Data Representation

CIS 2400 Recitation 0

What to expect in recitations

- General structure: Brief recap of topics covered in lecture + supplemental exercises for each topic
  - Bonus material will sometimes be covered, but you are only expected to know what is covered in lectures for homeworks and exams!
- Two a week (Tuesday 6:30-8:00pm, Wednesday 12:00-1:00pm)
- These are *optional* and at least one session will be recorded
- Concurrent OH (Tuesday 5:00-8:00pm in Moore 100A)
  - At 6:30 will switch to common area outside the room and use OHQ to handle questions one at a time

Recitation Outline

- Overview
  - Meaning of bits
  - Hex
  - Exercise
- Two's Complement
  - Alternate Explanation
  - Shifting
  - Exercise
- Floating Point
  - Alternative Explanation
  - Conversion
  - Limited Space & Underflow
  - Exercise
- Bonus: Some History of Computing

Overview of Data Representation

The Meaning of Bits

- *A sequence of bits can have many meanings!*
- Consider the hex sequence 0x4E6F21
  - Common interpretations include:
  - The decimal number 5140257
  - The characters “No!”
  - The background color of this slide
  - The real number 7.203034×10^{-39}
- A series of bits can also be code!
- It is up to the program/programmer to decide how to ***interpret*** the sequence of bits

Hex

- Base 16 representation of numbers
- Allows us to represent binary with fewer characters
  - 0b11110011 == 0xF3
    ^ **b**inary ^ he**x**
- One hex = a nibble (4 bits)
  - 0x1 = 0001

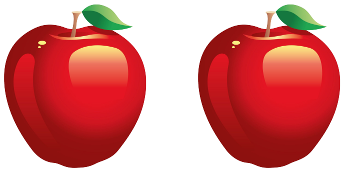
Decimal	Binary	Hex
0	0000	0x0
1	0001	0x1
2	0010	0x2
3	0011	0x3
4	0100	0x4
5	0101	0x5
6	0110	0x6
7	0111	0x7
8	1000	0x8
9	1001	0x9
10	1010	0xA
11	1011	0xB
12	1100	0xC
13	1101	0xD
14	1110	0xE
15	1111	0xF

Leading Zeros

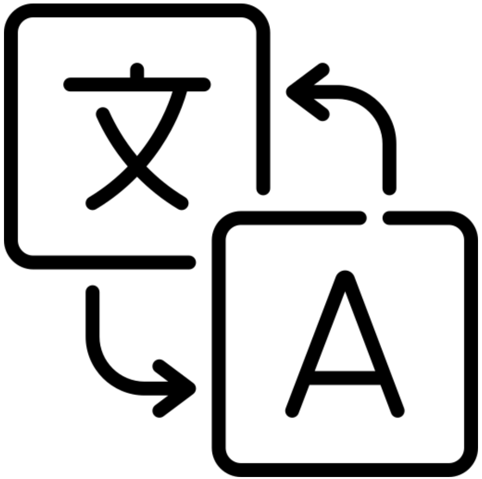
- Leading zeros indicate size, not value
  - 0x0001 = 0000 0000 0000 0001 (16-bit)
  - 0x01 = 0000 0001 (byte, 8-bit)
  - 0x1 = 0001 (nibble, 4-bit)

- Different representations for the same value: ONE

Multiple Representations, One True Value



- 0x2 (hex) = 0010 (binary) = 2 (decimal)
- Yes = Ja = Oui = نعم = 是的
- Using hex does not magically add extra apples!
- Changing languages does not change the true meaning of the word (unless you’re a picky linguist)
- Representation = scope of looking at things



Exercises

Convert the following from binary to decimal

1. 1111 15
2. 10011 19

Do the following conversions between binary and hex (and decimal if you want)

3. $1101\ 0101_b$ D5
4. 0E 0000 1110
5. AB 1010 1011

Exercise: Hex and Binary

Convert the following hex numbers to a string if we interpret each 2 digit hex number as an ASCII character:

47 69 67 61 63 68 61 64 21

Gigachad!

Two’s Complement

Representing negative numbers

Representing negatives can be a challenge! It's not as intuitive as positive numbers.

Two alternative methods → Why aren't these representations ideal?

Representation 1: Sign-magnitude

- Intuition: represent +/- in the MSB, setting - as 1 and + as 0
- e.g. 2 = **0**010, -2 = **1**010

Issues:

- -0?
- Math?

Representation 2: One's complement

- Intuition: to go from positive to negative and vise versa, flip the bits
- e.g. 2 = 0010, -2 = 1101

Issues:

- -0?

2C Recap

A negative number is the same as a positive number, except the MSB is now negative

e.g.

1 0 0 1

-2^3 2^2 2^1 2^0

-8 + 0 + 0 + 1 = -7

Question: What is the bit representation of the minimum number for any n-bit 2C number? How about the maximum number?

Minimum: MSB is 1, rest are 0 → 1000…

Maximum: MSB is 0, rest are 0 → 0111…

2C Recap

Alternate explanation: to store X, we compute 2^n + X and knock off MSB (bit n + 1)

e.g. X = -7 in n = 4 bits

Compute 2^n = 10000

Add X (i.e. subtract 7)

~~1~~0000

-0111

~~1~~1001

e.g. X = +7 in n = 4 bits

Compute 2^n = 10000

Add X (i.e. add 7)

10000

+0111

~~1~~0111

Exercises

Convert the following from 2C to decimal or vise-versa:

1. 1111 -1
2. 0110 6
3. 11010101 -43
4. -2 1110
5. -7 1001

Flip the signs of the following 2C numbers:

1. 1011 0101
2. 0110 1010
3. 0000 0000

Shifting and 2C

- Left shifting: Always pad with 0's → **what is the effect of this for decimal numbers?**
  - 1111 becomes 1110
  - 0010 becomes 0100

Multiplying by two! (usually)

- Right shifting: Logical Shift
  - Always pad with 0's
  - 1111 becomes 0111
  - 0010 becomes 0001
- Arithmetic Shift → **why is this called arithmetic shift?**
  - Pad with the MSB
  - 1111 becomes 1111
  - 0010 becomes 0001

It retains the sign!

Floating Point

Floating Point

More numbers exist than just whole numbers so what do we do if we have a decimal? How can we represent this through binary?

Ideas?

Fixed Point

We can stick a decimal point in the middle and then write the whole number on the left and the decimal on the right.

2.5 →

1.25 →

40.90 →

What's the problem with fixed point?

IEE

Breaks a number down to scientific notation and then represents it.

-1.523 x 10^6

Order of numbers : Sign , Ones place , Mantissa , Exponent

We only need the Sign, Mantissa, and Exponent to represent the number using IEE format!

How does IEE Format Work?

1) We'll be doing 32 bit format IEE. 1 bit is given to the sign , 8 bits go to the exponent and the other 23 go to the Mantissa

2) Sign is 1 if negative 0 if positive

3) The exponent should be represented as an unsigned 8 bit integer (so we can only represent a number that falls between which two numbers?)

3) The remaining bits go to the Mantissa

Exercise

Convert .75 to IEE format

1) Mark the sign bit then ignore it
2) Convert number to fixed point binary
   a) Similar strategies to decimal -> unsigned int ▪
3) Multiply by ‘1’
4) Shift the point by changing the exponent
   a) Shift bits to the left: decrement exponent
   b) Shifting bits to the right: increment exponent
5) Add bias to the exponent then store •
   a) “bias” for floats is 127
6) Store mantissa
   a) Truncate extra bits, or pad out with 0’s if not enough

Do it yourself!

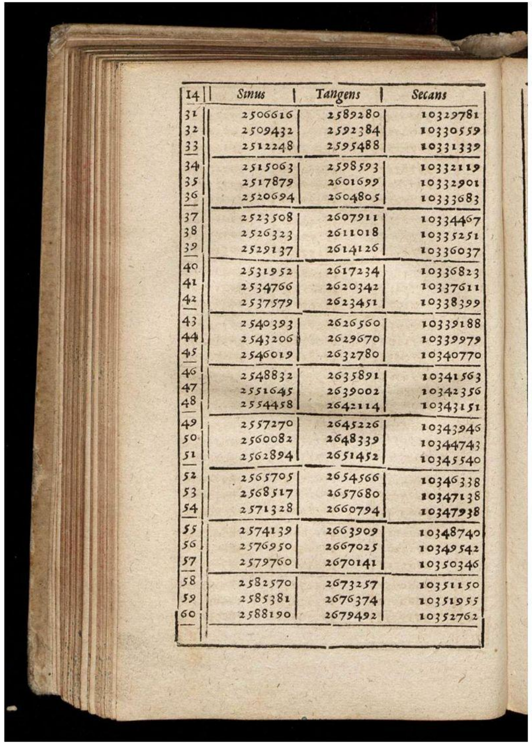
Convert 85.125 to IEE format

1) Mark the sign bit then ignore it
2) Convert number to fixed point binary
   a) Similar strategies to decimal -> unsigned int ▪
3) Multiply by ‘1’
4) Shift the point by changing the exponent
   a) Shift bits to the left: decrement exponent
   b) Shifting bits to the right: increment exponent
5) Add bias to the exponent then store
   a) “bias” for floats is 127
6) Store mantissa
   a) Truncate extra bits, or pad out with 0’s if not enough

Bonus Material!
Some History of Computing

Mathematical Tables

- Before non-human calculators, you'd look up the result of a computation in a table
- These were calculated by "Human Computers"
    - Some tables (such as logarithm tables) were prone to errors. Calculations were complicated…



14	Sinus	Tangens	Secans
31	2506616	2589280	10329781
32	2509432	2592384	10330559
33	2512248	2595488	10331339
34	2515063	2598593	10332119
35	2517879	2601699	10332901
36	2520694	2604805	10333683
37	2523508	2607911	10334467
38	2526323	2611018	10335251
39	2529137	2614126	10336037
40	2531952	2617234	10336823
41	2534766	2620342	10337611
42	2537579	2623451	10338399
43	2540393	2626560	10339188
44	2543206	2629670	10339979
45	2546019	2632780	10340770
46	2548832	2635891	10341563
47	2551645	2639002	10342356
48	2554458	2642114	10343151
49	2557270	2645226	10343946
50	2560082	2648339	10344743
51	2562894	2651452	10345540
52	2565705	2654566	10346338
53	2568517	2657680	10347138
54	2571328	2660794	10347938
55	2574139	2663909	10348740
56	2576950	2667025	10349542
57	2579760	2670141	10350346
58	2582570	2673257	10351150
59	2585381	2676374	10351955
60	2588190	2679492	10352762

Difference Engine

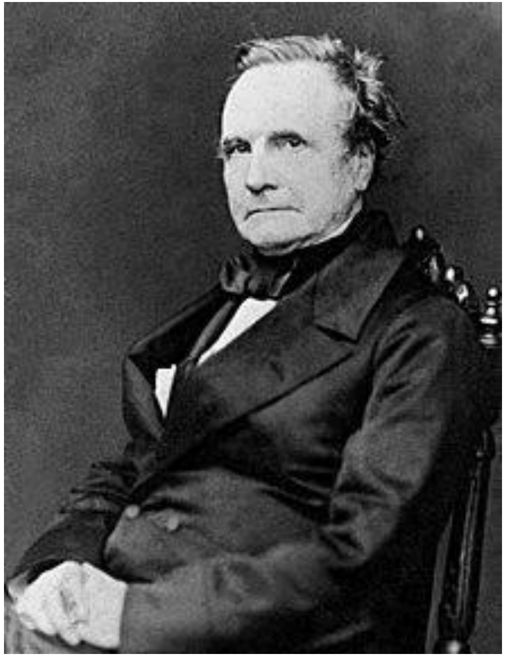
- Instead of calculating mathematical tables by hand, calculate them with an automatic mechanical calculator!
  - The Difference Engine!
- This was the inspiration for Charles Babbage
- Funding was received by British government. Human Computers were time consuming and expensive
  - This sounds a lot like the argument for automation today

First “Computer”?

- Was Charles Babbage the first person to design mechanical “computers”?
- Antikythera mechanism 205-87 BC
- Adding machines
- Johann Helfrich von Müller
- First to come up with the idea of calculating mathematical tables by machine. (1786)
    - Difference Engine
- Plan was never funded

Analytical Engine

- The first proposed general-purpose computer
  - First described in 1837
  - First Turing Complete computer!
  - Never built ☹️
- Babbage turned interest away from the Difference Engine
- British government stopped funding
  - Was more interested in seeing the output of the engine, not the development of such machines

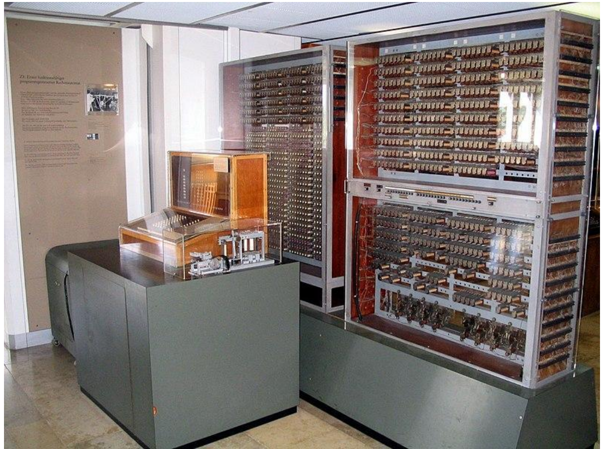


Ada Lovelace

- Worked with Charles Babbage on the analytical engine
- Described an algorithm to computer Bernoulli numbers
- “First” programmer is disputed.
  - Babbage already had some “programs”
  - She is the first to *publish* a program
- Was first to see the potential of computers as something that would act on more than numbers

The Analytical Engine Influence

- Had basic architecture very similar to modern machines
  - Instruction based
  - Had an I/O unit
  - Turing complete
- Would be unknown to the builders of computers in the 1930’s and 1940’s
- First built general purpose computer
  the Z3 in 1941
  - Over 100 years since the analytical engine was first described



This is just the very beginning

There's a lot more that went into the development of computers from the Z3 to your laptops

- Mark I
- ENIAC
- etc.

Recommended reading if you're interested in computing history:

- *The Innovators* by Walter Isaacson
- *The Perfectionists: How Precision Engineers Created the Modern World* by Simon Winchester

That’s all we have for today!

Reminders:

- TA-lead recitations will take place on
  - Tuesdays 6:30-8:00pm in Moore 100A
  - Wednesday 12:00-1:30pm in Moore 100C
- Check the course website for OH times
- Binary Representation quiz is due **this Friday** at 11:59pm