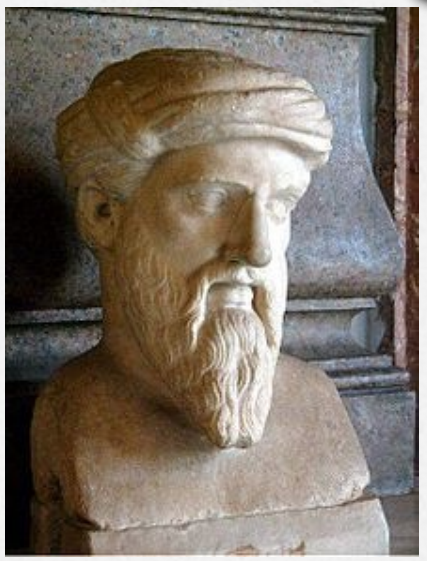
COMP1003 Computer Organization

Lecture 2

Bits: Data Representation and Manipulation

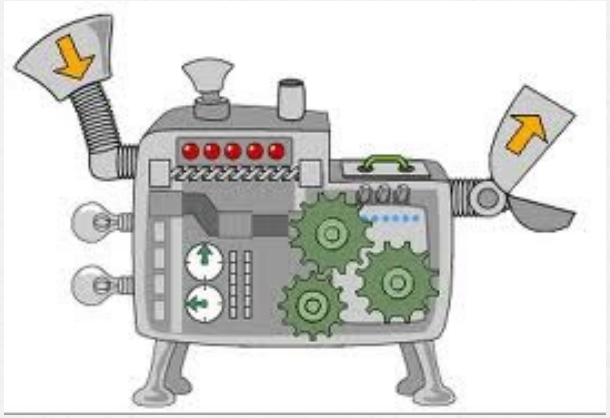
United International College

Numbers



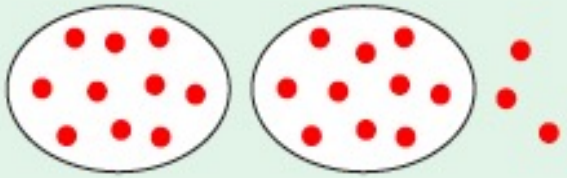
- “All things are numbers.” Pythagoras (毕达哥拉斯 570-490 BC)

- Computers are number crunching machines
  - Input: numbers
    - Manipulations on numbers
  - Output: numbers



Numbers

The number system we use in this course is based on the **Hindu-Arabic system** which uses the digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.



The number of dots shown here is twenty three. We write this as 23, which means '2 tens and 3 ones'.

How was the number 23 written by:

- ancient Egyptians
- Mayans
- ancient Greeks
- Chinese and Japanese?
- Romans

The Egyptian Number System

The Egyptians used a tally system based on the number ten. Ten of one symbol could be replaced by one of another symbol. We call this a **base ten system**.

1	10	100	1000
			
staff	hock	scroll	lotus flower

10 000	100 000	1 000 000	10 000 000
			
bent stick	burbay fish	astonished man	religious symbol

The order in which the symbols were written down did not affect the value of the numerals. The value of the numerals could be found by adding the value of the symbols used.

So,  or  would still represent 35.

The **Egyptian system** did not have **place values**.

Roman Number System

1	2	3	4	5	6	7	8	9	10	
I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**	
20	30	40	50	60	70	80	90	100	500	1000
XX	**XXX**	**XL**	**L**	**LX**	**LXX**	**LXXX**	**XC**	**C**	**D**	**M**

Unlike the Egyptian system, numbers written in the Roman system had to be written in order.

For example:

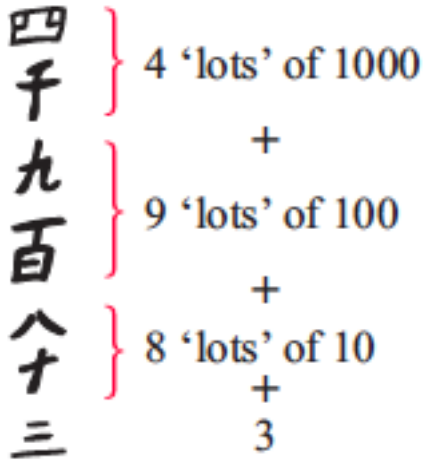
IV stands for 1 before 5	or 4	whereas VI stands for 1 after 5	or 6.
XC stands for 10 before 100	or 90	whereas CX stands for 10 after 100	or 110.

Larger numerals were formed by placing a stroke above the symbol. This made the number 1000 times as large.

5000	10 000	50 000	100 000	500 000	1 000 000
$\overline{\mathbf{V}}$	$\overline{\mathbf{X}}$	$\overline{\mathbf{L}}$	$\overline{\mathbf{C}}$	$\overline{\mathbf{D}}$	$\overline{\mathbf{M}}$

The Chinese System

1	2	3	4	5	6
一	二	三	四	五	六
7	**8**	**9**	**10**	**100**	**1000**
七	八	九	十	百	千

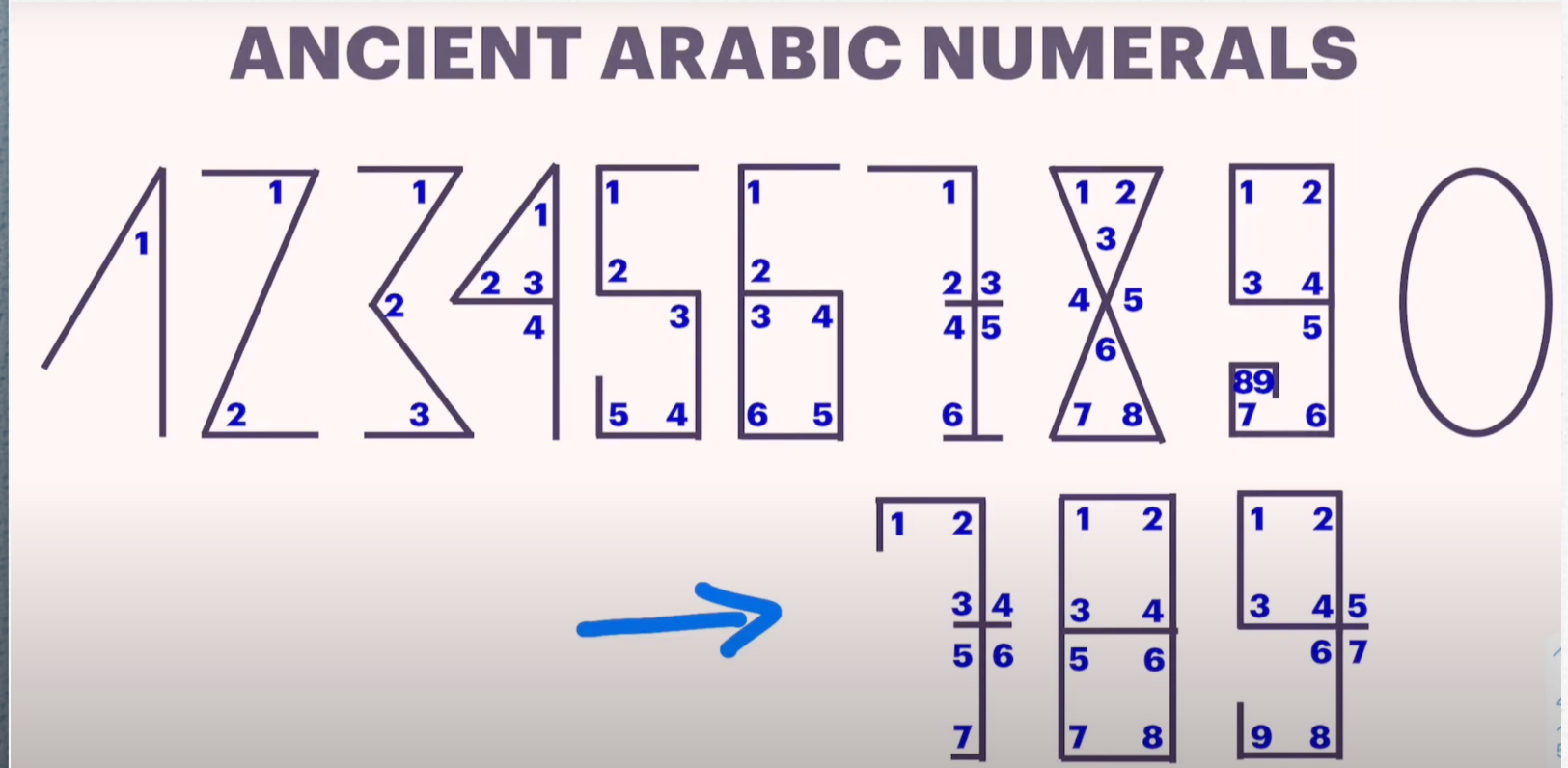


The Hindu-Arabic System

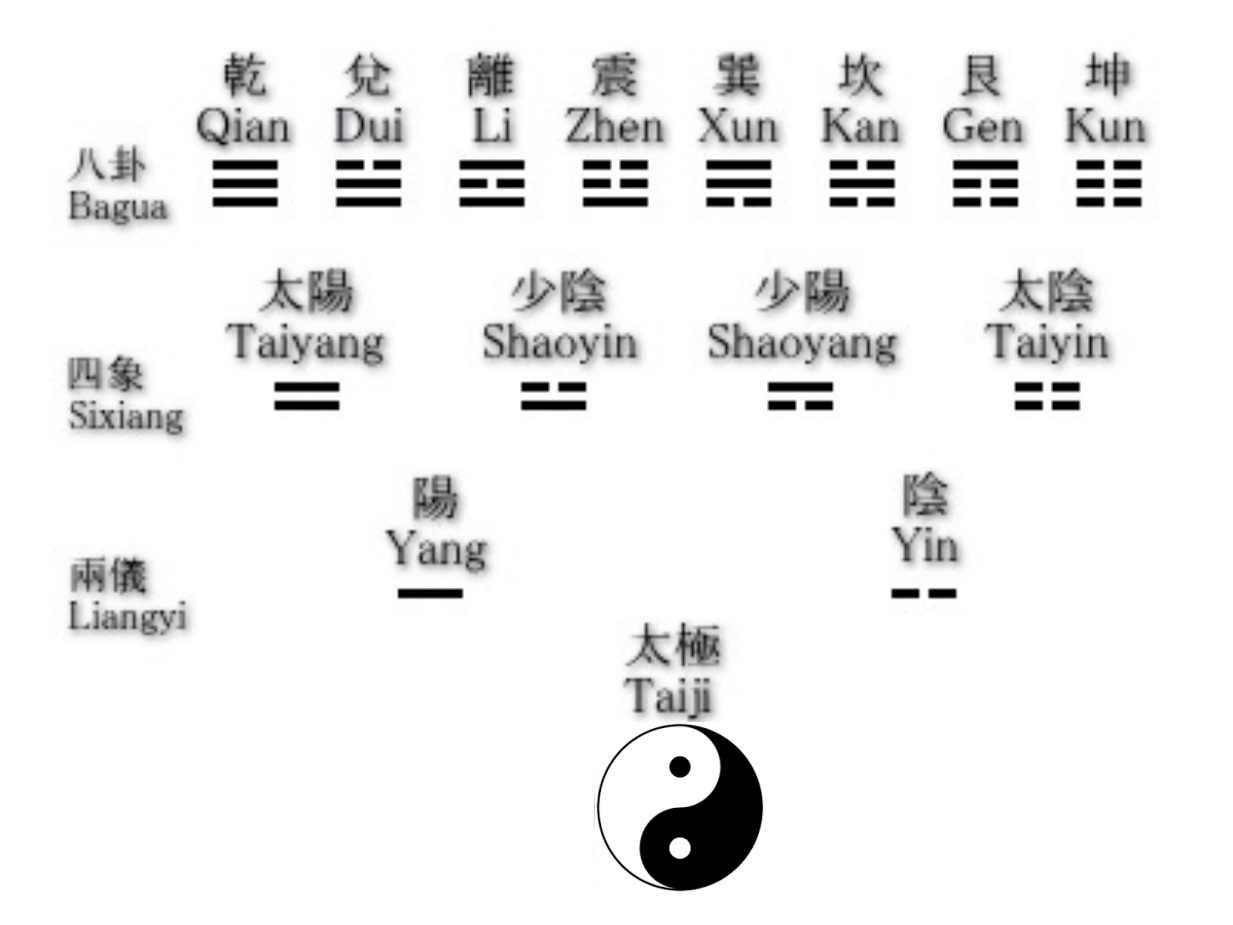
ordinal number	one	two	three	four	five	six	seven	eight	nine
Hindu-Arabic numeral									
modern numeral	1	2	3	4	5	6	7	8	9

The digits 3 and 8 are used to form the numeral 38 for the number 'thirty eight' and the numeral 83 for the number 'eighty three'.

Arabic Numerals based on the number of Angles



The Chinese Binary System



Binary System

In 1666 Leibniz noted with fascination how the I Ching hexagrams correspond to the binary numbers from 000000 to 111111, and concluded that this mapping was evidence of major Chinese accomplishments in the sort of philosophical mathematics he admired

Goittfried Leibniz
(1646-1716)

Leibniz's Dream

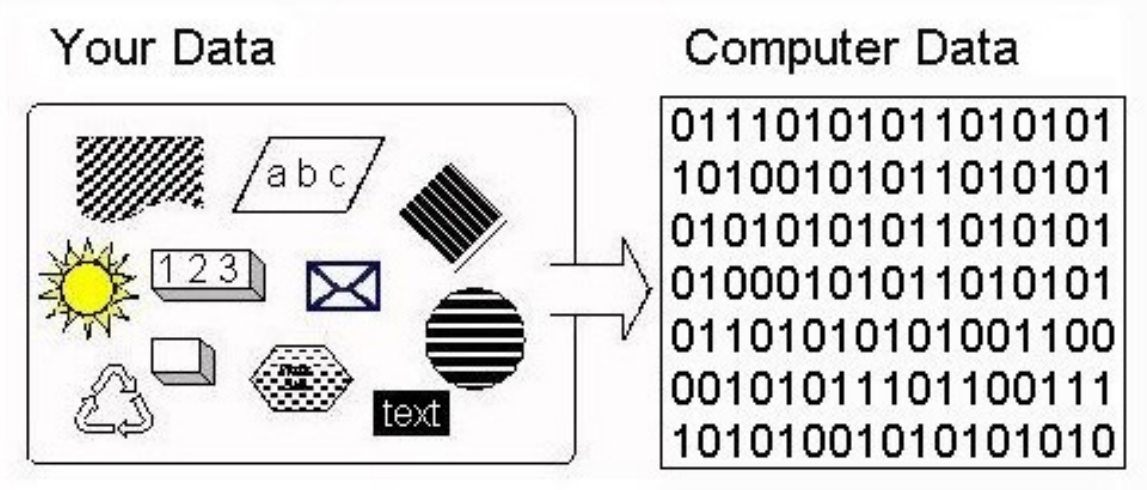
- The first man who dreamed for a general problem solver - a logical thinking machine that could "calculate everything" using binary numbers
- *If we had such an universal tool, we could discuss the problems of the metaphysical or the questions of ethics in the same way as the problems and questions of mathematics or geometry. That was my aim: Every misunderstanding should be nothing more than a miscalculation (...), easily corrected by the grammatical laws of that new language. Thus, in the case of a controversial discussion, two philosophers could sit down at a table and just calculating, like two mathematicians, they could say, 'Let us check it up'*

Binary Numbers: Bits

- Computers are **binary machines**: only 0s and 1s
- Think: Why do computers use binary system?
- BIT = Binary digITs; 1 bit: 0 or 1
- 1 Byte = 8 bits
- A word is a fixed-sized piece of data handled as a unit

Data Representation

- Data Representation refers the methods used internally to represent information stored in the computer (in binary)



Numeric Data Representation

- Unsigned integers
- Signed integers
  - Sign-magnitude
  - 1's complement
  - 2's complement
- Real number representation
- Floating-point numbers

Unsigned Integer Representation

- How many different things can *n* bits represent?
- What is the range of non-negative numbers that can be represented by *n* bits?
- $10110101_2 = (\_\_\_?\_\_\_)_{10} = 0x\_\_?\_\_\_$

Signed Integer Representations

- The left most bit (the most significant bit) is used as the sign bit
- By convention, a value of 0 in the sign bit indicates a positive number, whereas a value of 1 indicates a negative number
  - Sign-magnitude: the remaining bits indicate the magnitude
  - 1's complement: the complement of its positive counterpart
  - 2's complement: one greater than the 1's complement of the positive value

Base 10: decimal

1000's column 100's column 10's column 1's column

$$5\,3\,7\,4_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

Five thousands | Three hundreds | Seven tens | Four ones

Base 2: binary

8's column 4's column 2's column 1's column

$$1\,1\,0\,1_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

One eight | One four | No two | One one

Base 16 or Hexadecimal

{ 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F }

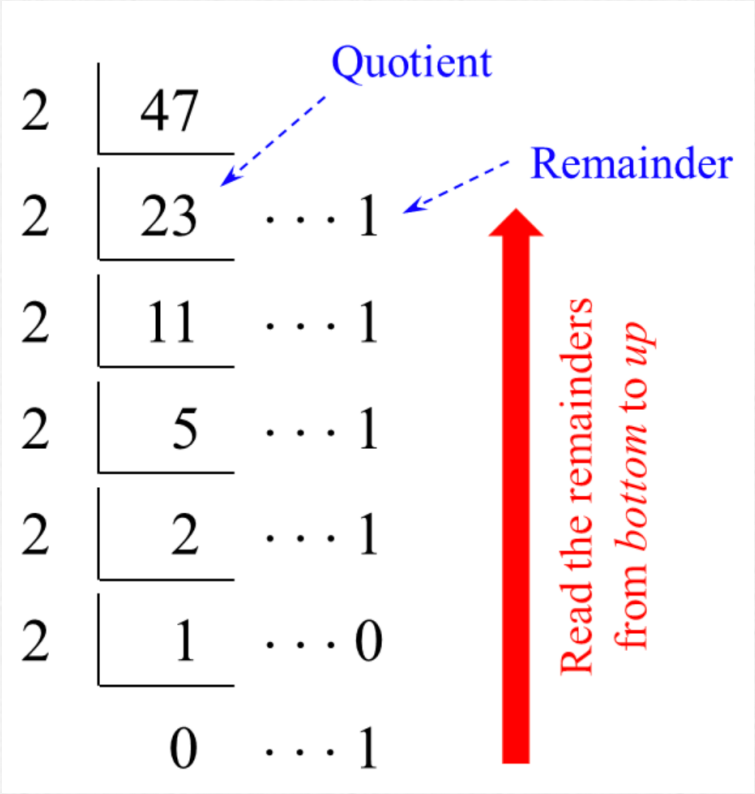
$$
\begin{aligned}
&1C2 \\
&= (1 \cdot 16^2) + (C \cdot 16^1) + (2 \cdot 16^0) \\
&= (1 \cdot 16^2) + (12 \cdot 16^1) + (2 \cdot 16^0) \\
&= (1 \cdot 256) + (12 \cdot 16) + (2 \cdot 1) \\
&= 256 + 192 + 2 \\
&= 450
\end{aligned}
$$

```
Base-16 | A    C    D    C
Base-2  | 1010 1100 1101 1100

Base-16 | 1    0
Base-2  | 0001 0000
```

Base-10	Base-2	Base-16
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Number Conversion



Base 10: 47

Base 2: 101111

Base 16: 2F or 0x2F

Two's complement representation

- To get the two's complement of a negative binary number, all bits are inverted, or "flipped", by using the bitwise NOT operation (one's complement);
- the value of 1 is then added to the resulting value, ignoring the overflow which occurs when taking the two's complement of 0.

EXAMPLE: *Find the two's complement of –17*

Step 1: $17_{10} = 0001\ 0001_2$

Step 2: Take the complement: 1110 1110

Step 3: Add 1: 1110 1110 + 1 = 1110 1111.

Example 1

- What does the binary number 10110101_2 represent?
  - unsigned integer: = (___?___)$_{10}$
  - Sign-magnitude: = (___?___)$_{10}$
  - 1's complement: =(___?___)$_{10}$
  - 2's complement: =(___?___)$_{10}$

Example 2

- Represent 53 into binary
  - unsigned integer: (_____?_____)$_2$
  - Sign-magtinute: (_____?_____)$_2$
  - 1's complement: (_____?_____)$_2$
  - 2's complement: (_____?_____)$_2$

Example 3

- Represent -53 into binary
  - Sign-magtinute: (_____?_____)$_2$
  - 1's complement: (_____?_____)$_2$
  - 2's complement: (_____?_____)$_2$

Real Numbers Representation

- Can all integers be represented in binary?
- Can all real numbers be represented in binary?
- Positive binary fractions (fixed point)
  - $(6.625)_{10} = (\_\_?\_\_\_)_2$
  - $(0.100\ 1001) = (\_\_\_?\_\_\_)_{10}$
  - Approximate 0.9 as a binary fraction (use 8 bits)

Floating Point Numbers

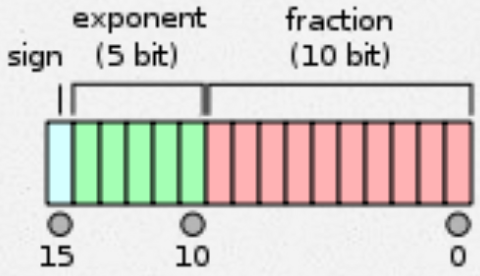
- Floating point: a number's radix point can "float", i.e., it can be placed anywhere relative to the significant digits of the numbers
- Scientific notation: $1500 = 1.5 \times 10^3 = 0.15 \times 10^4 = 15 \times 10^2$

$$\text{significand} \times \text{base}^{\text{exponent}}$$

$$1.2345 = \underbrace{12345}_{\text{significand}} \times \underbrace{10}_{\text{base}}^{\overbrace{-4}^{\text{exponent}}}$$

Half Precision Floating Numbers

- base 2, 16 bits
  - 5 for Exponent;
  - 1 for the sign bit;
  - 10 for significand (fraction, or mantissa)
  - Exponent bias (offset): $2^4 - 1 = 15$; range: [-15, 16]
  - Leading bit convention: the leading bit of the significand is always 1 (thus it needs not be represented in memory)

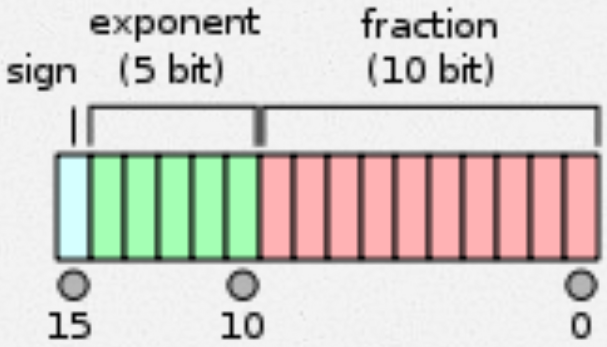


$$(-1)^{\text{sign bit}} * (1+\text{fraction}) * 2^{\text{exponent} - \text{bias}}$$

Example 1

- 0 01111 0000000000 = (__?__)$_{10}$
- Sign bit: 0
- Exponent: 01111 → 0 (offset 15)
- Mantissa: 0000000000 → 1.0 (hidden leading bit)

- + 1.0 x 2^0 = 1

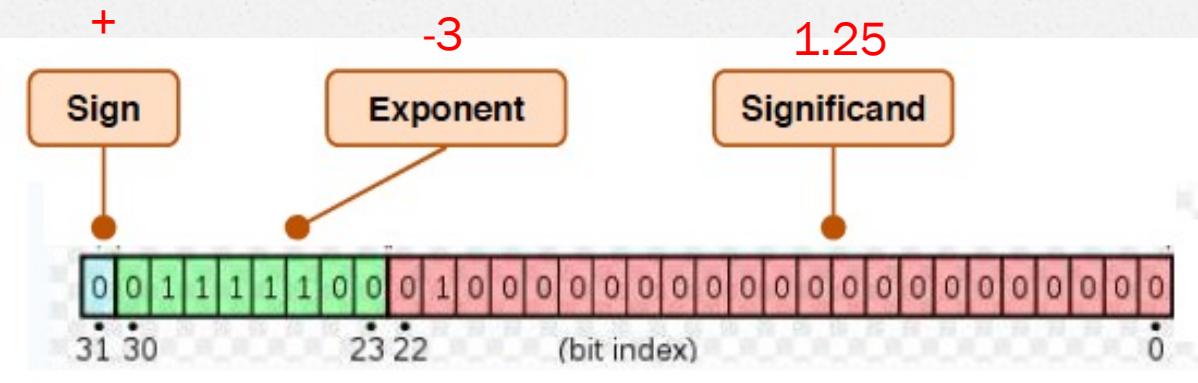


Example 2: IEEE754 Single Point Precision Numbers

- 32 bits: 1 for sign; 8 for exponent; 23 for Mantissa
- What is the Exponent Bias?
  - $2^7 - 1 = 127$
- 12.375 = (____?____)$_2$
  - $12.375 = (1100.011)_2 = 1.100011 \times 2^3$
  - Sign bit: 0;
  - Exponent: 3+ 127 = 130 → 1000 0010
  - Mantissa: 1000 1100 0000 0000 000
- Answer: 0100 0001 0100 0110 0000 0000 0000 0000
  = 0x41460000

In Class Exercise 1

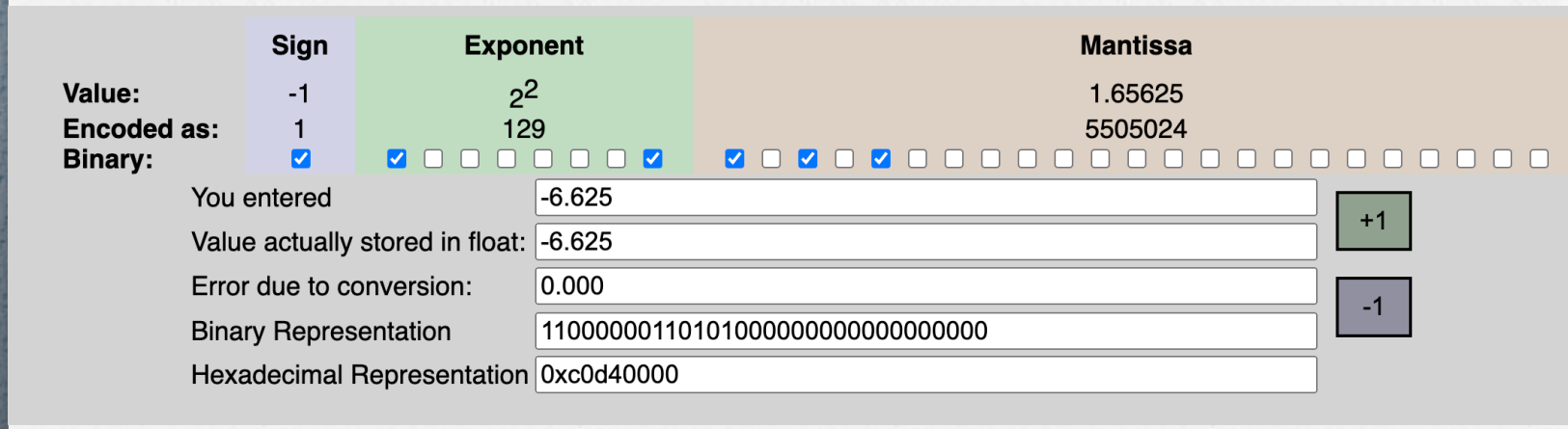
- What number is this?



Answer: $+1.25 \times 2^{-3} = 0.15625$

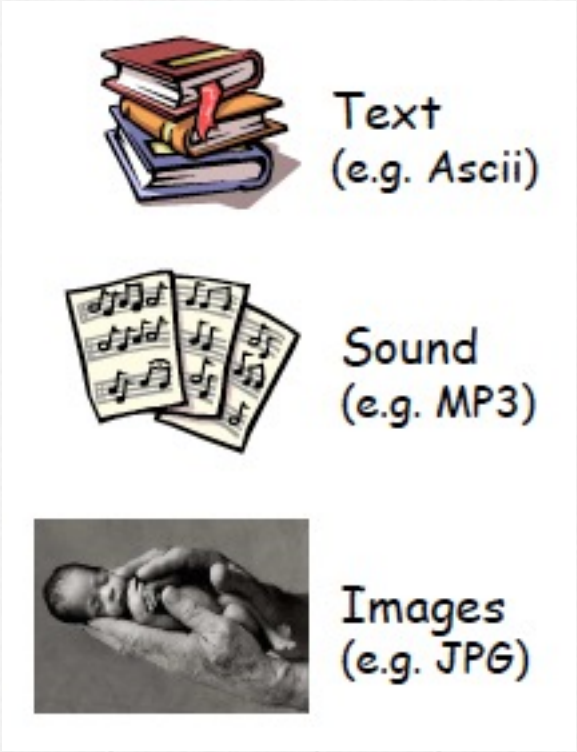
In Class Exercise 2

- What is -6.625's 32 bits floating point representation?



Non-numeric Data Representation

- Textual information
- Audio information
- Colors & Images
- Video information



ASCII Code

- the American Standard Code for Information Interchange
- 7 bits to represent 128 characters
  - 0~31: control characters
  - 32~127: symbols, digits and letters
- A -----> 0100 0001
- What is the ASCII code for the letter E? And the letter e?

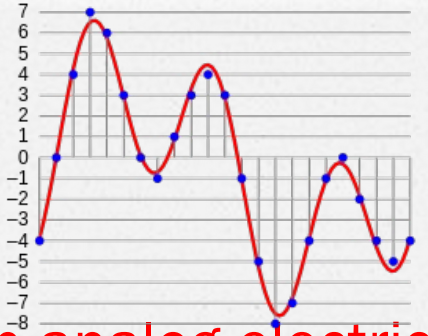
ASCII Code

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Universal Character Set

- 16 bits
- Support unlimited characters
- To permit backward compatibility, ASCII is a subset of Unicode.
- What is the following message?
  - U+0048 U+0065 U+006C U+006C U+006F

Digital Audio



- A microphone converts sound to an analog electrical signal
- An analog-to-digital converter (ADC) converts the analog signal to a digital signal through sampling
- CD audio, for example, has a sampling rate of 44.1 kHz (44,100 samples per second)
- An digital-to-analog converter performs the reverse process, from a digital to an analog signal
- An analog signal can be amplified and send to a speaker to produce sound

Image & Video

- A video consists of a stream of frames, or images, displayed at n>16 frames per second
- An Image consist of a collection of pixels
- Pixels are tiny dots of color
- A pixel's color is represented by a binary number, its RGB value
- Thus a video can be seen as a huge binary number

Operations on Bits

- Binary Arithmetic Operations
  - Addition and subtraction
  - Sign Extension
  - Overflow
- Boolean Logic Operations
  - AND
  - OR
  - NOT
  - XOR

Binary Addition Rules

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0, with a carry bit to the next more significant bit;
  Sum = 0 , Carry = 1

Exercises

- 01001011 + 00101110 = ?

Binary Subtraction Rules

- 0 - 0 = 0
- 1 - 0 = 1
- 1 - 1 = 0
- 0 - 1 = 1, with a borrow bit from the next more significant bit;
  difference = 1, borrow =1

Binary Subtraction using 2's Complement Representation

- The decimal subtraction 29 - 7 = 22 is the same as adding (29) + (-7) = 22
- Convert the number to be subtracted to its two's complement:

  29 →00011101,

  -7 → 11111001
- Add these two numbers (ignore the final carry bit)

Exercises

- Try out the following additions in 8-bit 2's complement:
  - 24 + 9
  - 24 – 9
  - 9 – 24
  - – 24 – 9

Sign Extension

- To extend a signed integer from 8 bits to 16 bits or from 16 bits to 32 bits, append additional bits on the left side of the number.
- Fill each extra bit with the value of the smaller number's most significant bit (the sign bit).
- 1111 1111 ---> 1111 1111 1111 1111
- 0000 0001 ---> 0000 0000 0000 0001

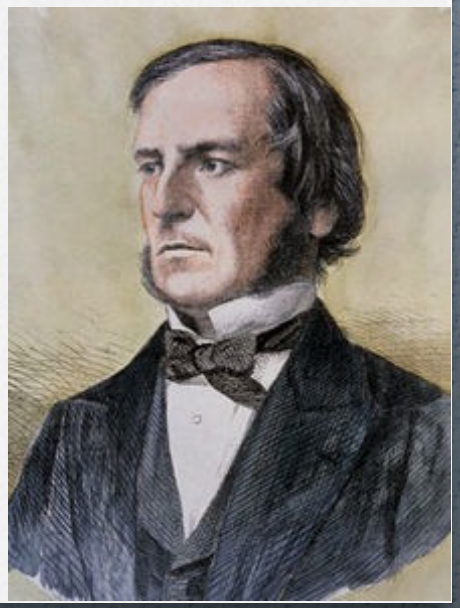
Overflow

- Overflow occurs when a calculation produces a result that is greater in magnitude than what a given register or storage location can store or represent

- For example, 01101011 + 00101110 = ?

Boolean Logic

- developed by George Boole (1815-1864) in the mid 1800s
- An Investigation of the Laws of Thought (1854)
- all values are either TRUE or FALSE
- the basis of modern digital computer logic



Logic Operations on Bits

- The AND operation
  - 0 and 0 = 0
  - 0 and 1 = 0
  - 1 and 0 = 0
  - 1 and 1 = 1
- The OR operation
  - 0 or 0 = 0
  - 0 or 1 = 1
  - 1 or 0 = 1
  - 1 or 1 = 1
- The NOT operation
  - Not 0 = 1
  - Not 1 = 0
- The XOR operation
  - 0 xor 0 = 0
  - 0 xor 1 = 1
  - 1 xor 0 = 1
  - 1 xor 1 = 0

Truth Table

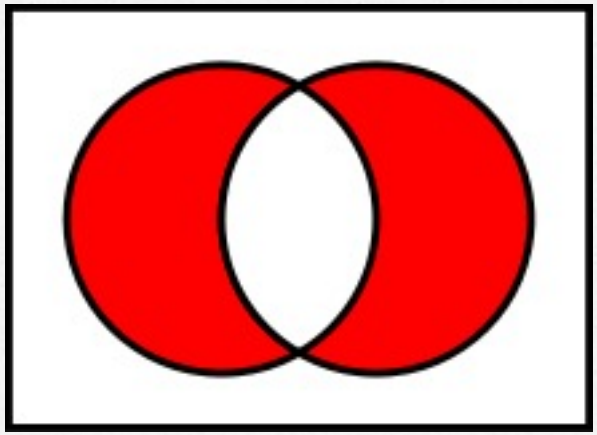
x	y	$x \wedge y$	$x \vee y$
0	**0**	0	0
1	**0**	0	1
0	**1**	0	1
1	**1**	1	1

x	$\neg x$
0	1
1	0

XOR

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

$A \oplus B$



Exercise

- What is the truth table for single bit binary addition A + B ?

What's Next

- From bits to Boolean logic