Fundamentals of Computer Systems

Thinking Digitally

Stephen A. Edwards

Columbia University

Summer 2021

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


0
1
ENTER
Space!


1001010101
0110001010
1100111000
1001100111
0111000110
1110101010
0011011000
1111010011
01001
0
1

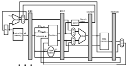
Engineering Works Because of Abstraction

Application Software

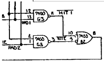
Operating Systems

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

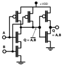
Architecture



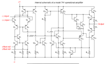
Micro-Architecture



Logic

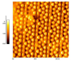


Digital Circuits



Analog Circuits

Devices

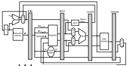
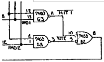
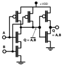
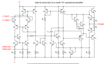
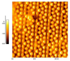


Physics

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	CS :-)

	Micro-Architecture	Second Half of 3827	EE :-(
	Logic	First Half of 3827	EE
	Digital Circuits	First Half of 3827	~~CS~~
	Analog Circuits	ELEN 3331	
	Devices	ELEN 3106	
	Physics	ELEN 3106 et al.	

Boring Stuff

`http://www.cs.columbia.edu/~sedwards/classes/2021/3827-summer/`

Prof. Stephen A. Edwards

sedwards@cs.columbia.edu

Lectures 4:10 – 6:40 PM, Mondays and Wednesdays
May 3–June 14

Weight	What	When
40%	Homeworks	See Webpage
60%	Final exam	June 18th

Submit homework online via Courseworks

Software You Need

Digital 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

The Inkscape SVG File Editor inkscape.org

You can do homework by downloading an SVG file from the class website, editing it in Inkscape, and uploading it 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

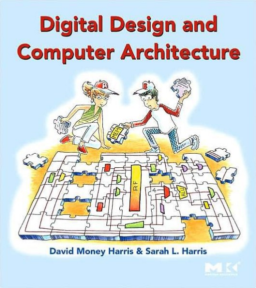
Test will be closed-book; you may use a single sheet of your own notes

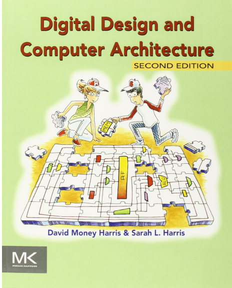
Optional Texts: Alternative 1

No required text. One option:

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

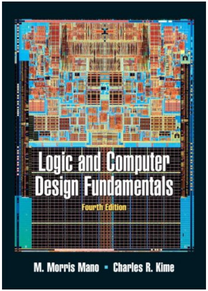
  Almost precisely right for the scope of this class: digital logic and computer architecture.



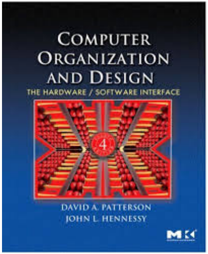


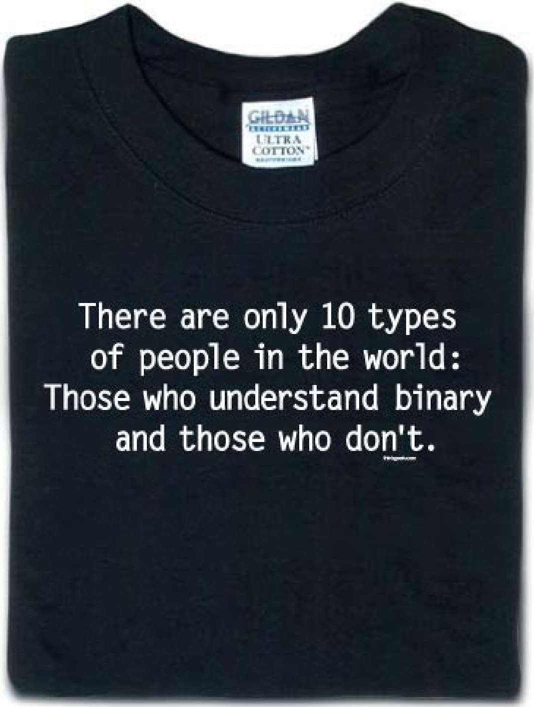
Optional Texts: Alternative 2

- 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.

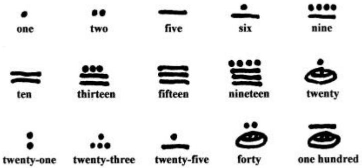




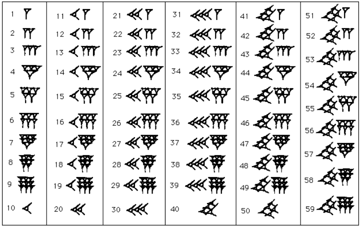
thinkgeek.com

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

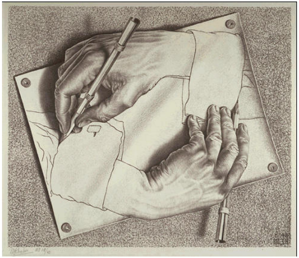
Ten figures: 0 1 2 3 4 5 6 7 8 9

2, 5, 10, ...

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

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

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

Base 16

$2^4 = 16$

$20_8 = 16_{10}$

$10_{16} =$

Base 8

0, 1, ..., 7

8^4 8^3 8^2

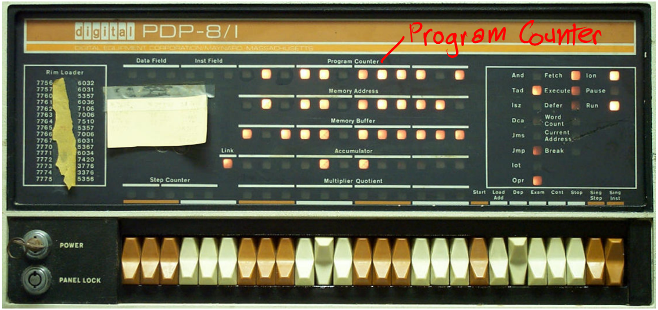
8 64 8s 1s

32 16 8s 4s 2s 1s

second octal digit

one octal digit

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

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

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.

zero

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

| C | A | F | E | F | 0 | 0 | D | Hex

0 11001010111111101111000000001101 Binary

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

Computers Rarely Manipulate True Numbers

Infinite memory still very expensive

$$\left.\begin{matrix}00000\\99999\end{matrix}\right\} 100{,}000$$

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?

100000_2 $\quad 2^5 = 32$

100000_8 $\quad 8^5 = 32768$

$100{,}000_{10}$ $\quad 10^5 =$

100000_{16} $\quad 16^5 =$

one megabyte of memory

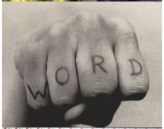
1M

1,048,576

Jargon

Bit — Binary digit: 0 or 1

Byte — Eight bits



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

LSD

$1010_2 = 10_{10}$

0

LSB — Least Significant Bit ("rightmost")

MSG

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

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

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

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

+	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

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

$$4 + 8 = 12$$

$$1 + 3 + 2 = 6$$

$$4 + 6 = 10$$

+	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

$$
\begin{array}{r}
\color{red}{1}\,\color{red}{1} \\
434 \\
+628 \\
\hline
\color{blue}{062}
\end{array}
$$

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

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

$$4+6 = \color{red}{1}\color{blue}{0}$$

+	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

$$\begin{array}{ccccc}
 & 1 & & 1 & \\
 & & 4 & 3 & 4 \\
+ & & 6 & 2 & 8 \\
\hline
 & 1 & 0 & 6 & 2
\end{array}$$

$$4 + 8 = 12$$

$$1 + 3 + 2 = 6$$

$$4 + 6 = 10$$

+	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

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

$$1 + 1 \quad = \quad 10$$

+	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

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

$$1 + 1 = 10$$

$$1 + 1 + 0 = 10$$

$$1 + 0 + 0 = 01$$

+	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

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

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

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

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

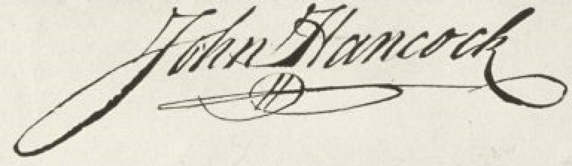
$$1 + 0 + 0 = \color{red}{0}\color{blue}{1}$$

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

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

Signed Numbers: Dealing with Negativity

How should we represent negative numbers?



Binary Signed Magnitude Numbers

$\cancel{-0} = 0$

The familiar notation: negative numbers have a leading –

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

$0 = +$
$1 = -$

$0000_2 = 0$

$0010_2 = 2$

$1010_2 = -2$

$1111_2 = -7$ $\quad -111_2$

$1000_2 = -0?$

Signed magnitude

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

$$\begin{array}{r} 2 \\ + \; -2 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0010 \\ + \; 1101 \\ \hline 1111 \end{array} = -0 = 0$$

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.

$0000_2 = 0$

$0010_2 = 2$

$1101_2 = -2$ -010

$1000_2 = -7$

$1111_2 = -0?$

negative 000

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.

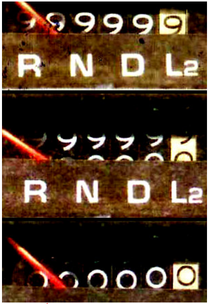
one's comp

0000 = 1111

I hate this


0 CALORIE COLA
pepsi MAX
zero calories
maximum taste
NOT ALL ZEROS ARE CREATED EQUAL
ZERO CALORIES. MAXIMUM PEPSI® TASTE.
pepsi MAX
zero calories
PEPSI, PEPSI MAX and the Pepsi Globe are registered trademarks of PepsiCo, Inc.

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.

-8 4 2 1s

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

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

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

$1000_2 = -8$

1110 = -2
1111 = +,-1
X1101
discard 1101 -3
0000_2 = 0
1+4-8=-3

0111+1 1000 X

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

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

Not as good as 1's complement

0001
1110
+ 1111

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

	16	15	15	16
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

How many we represent

same across all 4

symmetric

awkward

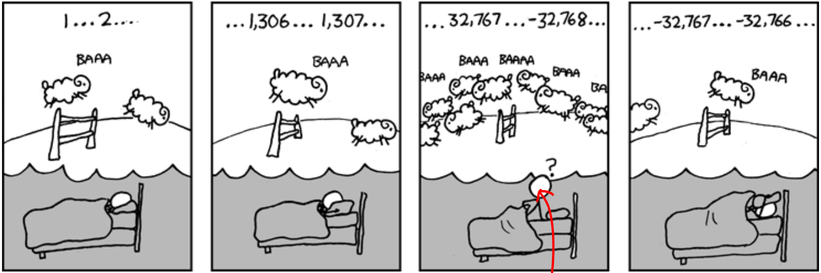
feels nice

Smallest number

Largest number

symmetric

, 32,767 , -32,768,



https://xkcd.com/571/

How many bits in his brain?

How many bits?

16 ~~bits~~, two's complement

Fixed-point Numbers

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

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

tenths hundredths

Also works in binary:

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

halves
quarters
eighths

Addition and subtraction algorithms the same.

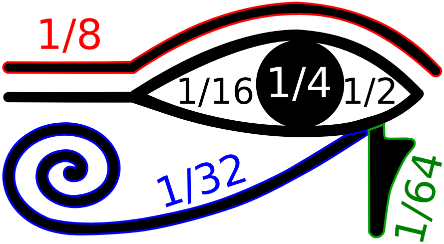
F

F a

u c

Interesting

The ancient Egyptians used binary fractions:



The Eye of Horus

Binary-Coded Decimal

0–9 in hex/binary

12:26:16

hrs min seconds

thinkgeek.com

59

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:

$$\begin{array}{r} 158_{10} \\ +242_{10} \\ \hline \end{array}$$

$$\begin{array}{r} 1 \\ 0001\,0101\,1000 \\ +0010\,0100\,0010 \\ \hline 1010 \end{array}$$ First group

0000

BCD Addition

Binary addition followed by a possible correction.

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

Example:

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

$$\begin{array}{rl} \overset{1}{} \\ 0001\,0101\,1000 & \\ +0010\,0100\,0010 & \\ \hline 1010 & \text{First group} \\ +\,0110 & \text{Correction} \\ \hline \\ \hline 0000 & \end{array}$$

>9 so add 6

BCD Addition

Binary addition followed by a possible correction.

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

Example:

$$
\begin{array}{r}
1 \\
158 \\
+242 \\
\hline
0
\end{array}
$$

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

BCD Addition

Binary addition followed by a possible correction.

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

Example:

```
  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:

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

$$
\begin{array}{rrrl}
1 & 1 & & \\
0001 & 0101 & 1000 & \\
+0010 & 0100 & 0010 & \\
\hline
 & & 1010 & \text{First group} \\
 & & +\,0110 & \text{Correction} \\
\cline{3-3}
 & 1010 & 0000 & \text{Second group} \\
 & +\,0110 & & \text{Correction} \\
\cline{1-2}
0100 & 0000 & & \text{Third group} \\
\hline
\end{array}
$$

BCD Addition

10+6 = 16

11+6 = 17

16 combinations ~~codes~~ patterns

−10 symbols

6 correction

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
     4    0    0
```

Floating-Point Numbers: “Scientific Notation”

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

$-\underbrace{1.356}_{\text{mantissa}} \times 10^{-13}$

IEEE 754 Single-Precision (32-bit)

Sign	8-bit Exponent	23-bit Fraction
1	1 0 0 0 0 0 0 1	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

explicit sign bit

implicit

“excess 127”

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

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

$$= -5.5$$

1×10^2

0.35×10^1

1.00
+.035
1.035×2

0.035×10^2

ASCII For Representing Characters and Strings

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