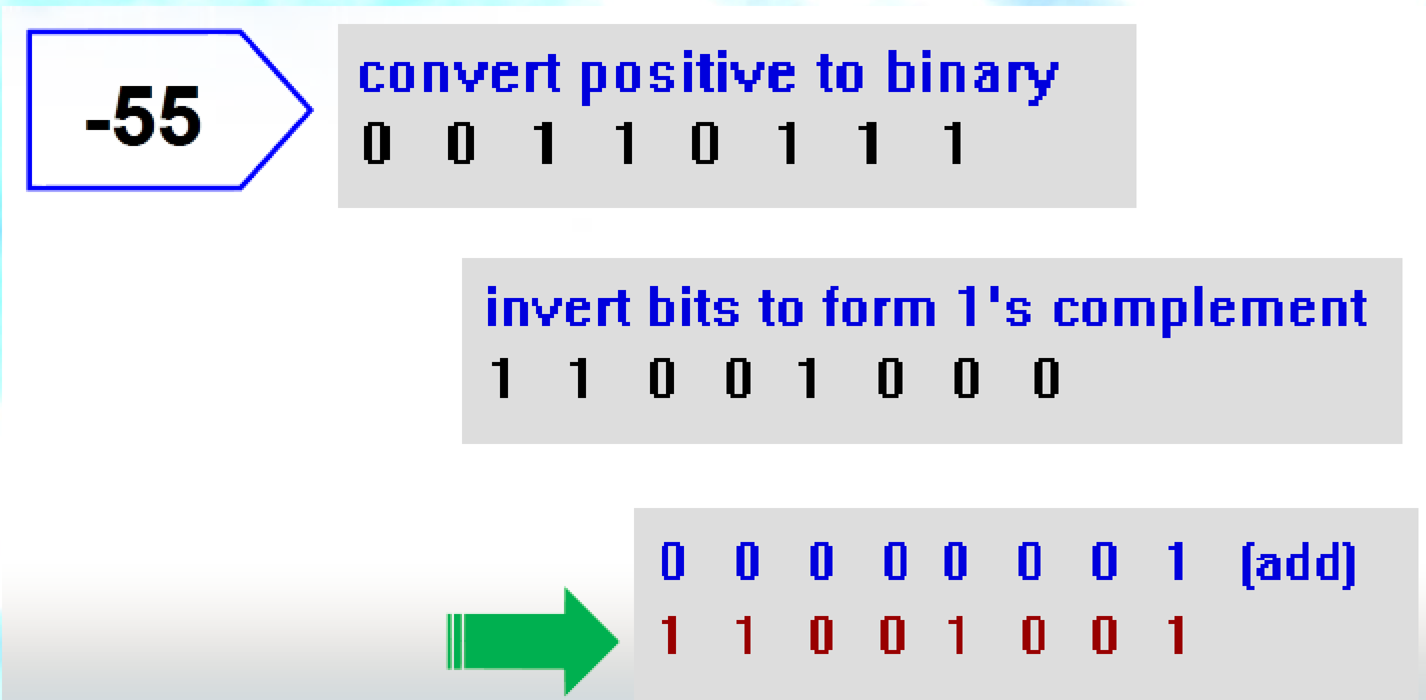
- Step 1:
  - Invert the positive binary equivalent of the number
- Step 2:
  - Add 1 to the result




0000
0
0001
1
0010
2
0011
3
0100
4
0101
5
0110
6
0111
7
1000
-8
1001
-7
1010
-6
1011
-5
1100
-4
1101
-3
1110
-2
1111
-1
+1

Conversion

- Example:
  - Convert -55 to 2’s complement



Addition & Subtraction

5 + 3 = 8	-7 - 3 = -10
1 1 1	
0 0101	1 1001
+ 0 0011	+1 1101
0 1000	11 0110

(-7) + (- 3) = -10 !!

Try this

- 10101 + 11110

Try this

- 10101 + 11110

```
  10101
  11110
 ------
 110011
```

step 1. 0+1 =1
step 2. 1+0 =1
step 3. 1+1 =0 carry1
step 4. 1+1+0 =0 carry 1
step 5. 1+1+1 =1 carry 1
step 6. 1+0+0 =1

Lecture 4

Data Representation

REAL NUMBERS

Real Numbers

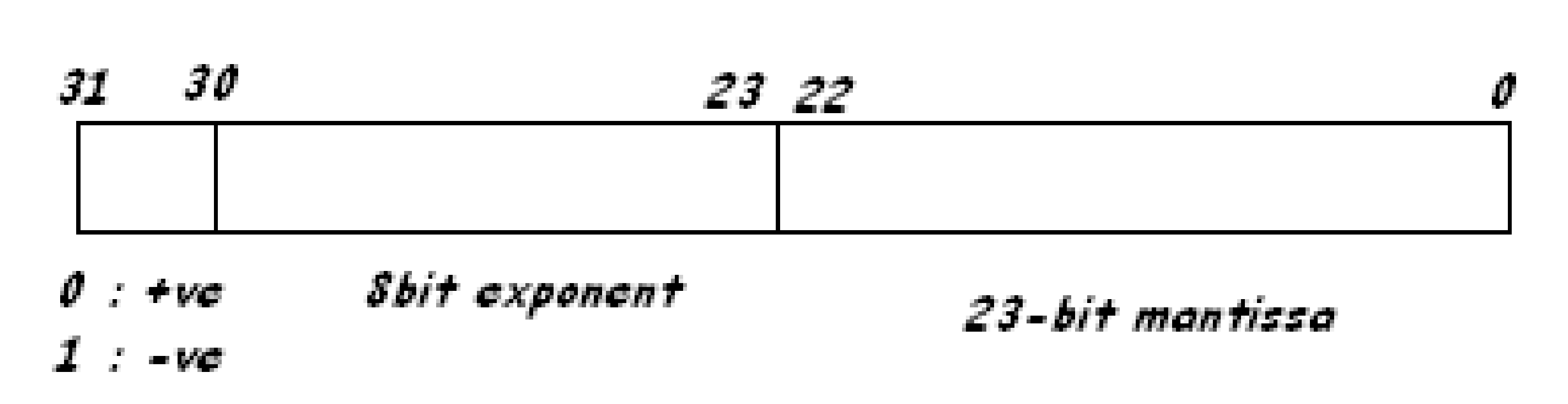
- 2 components
  - Whole number
  - Fractional component
  - E.g 4.6
    - The dot is what we called radix point
- Floating point notation
  - **Deals with tradeoff between range and precision**

Floating Point

- In Mathematics, we represent floating point using
  - Significant digits x base $^{\text{exponent}}$
  - E.g. 1.528535047×10^5

IEEE Standard 754 Floating Point Numbers

- Represent floating point
  - 32 bit numbers
  - 8 bit exponent
  - 23 bit mantissa (Significant)
  - 1 bit polarity



Example

- 1.**010111** x 2^{4}
  - Significant = **010111**
  - Exponent = **4**

- 1.**100100** x 2^{-8}
  - Significant = **1001**
  - Exponent = **-8**

Normalization

- In floating notation
  - $0.01 \times 10^1 = 0.1$
  - $1.00 \times 10^{-1} = 0.1$
- We do normalization
  - Expressing a number in scientific notation
  - Provide standard form of representation and to retain as many significant bits as possible for precision (to give better accuracy)

Normalization

- 10.625 → 1010.101
  → **1.010101×2^3** (Normalized)
- 0.375 → 0.011
  → **1.1×2^{-2}** (Normalized)

Converting decimals to binary

- Example 0.625
  - 0.625 x 2 = **1**.25
    - Therefore result = 0.**1**xxxx
  - Ignore whole number (0.25 x 2 = **0**.5)
    - Therefore result = 0. **10**
  - Ignore whole number (0.5 x 2 = **1**.0) ← result in 0 at the fraction hence end
    - Therefore result = 0. **101**
- Or you can consider it as
  - **1***(0.5) + **0***(0.25) + **1***(0.125)

Converting decimals to binary

- What about 0.622?
- It’s endless
  - We stop till we notice an infinite repeating pattern
    - **0.10011111001110110110010001011010**

- Try http://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html

Lecture 4

Data Representation II

BOOLEAN
(TO BE CONTINUED)