CSE 31
Computer Organization

Lecture 1 – Course Info
Number Representations

CSE 31: Fall 2018

- Lecturer
  - Chi Yan “Daniel” Leung
  - cleung3@ucmerced.edu
  - Office Room: AOA 126
  - Office Hours:
    - T/R: 10:00am – 12:00pm
    - W: 3:00pm – 4:00pm
    - By appointment
- TA
  - Pooya Tavallali ptavallali@ucmerced.edu
  - Satvik Kulshreshtha skulshreshtha@ucmerced.edu
- All email inquiries received before 5pm during school days will be replied within 48 hours
  - Please follow the guidelines below for proper email communications
    - https://cms.cerritos.edu/uploads/ifalcon/How_to_Email_your_Professor.pdf

Course Overview

- CatCourses
  - Check regularly for announcements.
  - Labs, Projects, and Reading/Homework Assignments (Linked to zyBooks) will be posted and submitted there.
  - Grades for assignments will also be found there (secure).
- 2 Lectures and 1 Lab per week
- 2 Mid-term exams (Sept. 26 and Nov. 7, in class)
- Final exam (Tuesday, Dec. 11, 11:30am, classroom)
- 10 lab assignments
- 6 homework assignments
- 2 projects

Course Objectives

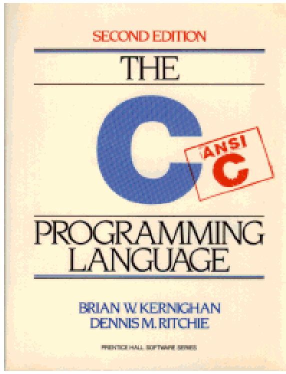
- Learning C
  - If you know one, you should be able to learn another programming language largely on your own
  - If you know C++ or Java, it should be easy to pick up their ancestor, C
- Assembly Language Programming
  - This is a skill you will pick up, as a side effect of understanding the Big Ideas
- Hardware design
  - We'll learn just the basics of hardware design

Course Policies

- Labs:
  - Giving each other help in finding bugs and in understanding the assignment is perfectly acceptable.
  - No late submission is allowed after 1 day beyond the due date.
  - Demo your work to your TA (or me) within ONE week after submission due date
  - Try to debug yourself before asking questions
  - Follow the guidelines (see below) to debug and ask your TA for help.
    - http://www.cplusplus.com/forum/articles/28767/
    - http://www.catb.org/esr/faqs/smart-questions.html

Course Material

- Text Books:
  - Computer Organization and Design from zyBooks
    - Sign up/sign in at zyBooks.com
    - Enter zyBook code: ***UCMERCEDCSE031LeungFall2018***
    - **You must subscribe your own copy. Participation grade will be partly evaluated based on the activities within the subscription account.**
  - The C Programming Language, Kernighan and Ritchie (K&R), 2nd edition



Prerequisites

- CSE30: Data Structures
- Math: logarithms, series, boolean logic, matrices, calculus …
- Coding: intermediate programming experience (Java, C, C++, …)
  - Coding in terminals??
- Curiosity: observe how the world is run by computers, and what problems we face.

Grading

- Homework: 10%
- Projects: 13%
- Lab assignments: 20%
- Mid-terms: 30%
- Final exam (comprehensive): 15%
- Participation (reading & labs): 12%
- Grades:
  - 90% of points at least an A
  - 80% at least a B
  - 70% at least a C

This is no curve in this class

Hints for success

- Attend lecture
- Read the textbook and do the activities
- Do & understand the labs and homework YOURSELF
- Create a portfolio to save all your work
- Take notes while reading and in lecture
- Ask questions

Policies

- Don’t copy someone else’s code
- Don’t give your code away
- Don’t outsource your assignments
- Don’t use electronic devices in exams
- Don’t use electronic devices during lecture for purposes other than note taking
- Turn off speakers/cellphone during class

No Cheating!

- Communicating information to another student during examination.
- Knowingly allowing another student to copy one's work.
- Offering another person's work as one's own.
- I am serious!

About me

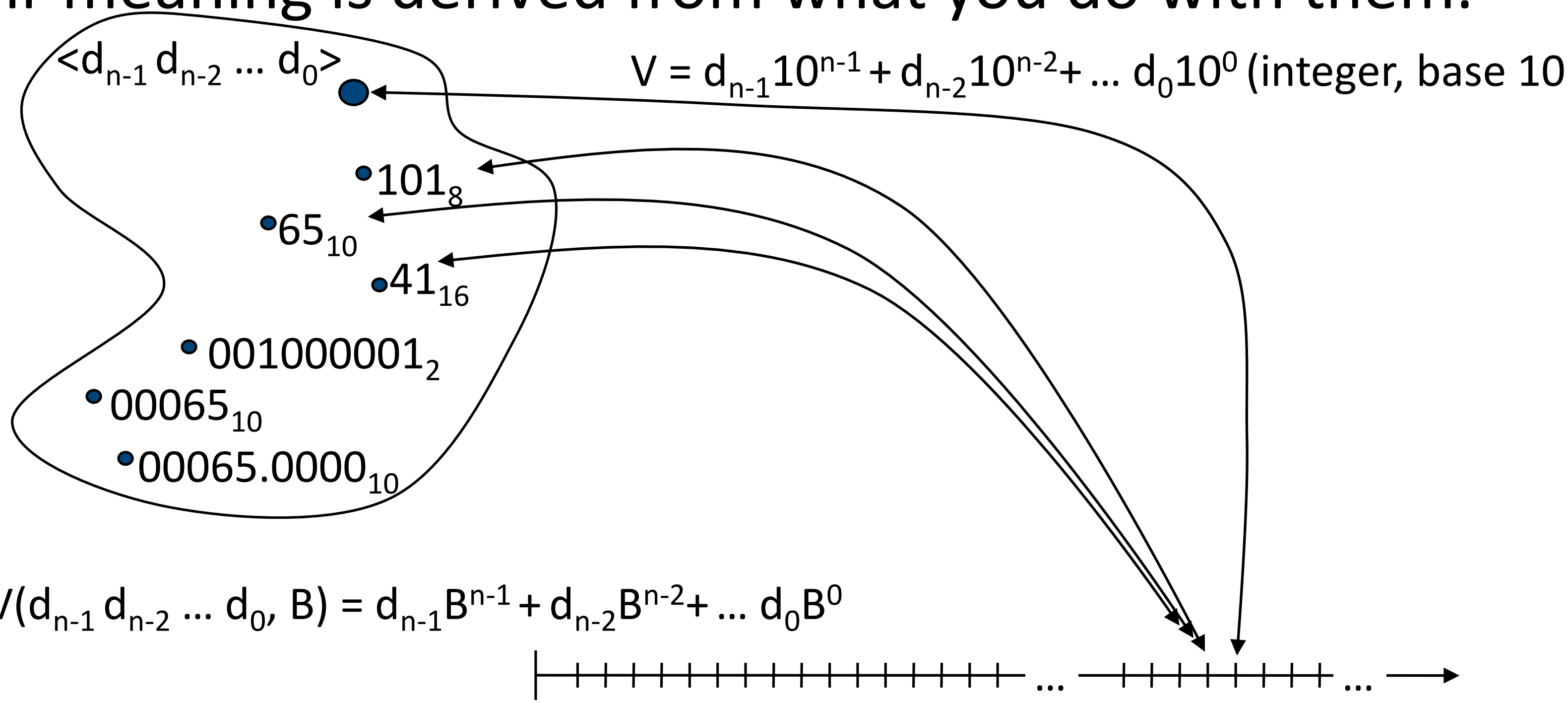
- Originally from Hong Kong
- B.S. at the University of Wisconsin, Madison
- M.S. at the California State University, Fresno
- PhD. at UCM
- Research interests: computer vision/image processing

Number Representations

- What do these numbers mean?
  - 101
  - 0101
- Depends on what representation!

Representation and Meaning

- Objects are represented as collections of symbols (bits, digits)
- Their meaning is derived from what you do with them.



Representation (how many bits?)

- Characters?
  - 26 letters → 5 bits ($2^5 = 32$)
  - upper/lower case + punctuation
    → 7 bits (in 8) ("ASCII")
  - standard code to cover all the world's languages → 8,16,32 bits ("Unicode") www.unicode.com
- Logical values?
  - 0 → False, 1 → True
- Colors? Ex: *Red (00)* *Green (01)* *Blue (11)*
- Remember: N bits → at most 2^N things

How many bits to represent π ?

a) 1
b) 9 (π = 3.14, so that's 011 . 001 100)
c) 64 (Since modern computers are 64-bit machines)
d) Every bit the machine has!
e) ∞

We are going to learn how to represent floating point numbers later!

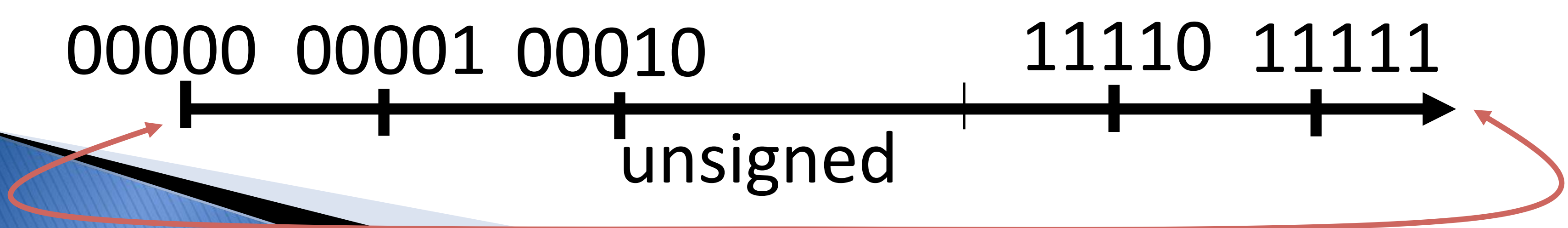
What to do with representations of numbers?

- Just what we do with numbers!
  - Add them
  - Subtract them
  - Multiply them
  - Divide them
  - Compare them
- Example: 10 + 7 = 17
  - …so simple to add in binary that we can build circuits to do it!
  - subtraction just as you would in decimal
  - Comparison: How do you tell if X > Y ?

$$
\begin{array}{ccccc}
 & 1 & 1 & & \\
 & 1 & 0 & 1 & 0 \\
+ & 0 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1
\end{array}
$$

What if too big?

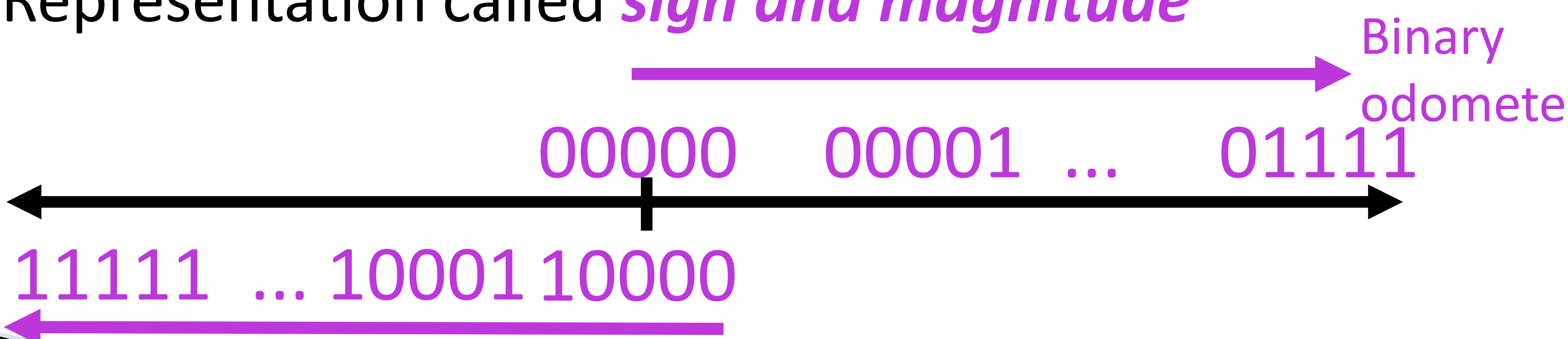
- Binary bit patterns above are simply representatives of numbers. Strictly speaking they are called “numerals”
- Numbers really have an ∞ number of digits
  - with almost all being same (00…0 or 11…1) except for a few of the rightmost digits
  - Just don’t normally show leading digits
- If result of add (or -, *, /) cannot be represented by these rightmost HW bits, ***overflow*** is said to have occurred.



Negative Numbers

- So far, ***unsigned numbers***

- Obvious solution: define leftmost bit to be sign!
  - 0 → + , 1 → –
  - Rest of bits can be numerical value of number
- Representation called ***sign and magnitude***

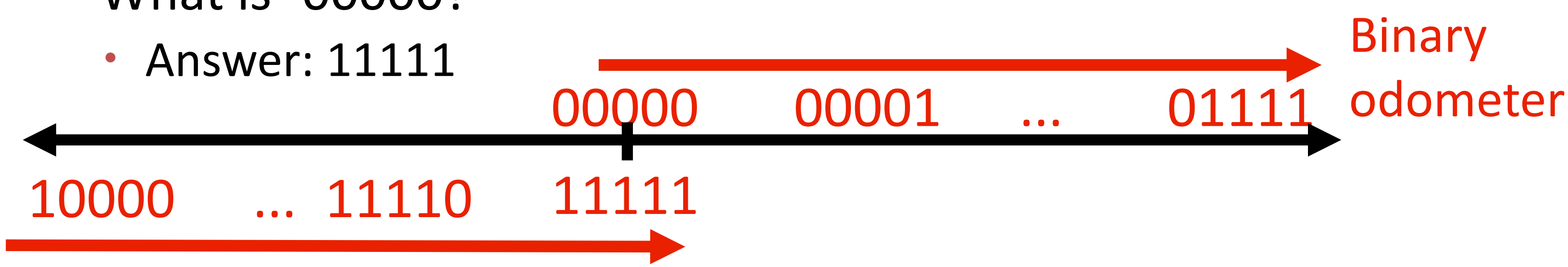


Shortcomings of Sign Magnitude?

- Arithmetic circuit complicated
  - Special steps depending whether signs are the same or not
- Also, two zeros
  - $0x00000000 = +0_{ten}$
  - $0x80000000 = -0_{ten}$
  - What would two 0s mean for programming?
- Also, incrementing "binary odometer", sometimes increases values, and sometimes decreases!
- Therefore sign and magnitude abandoned

Another try

- Complement the bits
  - Example: $7_{10} = 00111_2 \quad -7_{10} = 11000_2$
  - Called ***One's Complement***
  - Note: positive numbers have leading 0s, negative numbers have leadings 1s.
  - What is -00000?
    - Answer: 11111



  - How many positive numbers in N bits? 2^{N-1}
  - How many negative numbers? 2^{N-1}

Shortcomings of One's complement?

- Arithmetic is less complicate than sign & magnitude.
- Still two zeros
  - $0x00000000 = +0_{ten}$
  - $0xFFFFFFFF = -0_{ten}$
- Although used for a while on some computer products, one's complement was eventually abandoned because another solution was better.

Standard Negative # Representation

- Problem is the negative mappings "overlap" with the positive ones (the two 0s). Want to shift the negative mappings left by one.
  - Solution! For negative numbers, complement, then add 1 to the result
- As with sign and magnitude, & one's complement, leading 0s → positive, leading 1s → negative
  - 000000...xxx is ≥ 0, 111111...xxx is < 0
  - except 1...1111 is -1, not -0
- This representation is ***Two's Complement***
- This makes the hardware simple!

In C: short, int, long long, intN_t (C99) are all signed integers.

Two's Complement Formula

- Can represent positive and negative numbers in terms of the bit value times a power of 2:

  $d_{31} \times -(2^{31}) + d_{30} \times 2^{30} + \ldots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0$

- Example: 1101_{two}

  $= 1x\text{-}(2^3) + 1x2^2 + 0x2^1 + 1x2^0$

  $= -2^3 + 2^2 + 0 + 2^0$

  $= -8 + 4 + 0 + 1$

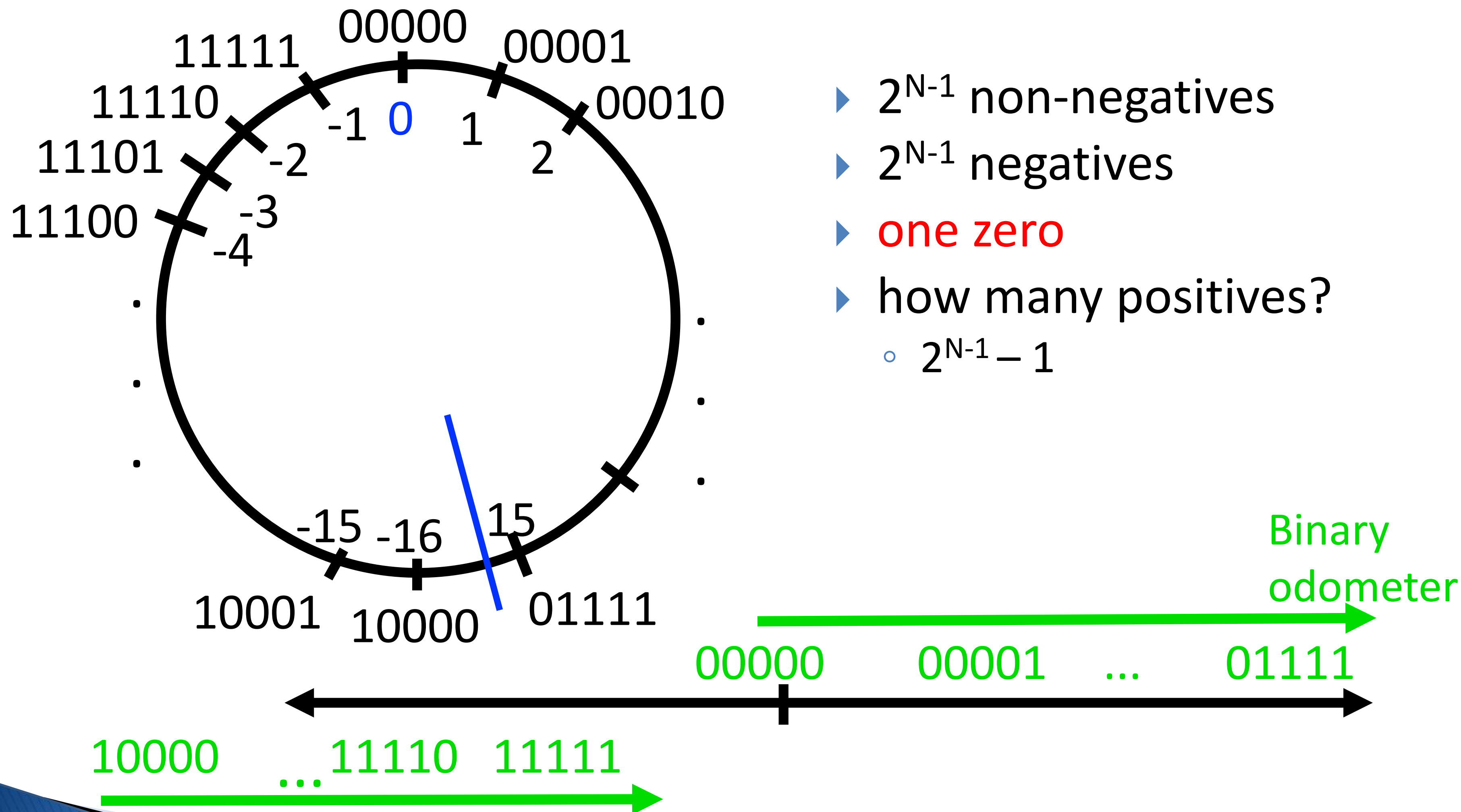
  $= -8 + 5$

  $= -3_{ten}$

Example: -3 to +3 to -3:

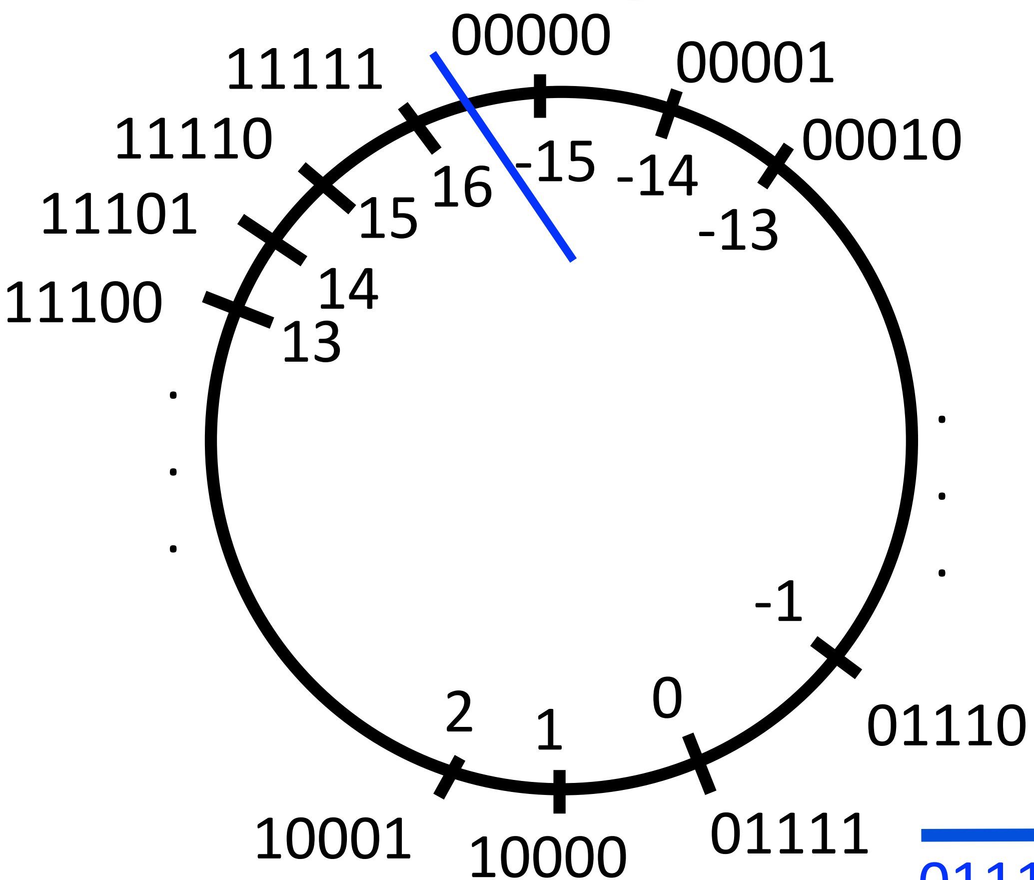
x :	1101_{two}	(-3)
x':	0010_{two}	
+1:	0011_{two}	(3)
()':	1100_{two}	
+1:	1101_{two}	(-3)

2's Complement Number "line": N = 5

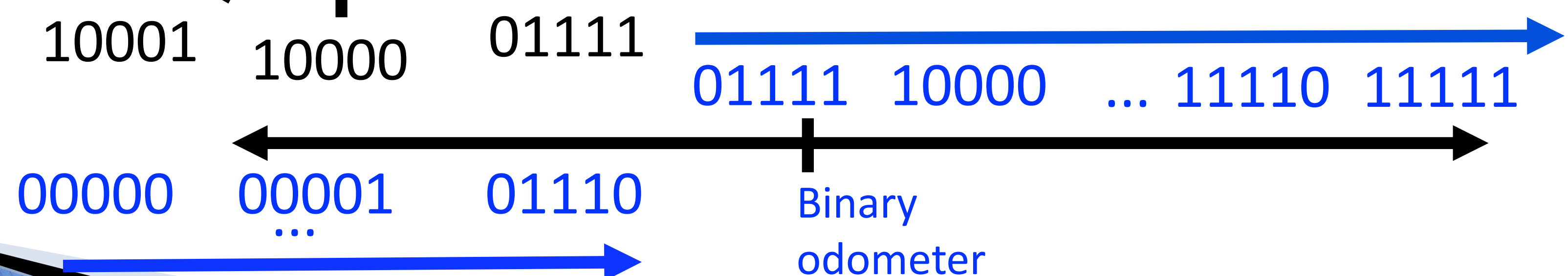


- 2^{N-1} non-negatives
- 2^{N-1} negatives
- one zero
- how many positives?
  - $2^{N-1} - 1$

Bias Encoding: N = 5 (bias = -15)



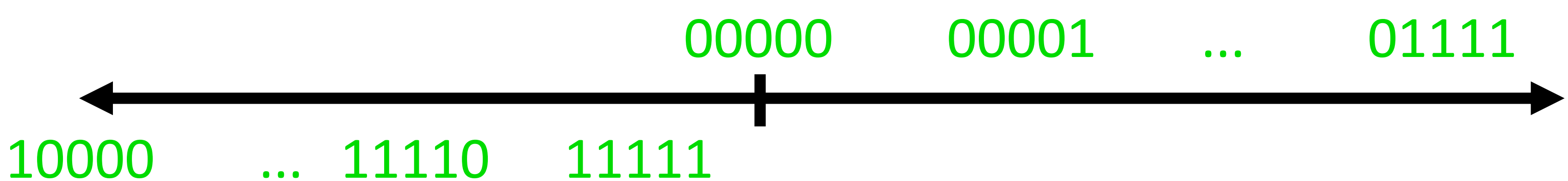
- Want 00… to represent the smallest number
- value = unsigned - bias
- Bias for N bits = $2^{N-1} - 1$
- one zero
- how many positives?
  - 2^{N-1}
  - (more than 2’s complement)



Summary

- We represent "things" in computers as particular bit patterns:
  - N bits → 2^N things
- Different integer encodings have different benefits; 1s complement and sign/mag have most problems.
- unsigned (C99's `uintN_t`) :

- 2's complement (C99's `intN_t`): universal, learn it!



- Overflow: numbers ∞ ; computers finite → errors!

Announcement

- Lab #1 starts next week (8/27)
- Reading assignment
  - Chapter 1.1 – 1.3 of zyBook
  - Chapter 1-3 of K&R (Review of C/C++ programming)