Binary Numbers

X. Zhang
Fordham Univ.

Numeral System

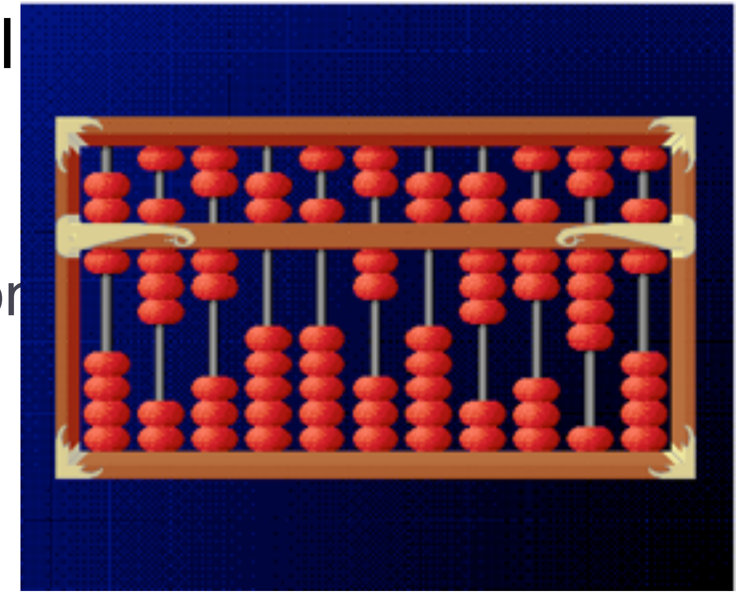
- A way for expressing numbers, using symbols in a consistent manner.
  - "11" can be interpreted differently:
    - in the binary symbol: *three*
    - in the decimal symbol: *eleven*
  - *"LXXX" represents 80 in Roman numeral system*
- For every number, there is a unique representation (or at least a standard one) in the numeral system

Modern numeral system

- Positional base 10 numeral systems
  - Mostly originated from India (Hindu-Arabic numeral system or Arabic numerals)
- Positional number system (or place value system)
  - use same symbol for different orders of magnitude
- For example, “1262” in base 10
  - the “2” in the rightmost is in “one’s place” representing “2 ones”
  - The “2” in the third position from right is in “hundred’s place”, representing “2 hundreds”
  - “one thousand 2 hundred and sixty two”
  - $1*10^3+2*10^2+6*10^1+2*10^0$

Modern numeral system (2)

- In base 10 numeral system
  - there is 10 symbols: 0, 1, 2, 3, …, 9
- Arithmetic operations for positional system is simple
  - Algorithm for multi-digit addition, subtraction, multiplication and division
  - This is a Chinese Abacus (there are many other types of Abacus in other civilizations) dated back to 200 BC



Other Positional Numeral System

- Base: number of digits (symbols) used in the system.
  - Base 2 (i.e., binary): only use 0 and 1
  - Base 8 (octal): only use 0,1,…7
  - Base 16 (hexadecimal): use 0,1,…9, A,B,C,D,E,F
- Like in decimal system,
  - Rightmost digit: represents its value times the base to the zeroth power
  - The next digit to the left: times the base to the first power
  - The next digit to the left: times the base to the second power
  - …
  - For example: binary number 10101

$= 1*2^4+0*2^3+1*2^2+0*2^1+1*2^0=16+4+1=21$

Why binary number?

- Computer uses binary numeral system, i.e., base 2 positional number system
  - Each unit of memory media (hard disk, tape, CD …) has two states to represent 0 and 1
  - Such physical (electronic) device is easier to make, less prone to error
    - E.g., a voltage value between 0-3mv is 0, a value between 3-6 is 1 …

Binary => Decimal

- Interpret binary numbers (transform to base 10)
  - 1101

  $= 1*2^3+1*2^2+0*2^1+1*2^0=8+4+0+1=13$
- Translate the following binary number to decimal number
  - 101011

Generally you can consider other bases

- Base 8 (Octal number)
  - Use symbols: 0, 1, 2, …7
  - Convert octal number 725 to base 10:

    $=7*8^2+2*8^1+5=\ldots$
  - Now you try:

  $(1752)_8 =$
- Base 16 (Hexadecimal)
  - Use symbols: 0, 1, 2, …9, A, B, C,D,E, F
  - $(10A)_{16} = 1*16^2+10*16^0=..$

Binary number arithmetic

- Analogous to decimal number arithmetics
- How would you perform addition?
  - 0+0=0
  - 0+1=1
  - 1+1=10 (a carry-over)
  - Multiple digit addition: 11001+101=
- Subtraction:
  - Basic rule:
  - Borrow one from next left digit

From Base 10 to Base 2: using table

- Input : a decimal number
- Output: the equivalent number in base 2
- Procedure:
  - Write a table as follows
  1. Find the largest two's power that is smaller than the number
     1. Decimal number 234 => largest two's power is 128
  2. Fill in 1 in corresponding digit, subtract 128 from the number => 106
  3. Repeat 1-2, until the number is 0
  4. Fill in empty digits with 0

…	512	256	128	64	32	16	8	4	2	1
			1	1	1	0	1	0	1	0

- Result is 11101010

From Base 10 to Base 2: the recipe

- Input : a decimal number
- Output: the equivalent number in base 2
- Procedure:
  1. Divide the decimal number by 2
  2. Make the remainder the next digit to the left of the answer
  3. Replace the decimal number with the quotient
  4. If quotient is not zero, Repeat 1-4; otherwise, done

Convert 100 to binary number

100 % 2 = 0
=> last digit

100 / 2 = 50

50 % 2 = 0
=> second last digit

50/2 = 25

25 % 2 = 1
=> 3rd last digit

25 / 2 = 12

The result is **1100100**

12 % 2 = 0 =>
4th last digit

12 / 2 = 6

6 % 2 = 0 =>
5th last digit

6 / 2 = 3

3 % 2 = 1
=> 6th last digit

3 / 2 =1

1 % 2 = 1
=> 7th last digit

1 / 2 = 0

Stop as the decimal # becomes 0

Data Representation in Computer

- In modern computers, all information is represented using binary values.
- Each storage location (cell): has two states
  - low-voltage signal => 0
  - High-voltage signal => 1
  - i.e., it can store a binary digit, i.e., bit
- Eight bits grouped together to form a byte
- Several bytes grouped together to form a word
  - Word length of a computer, e.g., 32 bits computer, 64 bits computer

Different types of data

- Numbers
  - Whole number, fractional number, …
- Text
  - ASCII code, unicode
- Audio
- Image and graphics
- video

How can they all be represented as binary strings?

Representing Numbers

- Positive whole numbers
  - We already know one way to represent them: i.e., just use base 2 number system
- All integers, i.e., including negative integers
  - Set aside a bit for storing the sign
    - 1 for +, 0 for –
- Decimal numbers, e.g., 3.1415936, 100.34
  - Floating point representation:
    - sign * mantissa * 2^{exp}
  - 64 bits: one for sign, some for mantissa, some for exp.

Representing Text

- Take English text for example
- Text is a series of characters
  - letters, punctuation marks, digits 0, 1, …9, spaces, return (change a line), space, tab, …
- How many bits do we need to represent a character?
  - 1 bit can be used to represent 2 different things
  - 2 bit … $2*2 = 2^2$ different things
  - n bit 2^n different things
- In order to represent 100 diff. character
  - Solve $2^n = 100$ for n
  - n = $\lceil \log_2 100 \rceil$, here the $\lceil x \rceil$ refers to the ceiling of x, i.e., the smallest integer that is larger than x:

$$\lceil \log_2 100 \rceil = \lceil 6.6438 \rceil = 7$$

There needs a standard way

- ASCII code: **American Standard Code for Information Interchange**
  - ASCII codes represent text in computers, communications equipment, and other devices that use text.
  - 128 characters:
    - 33 are non-printing control characters (now mostly obsolete) [7] that affect how text and space is processed
    - 94 are printable characters
    - space is considered an invisible graphic

ASCII code

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

There needs a standard way

- Unicode
  - international/multilingual text character encoding system, tentatively called Unicode
  - Currently: 21 bits code space
  - How many diff. characters?
- Encoding forms:
  - UTF-8: each Unicode character represented as one to four 8-but bytes
  - UTF-16: one or two 16-bit code units
  - UTF-32: a single 32-but code unit

In Summary