Computer Systems

Lecture 2

Binary and Two's Complement Numbers

Dr José Cano, Dr Lito Michala
School of Computing Science
University of Glasgow
Spring 2020

Outline

- Number representation
- Binary numbers
  - Converting binary to decimal
  - Converting decimal to binary
  - Binary addition
- Two's complement
  - Sign bit
  - Negating a number
  - Converting to decimal
- Hexadecimal

Quiz

- There will be a quiz each week
- You can do it any time starting after the Thursday lecture, and must finish by Friday in the following week
- The quiz is on Moodle
- 10% of the total assessment comes from the quiz average
- You are encouraged to refer to the course documents as you do the quiz
- Read the course documents!
  - Don't ignore them and use random Google searches instead!

Number representation

- There are several types of numbers: each has its own representation using bits
  - Integers
    - Non-negative integers use binary

      23, 0, 459 (must be >= 0)
    - Signed integers use two's complement

      48, -239 (can be negative)
  - Reals
    - Approximate real numbers use floating point

      3.14, 2.5e9, -351.02638134
- Computer hardware and programming languages support other representations
  - Examples: BCD numbers, fixed point fractional numbers, saturated numbers

Outline

- Number representation
- Binary numbers
  - Converting binary to decimal
  - Converting decimal to binary
  - Binary addition
- Two's complement
  - Sign bit
  - Negating a number
  - Converting to decimal
- Hexadecimal

Binary numbers

- Binary representation uses a word of k bits to represent a non-negative integer between 0 and $2^k - 1$
- Binary numbers cannot be negative
  - There are other ways to represent negative numbers (two's complement is most widely used)
- People often use terminology loosely, and say “binary” when they mean “word of bits”
  - Example: Integer variables are represented using words of bits, but they are never represented using binary
  - Binary numbers are always non-negative!
- Binary representation is similar to decimal, but it uses base 2 instead of base 10

Decimal number representation

$2053_{10} = 2 \times 10^{3} + 0 \times 10^{2} + 5 \times 10^{1} + 3 \times 10^{0}$

$= 2000 \quad + \quad 0 \quad + \quad 50 \quad + \quad 3$

$= 2053_{10}$

- Column values are powers of 10

$10^{0} = 1$ weight of rightmost digit

$10^{1} = 10$

$10^{2} = 100$

$10^{3} = 1000$ weight of leftmost digit

Binary number representation

$$1001_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 8 + 0 + 0 + 1$$

$$= 910$$

- This is how to convert a binary number to decimal!

- Column values are powers of 2

$2^0 = 1$	weight of rightmost bit
$2^1 = 2$	
$2^2 = 4$	
$2^3 = 8$	weight of leftmost bit

The powers of 2

- It's useful to know the value of each bit position!

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

- **Tip**: Don't memorise the table! Construct it whenever you need it
  - Just write down 1, and keep adding another value to the left by doubling the previous value

- **Exercise**: Convert binary number 10011 to decimal

Converting binary to decimal

- **Exercise**: Convert binary number 10011 to decimal

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
1	0	0	1	1

16 + 2 + 1 = 19

Converting decimal to binary

- When we convert a decimal number *x* to binary, we need to
  - Know the word size k of the result
  - Check that x will fit in the word: $0 \leq x \leq 2^k - 1$

- **Example**: convert decimal number 203 to an 8-bit binary number

- Check: $0 \leq 203 \leq 255$

- This holds, so we can indeed represent 203 in an 8-bit word

(1) Calculate the 128 column, remainder is 203

$203 \geq 128$ so enter 1

128	64	32	16	8	4	2	1
1							

The new remainder is 203 - 128 = 75

(2) Calculate the 64 column, remainder is 75

$75 \geq 64$ so enter 1

128	64	32	16	8	4	2	1
1	1						

The new remainder is 75 - 64 = 11

(3) Calculate the 32 column, remainder is 11

$11 \geq 32$ is **false** so enter 0

128	64	32	16	8	4	2	1
1	1	0					

The new remainder is still 11

(4) Calculate the 16 column, remainder is 11

$11 \geq 16$ is **false** so enter 0

128	64	32	16	8	4	2	1
1	1	0	0				

The new remainder is still 11

(5) Calculate the 8 column, remainder is 11

$11 \geq 8$ so enter 1

128	64	32	16	8	4	2	1
1	1	0	0	1			

The new remainder 11 - 8 = 3

(6) Calculate the 4 column, remainder is 3

$3 \geq 4$ is **false** so enter 0

128	64	32	16	8	4	2	1
1	1	0	0	1	0		

The new remainder still 3

(7) Calculate the 2 column, remainder is 3

$3 \geq 2$ so enter 1

128	64	32	16	8	4	2	1
1	1	0	0	1	0	1	

The new remainder is 3 - 2 = 1

(8) Calculate the 1 column, remainder is 1

$1 \geq 1$ so enter 1

128	64	32	16	8	4	2	1
1	1	0	0	1	0	1	1

The new remainder is $1 - 1 = 0$ and we're **finished**

Check the result: convert it back to decimal

$$
\begin{aligned}
11001011_2 &= 2^7 + 2^6 + 2^3 + 2^1 + 2^0 \\
&= 128 + 64 + 8 + 2 + 1 \\
&= 203
\end{aligned}
$$

- It's easier to convert binary to decimal, so it's worth checking!
- Also, note that when you convert decimal to binary, the remainder in the 1 column must be 0
  - If not, you've made a mistake

Binary addition

- You can add two binary numbers *x* and *y* the same way as adding decimal numbers
- Write one number above the other
- Work through each column, from right to left
- In each column, add the bit from x, the bit from y, and the carry from the column to the right
- This gives the sum bit *s* for the column, and the carry output which goes to the left

Adding bits

- Calculate *x* + *y* + *z* giving 2-bit result c, s (c is carry, s is sum)

Addition table

x	y	z	c	s	Result
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

- The sum is 1 if an odd number of inputs are 1
- The carry is 1 if two or more inputs are 1
- You can view *c* and *s* as a 2-bit binary number giving the result

Example

- Add two 8-bit binary numbers: $x + y$

  $x = 0010\ 1101 = 32 + 8 + 4 + 1 = 45$

  $y = 0100\ 1110 = 64 + 8 + 4 + 2 = 78$

- The correct answer is $x + y = 45 + 78 = 123$

Setting up the problem

		128	64	32	16	8	4	2	1
c									0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s									

(1) Add the weight 1 column

		128	64	32	16	8	4	2	1
c								0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s									1

(2) Add the weight 2 column

		128	64	32	16	8	4	2	1
c							0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s								1	1

(3) Add the weight 4 column

		128	64	32	16	8	4	2	1
c						1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s							0	1	1

(4) Add the weight 8 column

		128	64	32	16	8	4	2	1
c					1	1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s						1	0	1	1

(5) Add the weight 16 column

		128	64	32	16	8	4	2	1
c				0	1	1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s					1	1	0	1	1

(6) Add the weight 32 column

		128	64	32	16	8	4	2	1
c			0	0	1	1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s				1	1	1	0	1	1

(7) Add the weight 64 column

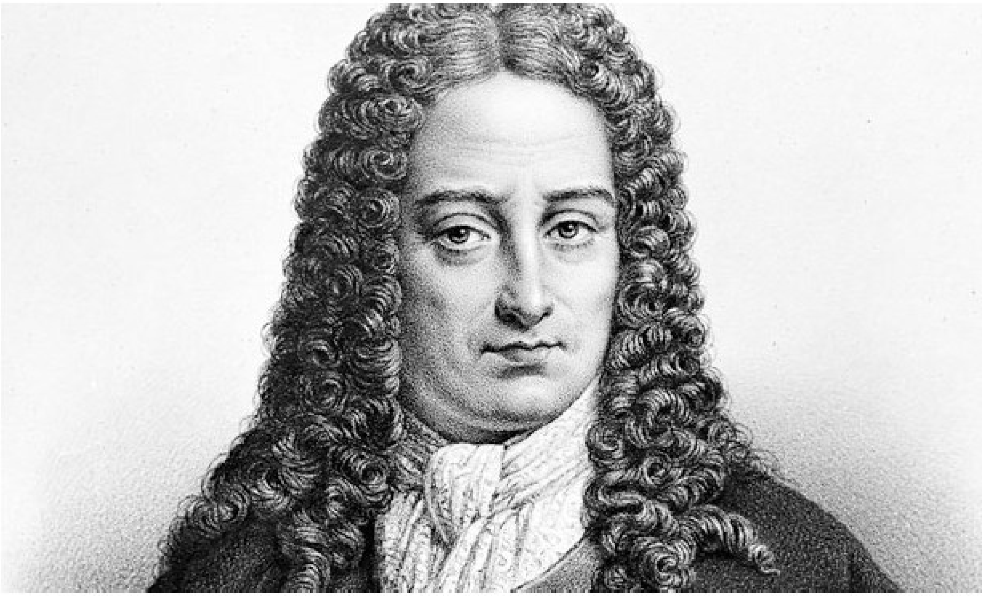
		128	64	32	16	8	4	2	1
c		0	0	0	1	1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s			1	1	1	1	0	1	1

(8) Add the weight 128 column

		128	64	32	16	8	4	2	1
c	0	0	0	0	1	1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s		0	1	1	1	1	0	1	1

- The result is 0111 1011 = 64 + 32 + 16 + 8 + 2 + 1 = 123 which is the right answer!

The discoverer of binary numbers



- Gottfried Wilhelm Leibniz (1646 - 1716)
- German mathematician and philosopher
- Invented the calculus and the binary number system (about 1680)
  - Isaac Newton also invented calculus independently around the same time

Outline

- Number representation
- Binary numbers
  - Converting binary to decimal
  - Converting decimal to binary
  - Binary addition
- Two's complement
  - Sign bit
  - Negating a number
  - Converting to decimal
- Hexadecimal

Two's complement

- Binary cannot represent negative numbers!

- Two's complement is a method for representing integers that can be negative or positive

- Remember that a k-bit word can represent 2^k different values
  - In binary, all those values represent nonnegative numbers from 0 to $2^k - 1$
  - In two's complement, half of those values represent negative integers, and half represent nonnegative integers
  - The range is -2^{k-1} to $2^{k-1} - 1$

Sign bit

- The sign bit is the lefmost bit of a two's complement number
  - If the sign bit is 1, the number is negative (< 0)
  - If the sign bit is 0, the number is nonnegative ($\geq$ 0)
  - If all the bits are 0, the number is 0

- Examples

0101 1100	is positive (92 > 0)
1001 1001	is negative (-103 < 0)
0000 0000	is the integer 0

How to interpret a two's complement number

- There are many ways to convert a two's complement word to/from decimal
- Our approach is based on how computers actually work, and is the easiest for humans to use
  - We have an algorithm to negate any two's complement number
  - If a two's complement number is nonnegative, it acts just like a binary number
  - If it is negative, just negate it and then use binary conversion to get the decimal number

Negating a two's complement number x

- Two steps
    1. Invert each bit (replace 0 by 1, replace 1 by 0)
    2. Add 1

- The result is the representation of -x

Example: -36 in two's complement

x	0010 0100	36_{10}
invert	1101 1011	
add 1	1101 1100	-36_{10}

Decoding a two's complement number

- If the sign bit is 0, then treat it just like a binary number

- If the sign bit is 1, then negate it and treat the result like a binary number

0010 0110 (is nonnegative)
$= 32 + 4 + 2$
$= 38$

1011 1000 (is negative)
0100 0111 (invert)
0100 1000 (add 1)
$= 64 + 8 = 72$
so 1011 1000 = -72

Outline

- Number representation
- Binary numbers
  - Converting binary to decimal
  - Converting decimal to binary
  - Binary addition
- Two's complement
  - Sign bit
  - Negating a number
  - Converting to decimal
- Hexadecimal

Hexadecimal: easier notation for writing words

- When working with machine language and assembly language, we frequently need to write down the values of words

- This is normally done using hexadecimal notation

- It's base 16 (binary is base 2, and decimal is base 10)

- Why use base 16?

- You can break a long word of bits into groups of 4 bits, and replace each group by the corresponding hex digit

Table of 4-bit numbers

word	hex value	bin value	tc value
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	8	-8
1001	9	9	-7
1010	a	10	-6
1011	b	11	-5
1100	c	12	-4
1101	d	13	-3
1110	e	14	-2
1111	f	15	-1

Why use use hex?

- Here's a 16 bit word: 0011110000101111

- We'll usually have about 20 of these to look at
  - Machine language programming on Sigma16

- And if you look at current commercial computers, the words are 64 bits (you would need to work with a couple dozen of these at a time)
  - 1000111010100111010011110000101111110110001010001010010101010001

- Hex representation of 0011 1100 0010 1111 is 3c2f

- It's easier with hex!

Arithmetic with hex

- It's easy to add hex numbers

- It is extremely rare to multiply or divide them
  – You will probably never need to do this

- To add two hex numbers, write them one above the other, and add by columns

- Just remember what each hex digit means
  – c + 2 means 12 + 2, which is 14, and that's hex digit e

- If the sum in a column is greater than 16, you add a carry of 1 to the column to the left
  – 004a + 0009 = 0053

A couple of tips

- We will write hex numbers with a dollar sign in front
  - 23 is decimal: 2 x 10 + 3, pronounced “twenty three”
  - $0023 is hex: 2 x 16 + 3 = 35, pronounced “zero zero two three”
  - Professionals pronounce hex numbers by saying every digit, including leading zeros, and never use teens, twenty, hundreds, etc for hex

Recap: A word has many meanings

- There are many ways to interpret the meaning of a word of bits
  - Binary, two's complement, floating point, character, and many more
- A word of bits has no inherent meaning!
- It has one meaning if interpreted as binary, another if interpreted as two's complement, and so on
- It is meaningless to ask "what does 1010 represent?"
- We can ask
  - "what does 1010 represent as a binary number?" (10)
  - "what does 1010 represent as a two's complement number?" (-6)

To do

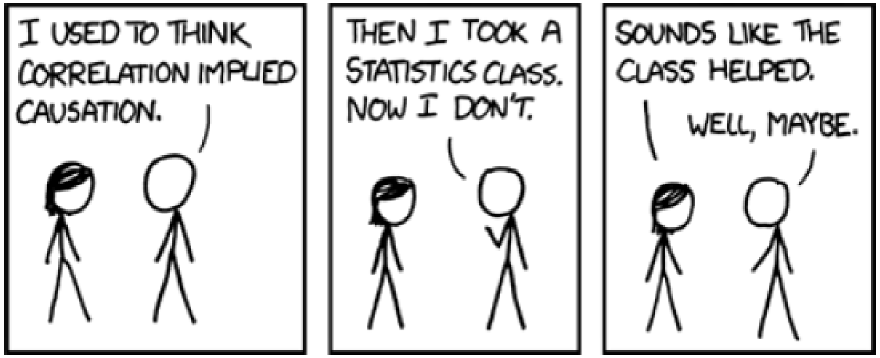
- Review the slides and work through the examples

- **Quiz 1 on Moodle**: this is assessed
  - Deadline: Friday next week (January 24)

- No lab this week: the first lab is next week

- Check Moodle for schedule, documents, announcements

- Over the weekend, the lab sheet for next week will be posted on Moodle
  - It contains problems about the first two lectures
  - Solve the problems
  - Discuss these at your lab next week



https://xkcd.com/552/