Fundamentals of Computer Systems

Thinking Digitally

Stephen A. Edwards

Columbia University

Summer 2020

The Subject of this Class

0

The Subjects of this Class

0 1

But let your communication be, Yea, yea; Nay, nay: for whatsoever is more than these cometh of evil.

— Matthew 5:37


SUPERCODER 2000
Air cooled coding keyboard for professional use.
Done
0
1

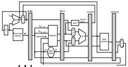
Engineering Works Because of Abstraction

Application Software

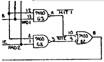
Operating Systems

```
;; voice 1 wave select
ld      a,(#CH1_W_NUM)
and     a
ld      a,(#CH1_W_SEL)
jr      nz,#00b4
ld      a,(#CH1_E_TABLE0)
```

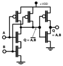
Architecture



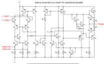
Micro-Architecture



Logic

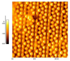


Digital Circuits



Analog Circuits

Devices



Physics

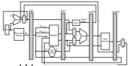
Engineering Works Because of Abstraction

Application Software — COMS 3157, 4156, et al.

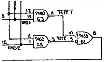
Operating Systems — COMS W4118

```
;; voice 1 wave select
ld      a,(#CH1_W_NUM)
and     a
ld      a,(#CH1_W_SEL)
jr      nz,#00b4
ld      a,(#CH1_E_TABLE0)
```

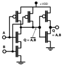
Architecture — Second Half of 3827



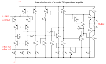
Micro-Architecture — Second Half of 3827



Logic — First Half of 3827

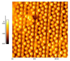


Digital Circuits — First Half of 3827



Analog Circuits — ELEN 3331

Devices — ELEN 3106



Physics — ELEN 3106 et al.

Boring Stuff

http://www.cs.columbia.edu/~sedwards/classes/2020/3827-summer/

Prof. Stephen A. Edwards

sedwards@cs.columbia.edu

Lectures 1:00 – 4:00 PM, Mondays and Wednesdays
May 27–July 1

Weight	What	When
40%	Homeworks	See Webpage
60%	Final exam	July 1st

Submit homework online via Courseworks

Software You Need

The Inkscape SVG File Editor inkscape.org

Do homework by downloading an SVG file from the class website, edit it in Inkscape, and upload it to Courseworks

The Digital Circuit Simulator github.com/hneemann/Digital

Circuit design problems: download (class website) .zip file with .dig files, edit with Digital, upload to Courseworks

SPIM: A MIPS32 Simulator spimsimulator.sourceforge.net

MIPS assembly coding:, download .zip file with .s files, edit in favorite text editor, test and debug in SPIM, upload to Courseworks

Rules and Regulations

Each assignment turned in must be unique; work must ultimately be your own.

Don't cheat: Columbia Students Aren't Cheaters

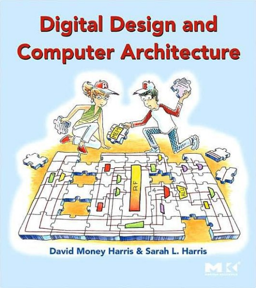
Test will be closed-book; you may use a single sheet of your own notes

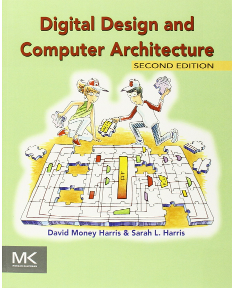
Optional Texts: Alternative 1

No required text. One option:

- David Harris and Sarah Harris. *Digital Design and Computer Architecture*. Either 1st or 2nd ed.

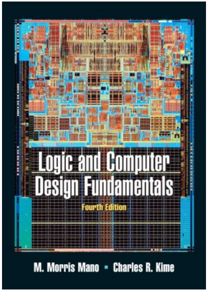
  Almost precisely right for the scope of this class: digital logic and computer architecture.



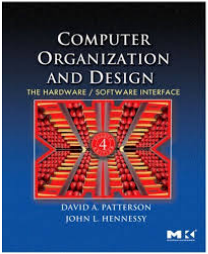


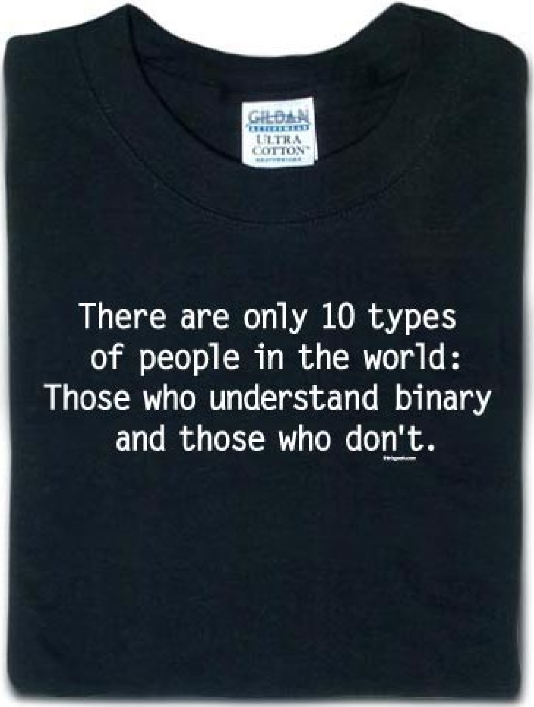
Optional Texts: Alternative 2

- M. Morris Mano and Charles Kime. *Logic and Computer Design Fundamentals.* 4th ed.



- David A. Patterson and John L. Hennessy. *Computer Organization and Design, The Hardware/Software Interface.* 4th ed.

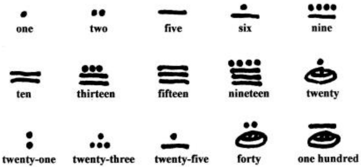




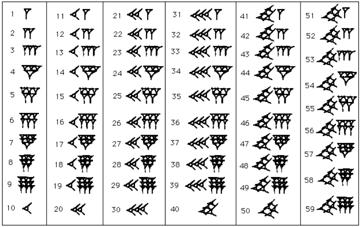
thinkgeek.com

Which Numbering System Should We Use?

Roman: I II III IV V VI VII VIII IX X



Mayan: base 20, Shell = 0



Babylonian: base 60

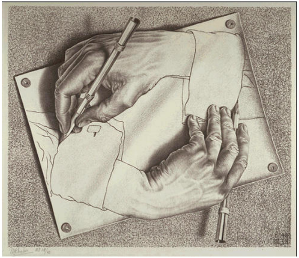
The Decimal Positional Numbering System

Ten figures: 0 1 2 3 4 5 6 7 8 9

$$730_{10} = 7 \times 10^2 + 3 \times 10^1 + 0 \times 10^0$$

$$990_{10} = 9 \times 10^2 + 9 \times 10^1 + 0 \times 10^0$$

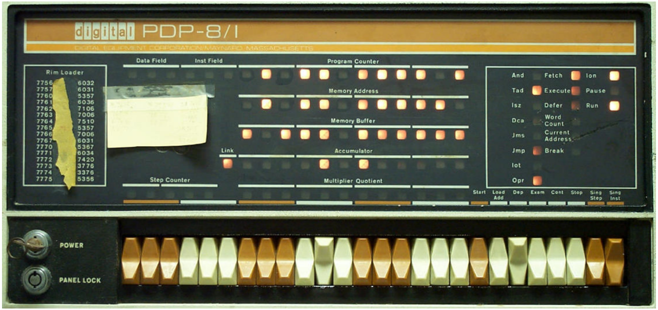
Why base ten?



Hexadecimal, Decimal, Octal, and Binary

Hex	Dec	Oct	Bin
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

Binary and Octal: Electronics Likes Powers of Two



DEC PDP-8/I, c. 1968

Oct	Bin
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111

$$
\begin{aligned}
\text{PC} &= 010\,110\,111\,101_2 \\
&= 0 \times 2^{11} + 1 \times 2^{10} + 0 \times 2^{9} + 1 \times 2^{8} + 1 \times 2^{7} + 0 \times 2^{6} + \\
&\quad 1 \times 2^{5} + 1 \times 2^{4} + 1 \times 2^{3} + 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0} \\
&= 2675_8 \\
&= 2 \times 8^3 + 6 \times 8^2 + 7 \times 8^1 + 5 \times 8^0 \\
&= 1469_{10}
\end{aligned}
$$

Hexadecimal Numbers

Base 16: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Instead of groups of 3 bits (octal), Hex uses groups of 4.

$$\begin{aligned}\texttt{CAFEF00D}_{16} &= 12 \times 16^7 + 10 \times 16^6 + 15 \times 16^5 + 14 \times 16^4 + \\ &\quad 15 \times 16^3 + 0 \times 16^2 + 0 \times 16^1 + 13 \times 16^0 \\ &= 3{,}405{,}705{,}229_{10}\end{aligned}$$

C	A	F	E	F	0	0	D	Hex

11001010111111101111000000001101 Binary

3	1	2	7	7	5	7	0	0	1	5	Octal

Computers Rarely Manipulate True Numbers

Infinite memory still very expensive

Finite-precision numbers typical

32-bit processor: naturally manipulates 32-bit numbers

64-bit processor: naturally manipulates 64-bit numbers

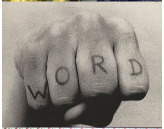
How many different numbers can you

represent with 5 {binary, octal, decimal, hexadecimal} digits?

Jargon

Bit — Binary digit: 0 or 1

Byte — Eight bits



Word — Natural number of bits for the processor, e.g., 16, 32, 64

LSB — Least Significant Bit (“rightmost”)

MSB — Most Significant Bit (“leftmost”)

Decimal Addition Algorithm

$$\begin{array}{r} 434 \\ +628 \\ \hline \end{array}$$

$$4 + 8 = 12$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$\begin{array}{r} 1 \\ 434 \\ +628 \\ \hline 2 \end{array}$$

$$4 + 8 = 12$$

$$1 + 3 + 2 = 6$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$\begin{array}{r} \color{red}{1} \\ 434 \\ +628 \\ \hline \color{blue}{62} \end{array}$$

$$4 + 8 = \color{red}{1}\color{blue}{2}$$

$$1 + 3 + 2 = \color{blue}{6}$$

$$4 + 6 = \color{red}{10}$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$
\begin{array}{r}
1\ \ 1 \\
434 \\
+628 \\
\hline
062
\end{array}
$$

$$4+8 = 12$$

$$1+3+2 = 6$$

$$4+6 = 10$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Decimal Addition Algorithm

$$
\begin{array}{r}
1\ \ 1\ \ \\
434 \\
+628 \\
\hline
1062
\end{array}
$$

$$4 + 8 = 12$$

$$1 + 3 + 2 = 6$$

$$4 + 6 = 10$$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18
10	10	11	12	13	14	15	16	17	18	19

Binary Addition Algorithm

$$\begin{array}{r} 10011 \\ +11001 \\ \hline \end{array}$$

$$1 + 1 \quad = \quad {\color{red}1}{\color{blue}0}$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$\begin{array}{r} \color{red}{1} \\ 10011 \\ +11001 \\ \hline \color{blue}{0} \end{array}$$

$$1 + 1 = \color{red}{1}\color{blue}{0}$$

$$1 + 1 + 0 = \color{red}{1}\color{blue}{0}$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$
\begin{array}{r}
11 \\
10011 \\
+11001 \\
\hline
00
\end{array}
$$

$$
\begin{aligned}
1+1 &= 10 \\
1+1+0 &= 10 \\
1+0+0 &= 01
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$
\begin{array}{r}
011 \\
10011 \\
+11001 \\
\hline
100
\end{array}
$$

$$
\begin{aligned}
1+1 &= 10 \\
1+1+0 &= 10 \\
1+0+0 &= 01 \\
0+0+1 &= 01
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Binary Addition Algorithm

$$\begin{array}{r} 0011 \\ 10011 \\ +11001 \\ \hline 1100 \end{array}$$

+	0	1
0	00	01
1	01	10
10	10	11

$$1 + 1 = 10$$

$$1 + 1 + 0 = 10$$

$$1 + 0 + 0 = 01$$

$$0 + 0 + 1 = 01$$

$$0 + 1 + 1 = 10$$

Binary Addition Algorithm

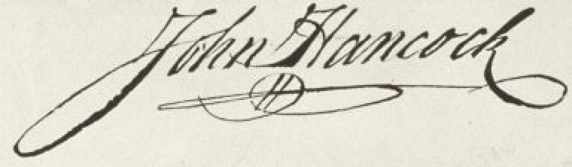
$$
\begin{array}{r}
10011 \\
10011 \\
+11001 \\
\hline
101100
\end{array}
$$

$$
\begin{aligned}
1+1 &= 10 \\
1+1+0 &= 10 \\
1+0+0 &= 01 \\
0+0+1 &= 01 \\
0+1+1 &= 10
\end{aligned}
$$

+	0	1
0	00	01
1	01	10
10	10	11

Signed Numbers: Dealing with Negativity

How should we represent negative numbers?



Binary Signed Magnitude Numbers

The familiar notation: negative numbers have a leading –

Binary signed-magnitude encoding: leading 1 indicates negative; remaining bits treated as binary.

$0000_2 = 0$

$0010_2 = 2$

$1010_2 = -2$

$1111_2 = -7$

$1000_2 = -0?$

Can be made to work, but addition is annoying:

If the signs match, add the magnitudes and use the same sign.

If the signs differ, subtract the smaller number from the larger; return the sign of the larger.

One's Complement Numbers

Like Signed Magnitude, a leading 1 indicates a negative One's Complement number. However, number magnitude is *complement* of remaining bits interpreted as binary.

To negate a number, complement (flip) each bit.

$0000_2 = 0$

$0010_2 = 2$

$1101_2 = -2$

$1000_2 = -7$

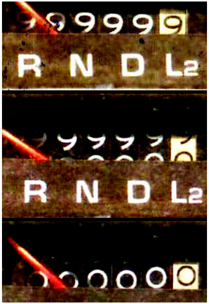
$1111_2 = -0?$

Addition is nicer: just add the one's complement numbers as if they were normal binary.

Really annoying having a −0: two numbers are equal if their bits are the same or if one is 0 and the other is −0.


NOT ALL
ZEROS
ARE CREATED
EQUAL
0 CALORIE COLA
pepsi MAX
zero calories
maximum taste
ZERO CALORIES. MAXIMUM PEPSI® TASTE.
pepsi MAX
zero calories

Two's Complement Numbers



Really neat trick: just make only the most significant bit represent a *negative* number instead of positive; treat the rest as binary.

$1101_2 = -8 + 4 + 1 = -3$

$1111_2 = -8 + 4 + 2 + 1 = -1$

$0111_2 = 4 + 2 + 1 = 7$

$1000_2 = -8$

Easy addition: just add in binary and discard any carry.

Negation: complement each bit (as in one's complement) then add 1.

Subtraction done with negation and addition.

Very good property: no -0

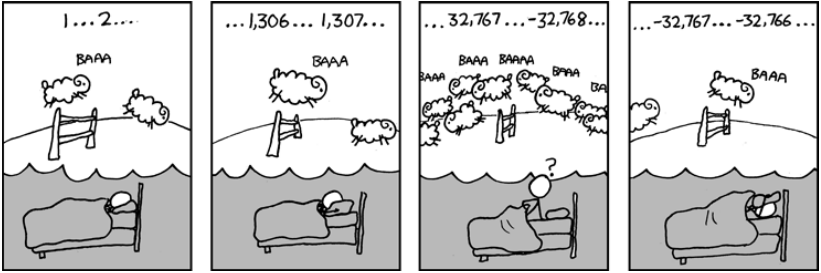
Two's complement numbers are equal if and only if all their bits are the same.

Number Representations Compared

Code	Binary	Signed Mag.	One's Comp.	Two's Comp.
0000	0	0	0	0
0001	1	1	1	1
⋮				
0111	7	7	7	7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
⋮				
1110	14	−6	−1	−2
1111	15	−7	−0	−1

Smallest number

Largest number



https://xkcd.com/571/

How many bits in his brain?

Fixed-point Numbers

How to represent fractional numbers? In decimal, we continue with negative powers of 10:

$$
\begin{aligned}
31.4159 &= 3 \times 10^1 + 1 \times 10^0 + \\
&\quad 4 \times 10^{-1} + 1 \times 10^{-2} + 5 \times 10^{-3} + 9 \times 10^{-4}
\end{aligned}
$$

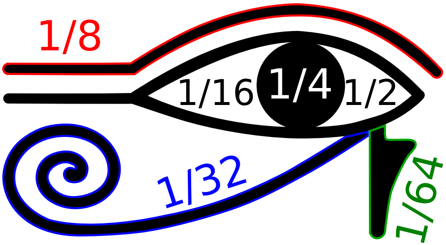
Also works in binary:

$$
\begin{aligned}
1011.0110_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + \\
&\quad 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} \\
&= 8 + 2 + 1 + 0.25 + 0.125 \\
&= 11.375
\end{aligned}
$$

Addition and subtraction algorithms the same.

F
F a
u c
Interesting

The ancient Egyptians used binary fractions:



The Eye of Horus

Binary-Coded Decimal

thinkgeek.com

Humans prefer reading decimal numbers; computers prefer binary.

BCD is a compromise: every four bits represents a decimal digit.

Dec	BCD
0	0000 0000
1	0000 0001
2	0000 0010
⋮	⋮
8	0000 1000
9	0000 1001
10	0001 0000
11	0001 0001
⋮	⋮
18	0001 1000
19	0001 1001
20	0010 0000
⋮	⋮

BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

$$\begin{array}{r} 158 \\ +242 \\ \hline \end{array}$$

$$\begin{array}{rl} 0001\,0101\,1000 & \\ +0010\,0100\,0010 & \\ \hline 1010 & \text{First group} \end{array}$$

BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

$$
\begin{array}{r}
158 \\
+242 \\
\hline
\end{array}
$$

$$
\begin{array}{rl}
0001\,0101\,1000 & \\
+0010\,0100\,0010 & \\
\hline
1010 & \text{First group} \\
+\,0110 & \text{Correction} \\
\hline
\end{array}
$$

BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

```
  1
 158
+242
----
   0
```

```
         1
 0001 0101 1000
+0010 0100 0010
---------------
           1010   First group
         + 0110   Correction
          -----
      1010 0000   Second group
```

BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

```
  1
  158
 +242
 ----
    0
```

```
                  1
    0001 0101 1000
  + 0010 0100 0010
  ----------------
              1010   First group
            + 0110   Correction
         ---------
         1010 0000   Second group
       + 0110        Correction
  --------------

  ----------------
```

BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

$$
\begin{array}{r}
11 \\
158 \\
+242 \\
\hline
00
\end{array}
$$

$$
\begin{array}{rrrl}
1 & 1 & & \\
0001 & 0101 & 1000 & \\
+0010 & 0100 & 0010 & \\
\hline
 & & 1010 & \text{First group} \\
 & & +\ 0110 & \text{Correction} \\
\hline
 & 1010 & 0000 & \text{Second group} \\
 & +\ 0110 & & \text{Correction} \\
\hline
0100 & 0000 & & \text{Third group} \\
\hline
\end{array}
$$

BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

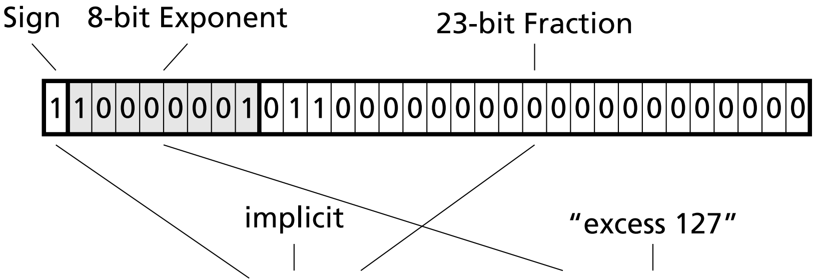
```
 11
 158
+242
----
 400
```

```
   1    1
 0001 0101 1000
+0010 0100 0010
---------------
           1010   First group
         + 0110   Correction
      ---------
      1010 0000   Second group
    + 0110        Correction
 ----------
 0100 0000        Third group
                  (No correction)
---------------
 0100 0000 0000   Result
```

Floating-Point Numbers: "Scientific Notation"

Greater dynamic range at the expense of precision
Excellent for real-world measurements

IEEE 754 Single-Precision (32-bit)



$$= -\ 1.0110000_2 \times 2^{10000001_2 - 127}$$

$$= -1.375 \times 2^2$$

$$= -5.5$$

ASCII For Representing Characters and Strings

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL