Warm-Up: Back to Elementary School

Without using a calculator add:

- 26577 and 952
- 99645 and 468
- 637 and 203
- 99 and 80

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

$$\begin{array}{r} 26577 \\ +\quad 952 \\ \hline \end{array}$$

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

$$\begin{array}{r} 26577 \\ +\quad 952 \\ \hline 9 \end{array}$$

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

$$\begin{array}{r} 1 \\ 26577 \\ +\quad 952 \\ \hline 29 \end{array}$$

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

```
   11
  26577
+   952
  -----
    529
```

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

$$
\begin{array}{r}
11 \\
26577 \\
+\quad 952 \\
\hline
7529
\end{array}
$$

Warm-Up: Back to Elementary School

Without using a calculator, how would you add 26577 and 952?

$$
\begin{array}{r}
{\scriptstyle 1\,1} \\
26577 \\
+\quad 952 \\
\hline
27529
\end{array}
$$

Warm-Up: Back to Elementary School

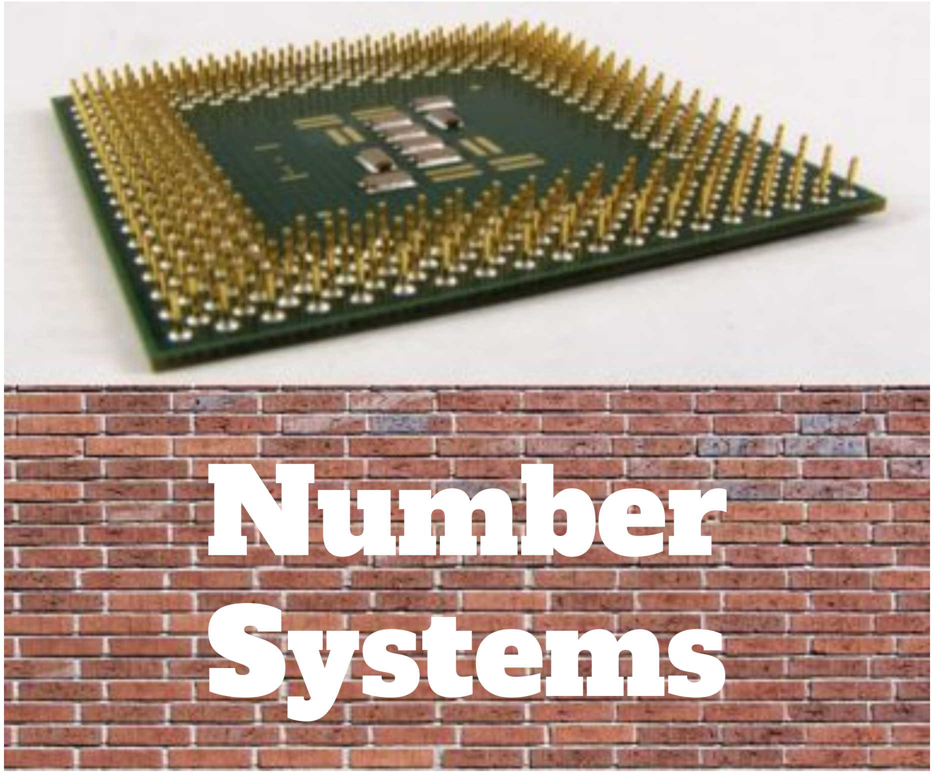
Without using a calculator add:

- 26577 and 952
- 99645 and 468
- 637 and 203
- 99 and 80

Warm-Up: Back to Elementary School

What is the sum of each pair? And how many times do you "carry the 1"?

- 26577 and 952 `27529; 2`
- 99645 and 468
- 637 and 203
- 99 and 80



The foundation upon which Computer Architecture is built

Unsigned Number Systems

For HW0 you ran one of:

Java Program

```
import java.io.*;
public class hw0
{
  public static void main(String args[])
  {
    short a;                        //declare 'a'
    a = 20000;                      //set to 2000
    a = (short)(a + a);             //double 'a'
    System.out.println(a);          //display 'a'
  }
}
```

C++ Program

```
#include <iostream>
int main()
  {
    short int a;                        //declare 'a'
    a = 20000;                          //set to 20000
    a = a + a;                          //double 'a'
    std::cout << a << std::endl;        //display 'a'
    return (0);                         //exit
  }
```

```
INPUT A
B = A + A
OUTPUT B
```

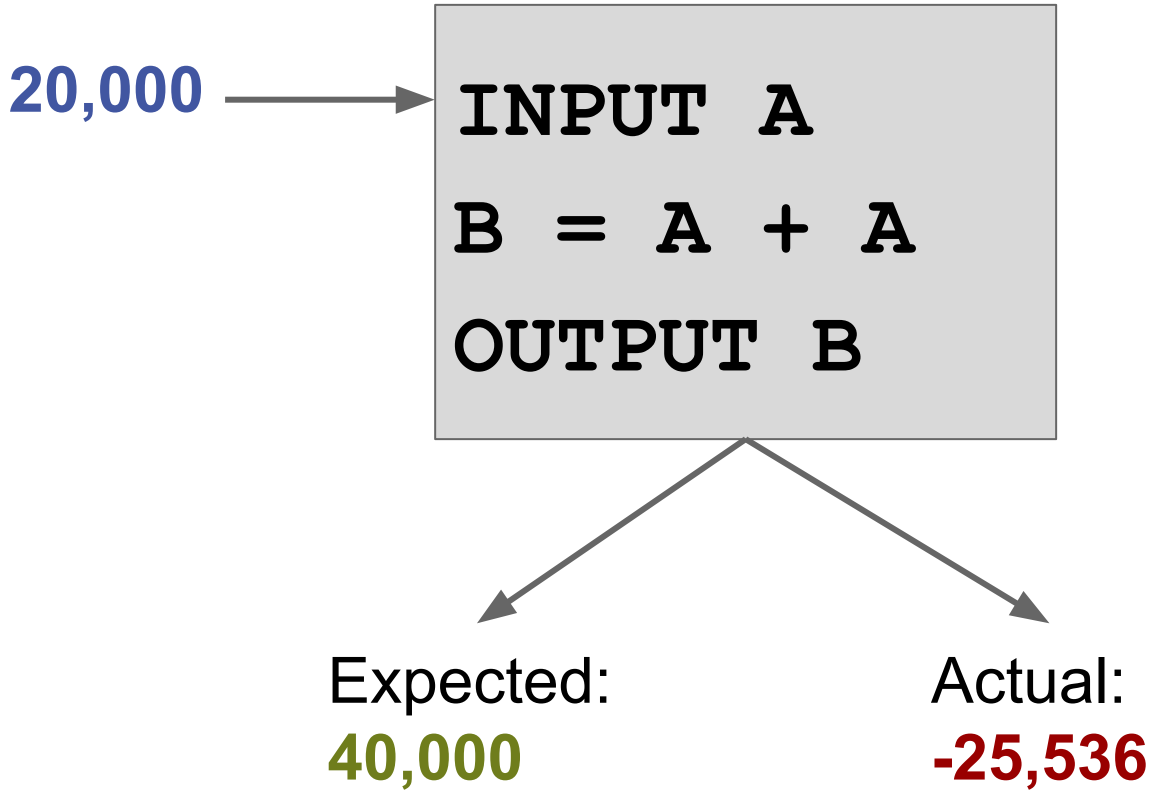
20,000

```
INPUT A
B = A + A
OUTPUT B
```

20,000

```
INPUT A
B = A + A
OUTPUT B
```

Expected:
40,000


20,000
INPUT A
B = A + A
OUTPUT B
Expected:
40,000
Actual:
-25,536

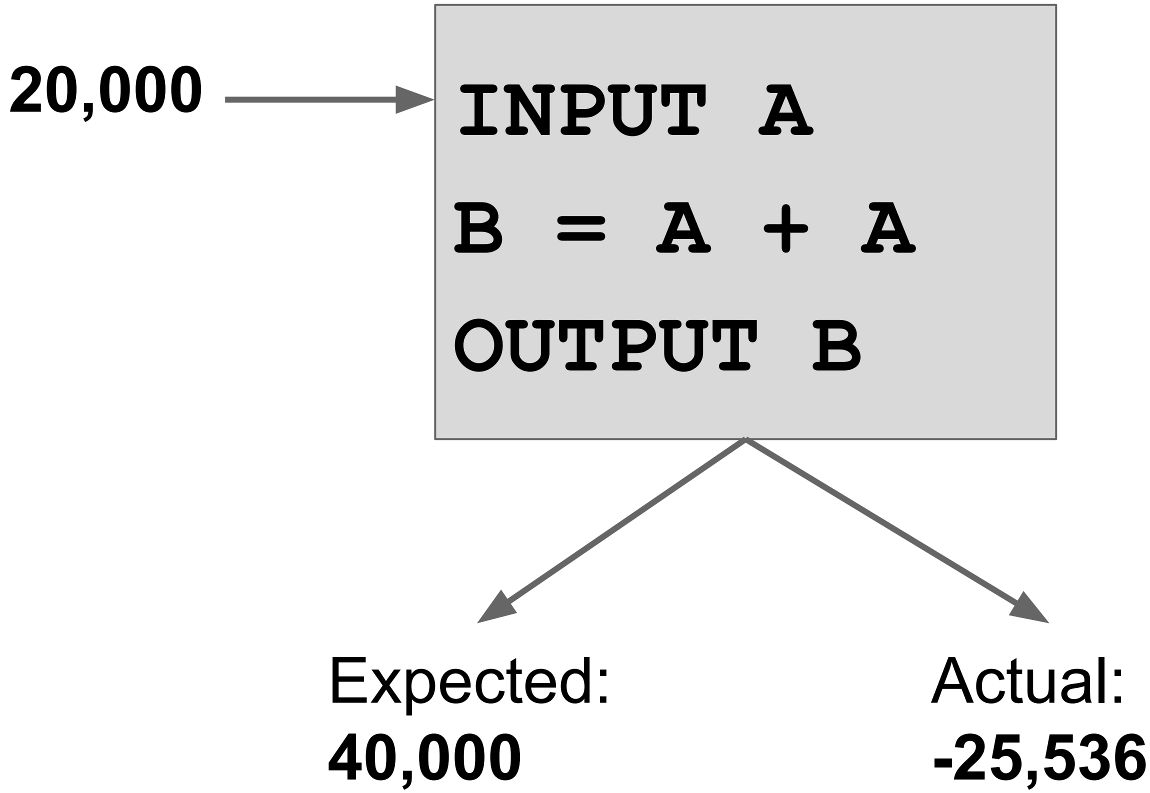
20,000

```
INPUT A
B = A + A
OUTPUT B
```

Expected:
40,000

Actual:
-25,536

Most knew result too big for data size



Most knew result too big for data size

Which is fine

20,000

```
INPUT A
B = A + A
OUTPUT B
```

Expected:
40,000

Actual:
-25,536

Most knew result too big for data size

Which is fine

BUT ...

20,000

```
INPUT A
B = A + A
OUTPUT B
```

Expected:
40,000

Actual:
-25,536

That is not the significant part of the problem!

20,000

```
INPUT A
B = A + A
OUTPUT B
```

Expected:
40,000

Actual:
-25,536

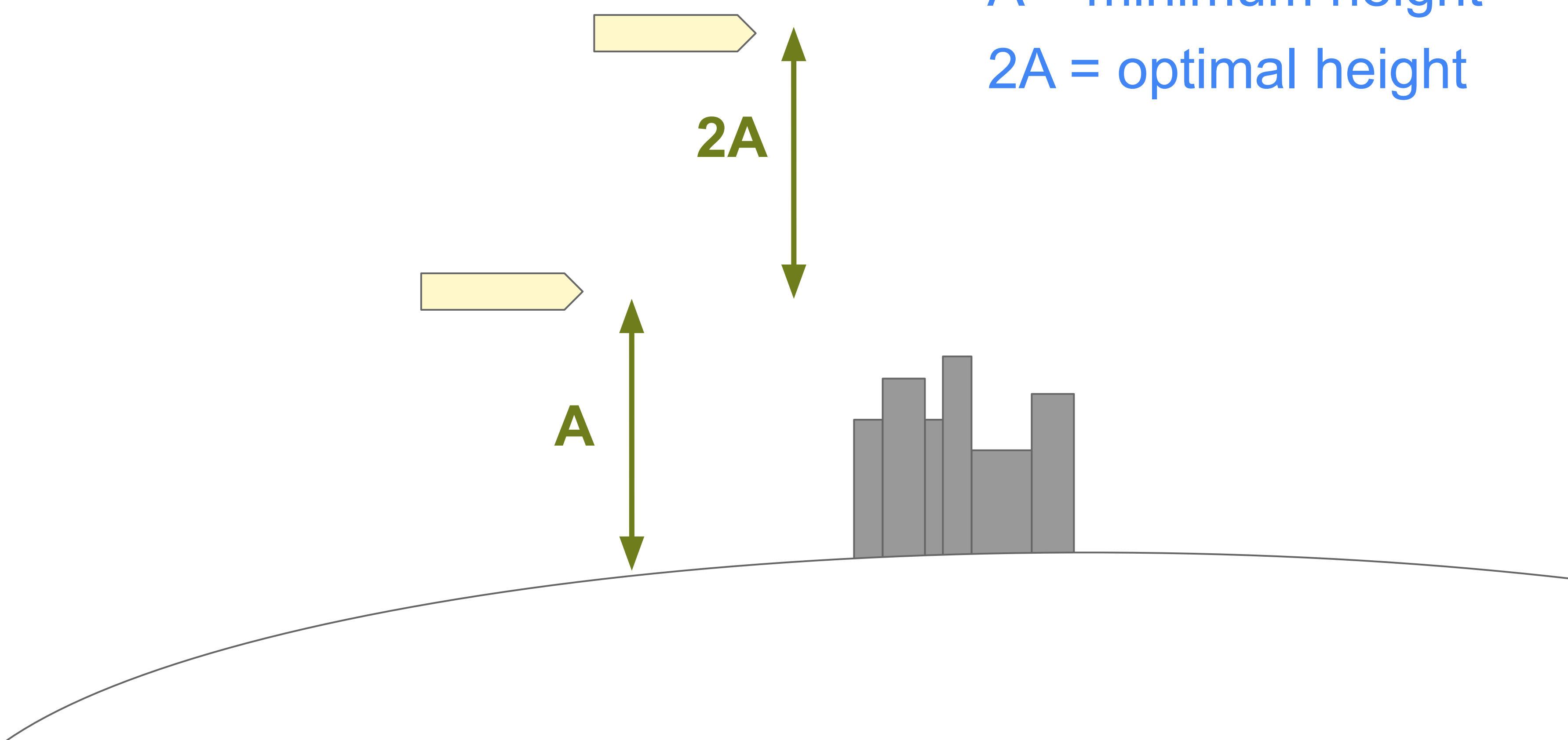
Runtime Systems did not provide a warning or error, they just output the wrong answer.

The application

Missile Control

A = minimum height

2A = optimal height



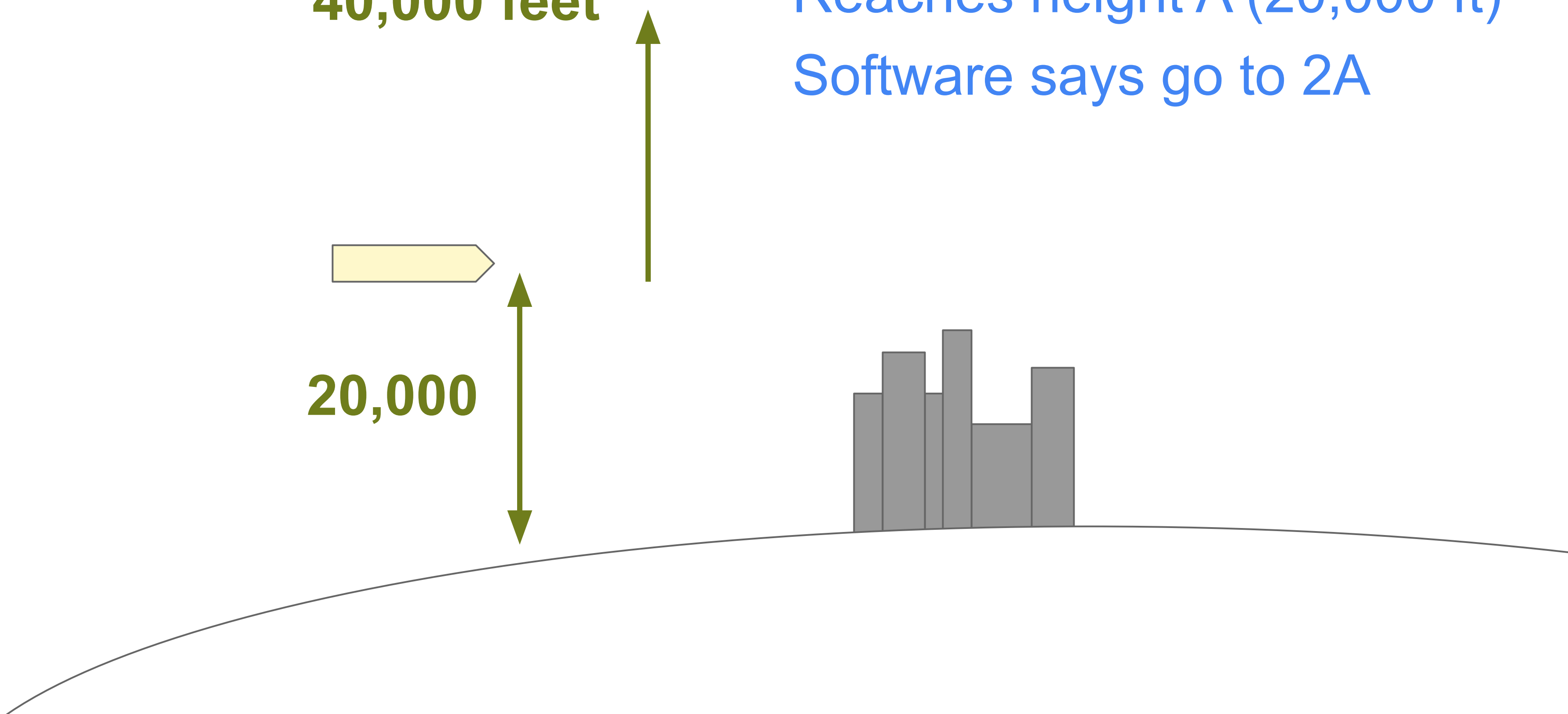
The application

Missile Control

Missile is launched.

Reaches height A (20,000 ft)

Software says go to 2A



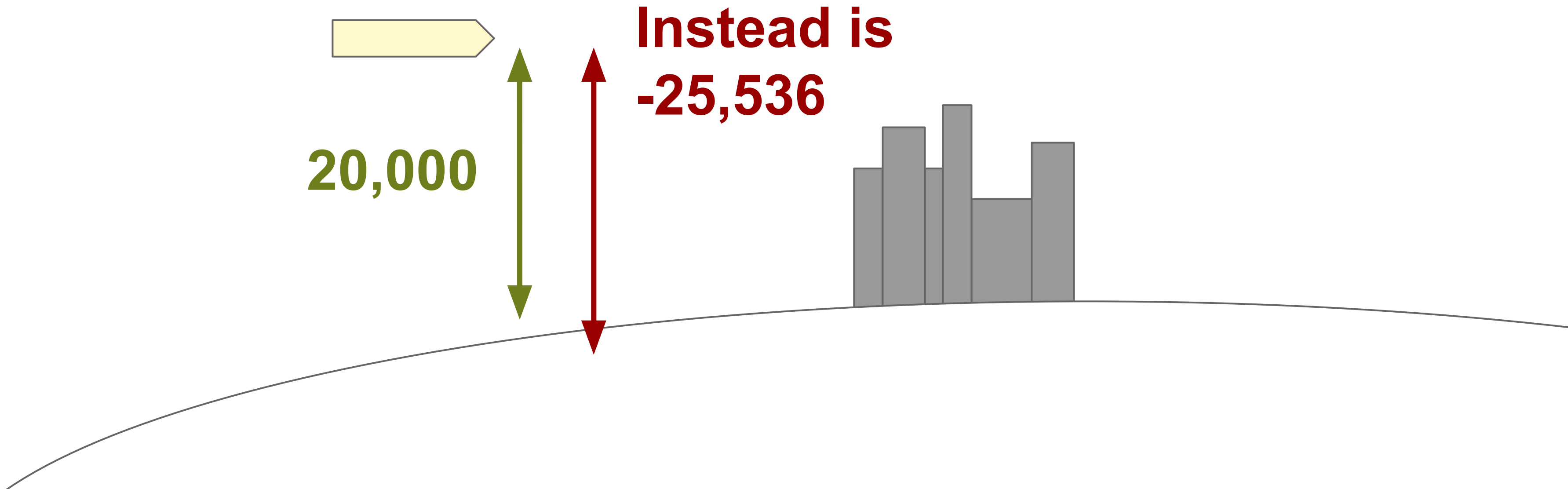
The application

Missile Control

Missile is launched.

Reaches height A (20,000 ft)

Software says go to 2A



The application

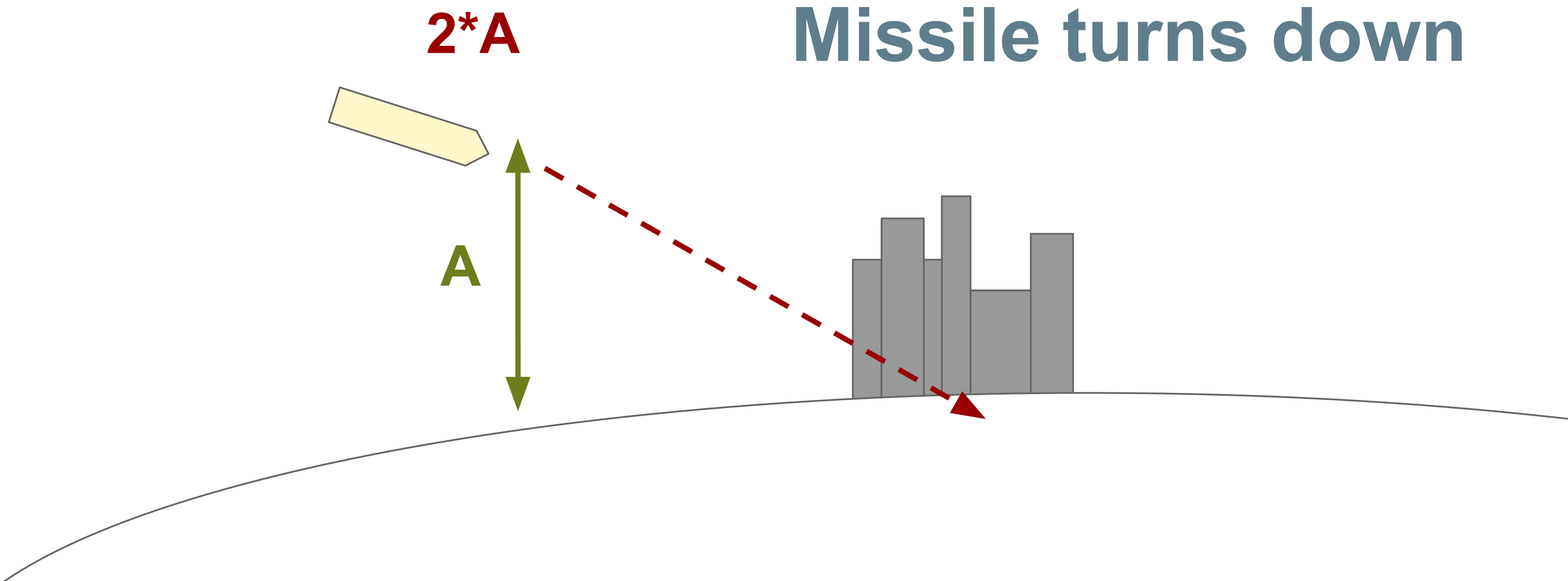
Missile Control

Missile is launched.

Reaches height A (20,000 ft)

Software says go to 2A

Missile turns down



The application

Missile Control

Missile is launched.

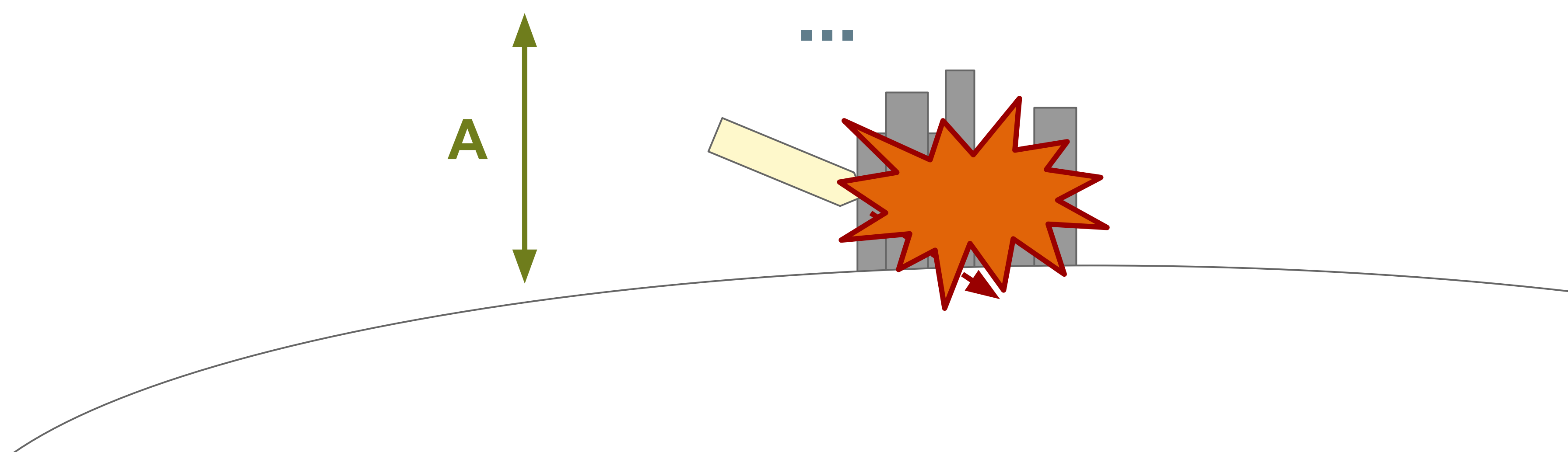
Reaches height A (20,000 ft)

Software says go to 2A

Missile turns down

...

2*A



But that sort of accident would never happen in the "Real World"

But that sort of accident would never happen in the "Real World"

… right?

But that sort of accident would never happen in the "Real World" … right?

1991 Patriot missile defense system in Dharan, Saudi Arabia fails to track and intercept an incoming Iraqi missile due to an error in the software when converting time to a 24 bit floating-point number.

28 Americans died when the missile hit Dharan Air Base

https://en.wikipedia.org/wiki/List_of_software_bugs
http://www.cs.unc.edu/~smp/COMP205/LECTURES/ERROR/lec23/node4.html

But that was a long time ago …

2015: U.S. Federal Aviation Authority requires Boeing 787 Dreamliner planes to be “rebooted” at least once every 248 days due to (likely) an integer overflow issue that could trigger loss of power to the aircraft

2016: A slot machine incorrectly prints out a winning ticket for $42,949,672.76 … on a machine where the maximum winnings are $10,000

April 20, 2020: Microsoft releases an out-of-band update for software that uses AutoDesk FBX including Microsoft Office, Paint 3D to patch multiple vulnerabilities including an integer overflow vulnerability

https://www.engadget.com/2015-05-01-boeing-787-dreamliner-software-bug.html
https://en.wikipedia.org/wiki/Integer_overflow#Examples
https://www.desmoinesregister.com/story/news/crime-and-courts/2015/04/25/supreme-court-casino-jackpot-denied/26356437/
https://www.autodesk.com/trust/security-advisories/adsk-sa-2020-0002
https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/ADV200004

Not all number systems are amenable to performing arithmetic

Without using a calculator, how would you add:

- II and I?
- III and I?
- XI and V?

Not all number systems are amenable to performing arithmetic

Without using a calculator, how would you add:

- II and I?
- III and I?
- XI and V?

II	III	XI
+ I	+ I	+ V
III	IV	XVI

Positional Number Systems

Designed to allow straightforward rules when performing arithmetic

Name	Base	Digits (0 thru base-1)

Common Set of Characteristics

Positional Number Systems

Designed to allow straightforward rules when performing arithmetic

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9

Common Set of Characteristics

Positional Number Systems

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9

Humans tend to use base 10. *Why?*

Positional Number Systems

Humans tend to use base 10. *Why?*

https://commons.wikimedia.org/wiki/File:Two_hand,_ten_fingers.jpg

Pre-Columbian Mesoamerican cultures such as the Maya used a base-20 system (using all twenty fingers and toes)

Positional Number Systems

Set of positions represented by the base raised to successively higher powers

10^2 10^1 10^0

Positional Number Systems

Set of positions represented by the base raised to successively higher powers

$$179 = (1 \times 10^2) + (7 \times 10^1) + (9 \times 10^0)$$

A number is represented as the sum of the positional value times a digit

Positional Number Systems

Set of positions represented by the base raised to successively higher powers

$$179 = (1 \times 10^2) + (7 \times 10^1) + (9 \times 10^0)$$

$$179 = 100 + 70 + 9$$

value = digit * position value

Positional Number Systems

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9

What do computers use?

Two Types of Computer

Digital
Computer

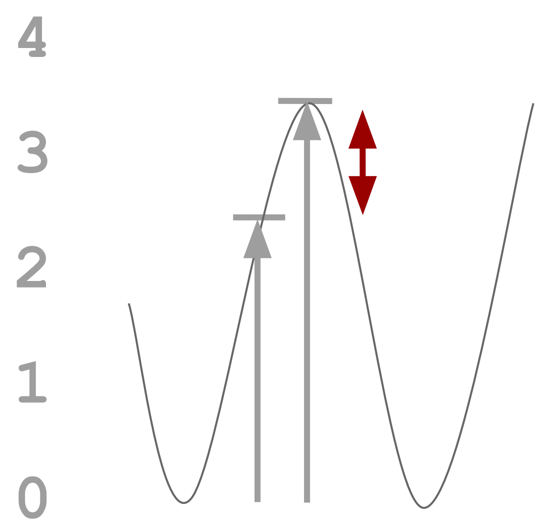
Two Types of Computer

Digital
Computer

Heathkit **Analog**
Computer circa 1956

Two Types of Computer

Analog Computer



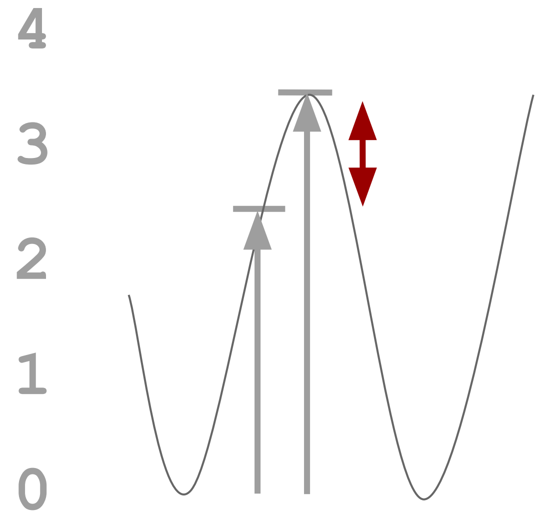
~2.5 to ~3.5 volts is 3

Uses Voltage Level to represent number

Two Types of Computer

Analog Computer

Error Prone



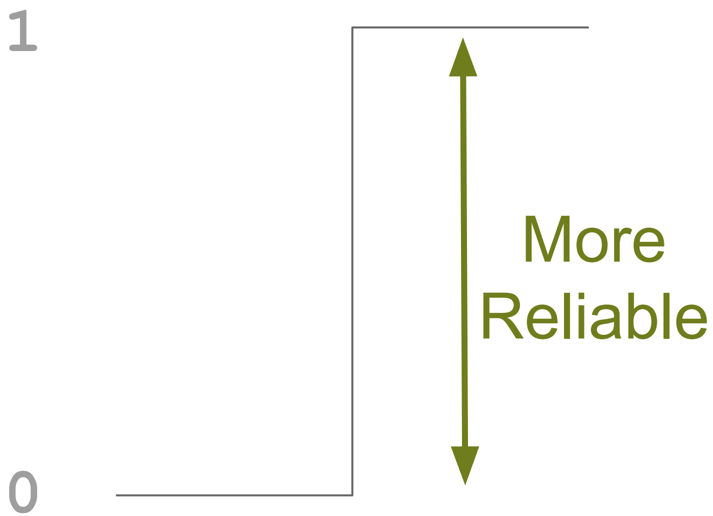
Being a little bit off gives the wrong number

~2.5 to ~3.5 volts is 3

Uses Voltage Level to represent number

Two Types of Computer

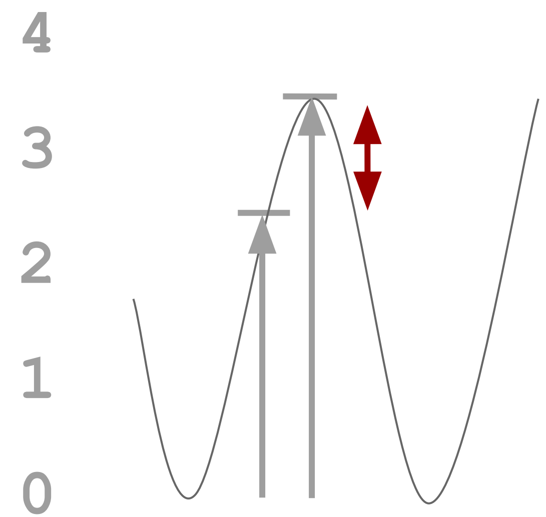
Digital Computer



Full Range

Uses Voltage On/Off to represent a number

Analog Computer

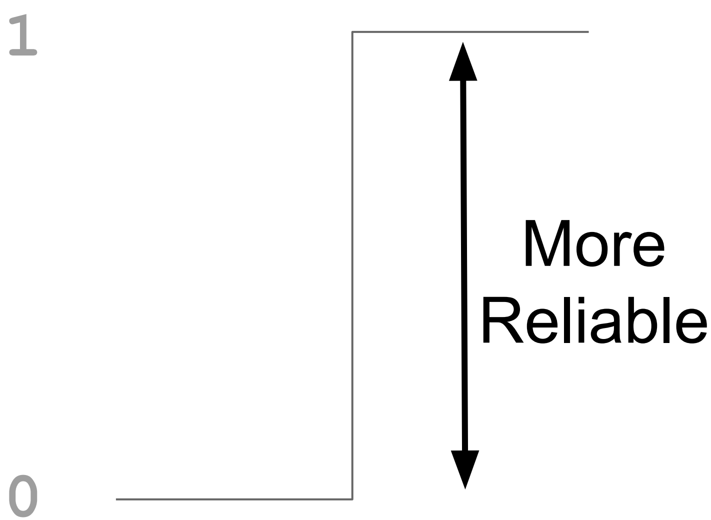


~2.5 to ~3.5 volts is 3

Uses Voltage Level to represent number

Two Types of Computer

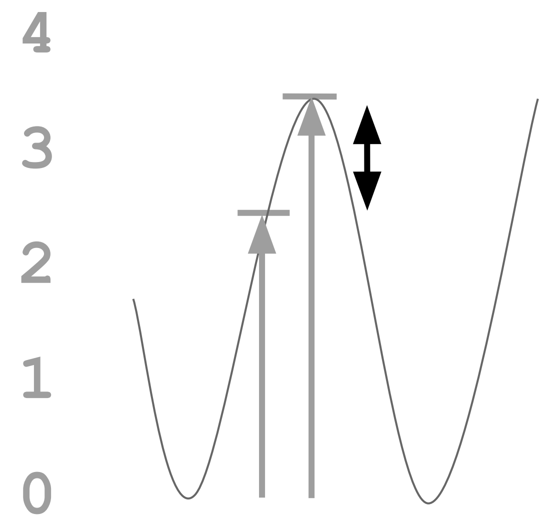
Digital Computer



Full Range

Uses Voltage On/Off to represent a number

Analog Computer



~2.5 to ~3.5 volts is 3

Uses Voltage Level to represent number

Digital Computers only need to represent two values: 0 and 1

Base 2

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9
Binary	**2**	**0-1**

Digital computers use base 2

Base 2

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9
Binary	**2**	**0-1**

$$1011_2 = (1 \times 2^3)+(0 \times 2^2)+(1 \times 2^1)+(1 \times 2^0)$$

Set of positions represented by the base raised to successively higher powers

Base 2

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9
Binary	**2**	**0-1**

$$1011_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

Set of positions represented by the base raised to successively higher powers

A number is represented as the sum of the positional value times a digit

Base 2

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9
Binary	**2**	**0-1**

$$1011_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

Do the math in decimal to get the decimal value

$$1011_2 = 8 + 0 + 2 + 1 = 11_{10}$$

Rule for adding PNS digits

If the sum of the digits is less than the base then the result is represented by a single digit

Else (the sum is equal or greater than the base) then the result is reduced by the base and represented by a single digit and a carry into the next higher position

Rule for adding PNS digits

If the sum of the digits is less than the base then the result is represented by a s

Else (the … ter than the ba… reduced by the … ed by a single digit and a carry into the next higher position

Rule for adding PNS digits

If the sum of the digits is less than the ba... nted by...

Else ... than the ... luced by ... y a sin... e next higher position

```
   11
 26577
+  952
 27529
```

Rule for adding PNS digits

If the sum of the digits is less than the base then the result is represented by a single digit

Else (the sum is equal or greater than the base) then the result is reduced by the base and represented by a single digit and a carry into the next higher position

Rule for adding PNS digits

If the sum of the digits is less than the ba… *nted by*…

Else … *than the* … *uced by* … *y a sin* … *e next higher position*

This rule is very basic for unsigned numbers

Rule for adding PNS digits

If the sum of the digits is less than the
ba… nted
by…

This rule

becomes critical

when we address signed numbers

Else (… than
the… luced
by… by a
sin… next
higher position

Rule for adding PNS digits

If the sum of the digits is less than the
ba… nted
by…

Else … than
the… uced
by… y a
sin… next
higher position

Key to the most important decision to be made by the designer of a new computer

Rule for adding PNS digits

If the sum of the digits is less than the
ba ... nted
by

Else ... than
the ... uced
by ... y a
si ... next
higher position

source of many security vulnerabilities

Decimal

```
   7
+  2
----
   9
```

```
   7
+  5
----
```

Binary

```
   0
+  0
----
```

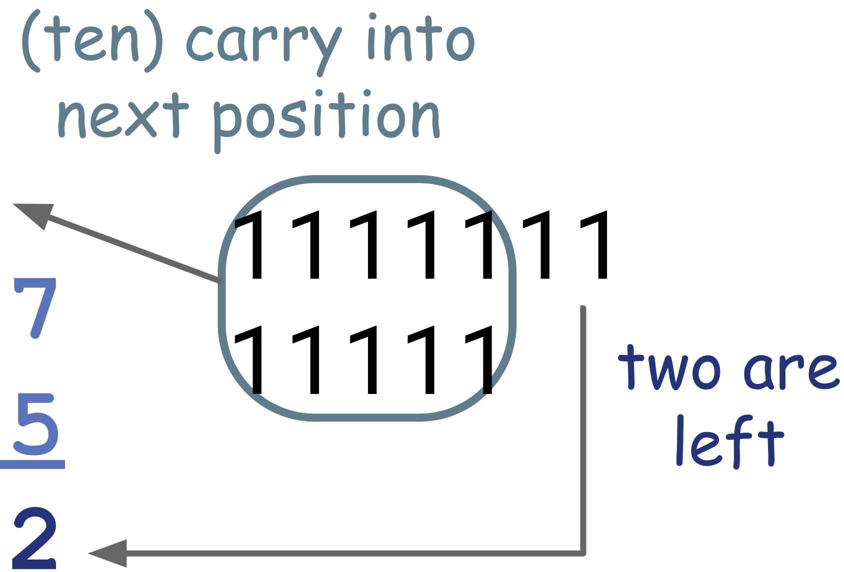
```
   0
+  1
----
```

```
   1
+  0
----
```

```
   1
+  1
----
```

Decimal

```
    7          1
+   2          7
-----      +   5
    9      -----
              12
```



Binary

```
    0         0         1         1
+   0     +   1     +   0     +   1
-----     -----     -----     -----
```

Decimal

```
   7        7
+  2     +  5
----     ----
   9       12
```

Binary

```
   0        0        1        1
+  0     +  1     +  0     +  1
----     ----     ----     ----
   0        1        1
```

Decimal

$$\begin{array}{r} 7 \\ +\ 2 \\ \hline 9 \end{array} \qquad \begin{array}{r} 7 \\ +\ 5 \\ \hline 12 \end{array}$$

Binary

carry into next position

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ 1 \\ +\ 1 \\ \hline 10 \end{array}$$

Apply the Rule to Multiple Digits

Did this in the warm-up with Base-10

```
   11
 26577
+  952
 -----
 27529
```

Apply the Rule to Multiple Digits

$$\begin{array}{rcl} 1011_2 & = & 11_{10} \\ +\ \underline{1011}_2 & = & \underline{11}_{10} \end{array}$$

Apply the Rule to Multiple Digits

$$
\begin{array}{rcl}
 & 1011_2 & = 11_{10} \\
+ & \underline{1011}_2 & = \underline{11}_{10} \\
 & & \, 22_{10}
\end{array}
$$

Apply the Rule to Multiple Digits

$$\begin{array}{rrcr} & \overset{1}{} & & \\ & 1011_2 & = & 11_{10} \\ + & \underline{1011}_2 & = & \underline{11}_{10} \\ & 0_2 & & 22_{10} \end{array}$$

Apply the Rule to Multiple Digits

$$
\begin{array}{rrcl}
 & \overset{}{11} & & \\
 & 1011_2 & = & 11_{10} \\
+ & \underline{1011}_2 & = & \underline{11}_{10} \\
 & 10_2 & & 22_{10}
\end{array}
$$

Apply the Rule to Multiple Digits

$$
\begin{array}{rrcr}
 & {\scriptstyle 11} & & \\
 & 1011_2 & = & 11_{10} \\
+ & \underline{1011}_2 & = & \underline{11}_{10} \\
 & 110_2 & & 22_{10}
\end{array}
$$

Apply the Rule to Multiple Digits

$$
\begin{array}{rcr}
\scriptstyle 1\ \ 11 & & \\
1011_2 & = & 11_{10} \\
+\ \underline{1011}_2 & = & \underline{11}_{10} \\
10110_2 & & 22_{10}
\end{array}
$$

Apply the Rule to Multiple Digits

$$\begin{array}{rcl} \scriptstyle 1\ \ 11 & & \\ 1011_2 & = & 11_{10} \\ +\ \underline{1011}_2 & = & \underline{11}_{10} \\ 10110_2 & = & 22_{10} \end{array}$$

Are these the same?

Verify our result

$$? = 10110_2$$

Verify our result

$? = 10110_2$

$? = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

Expand to true positional format

Verify our result

$? = 10110_2$

$? = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

$22_{10} = 16 + 0 + 4 + 2 + 0$

Do the math

Concepts so far

- ☑ Addition
- ☑ Positional Number Systems
- ☐ Conversion Among Bases

Conversion of Binary to Decimal

Expansion of Powers

$10110_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

$10110_2 = 16 + 4 + 2$

$10110_2 = 22_{10}$

Full Expansion

Conversion of Binary to Decimal

Expansion of Powers

$10110_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

$10110_2 = 16 + 4 + 2$

$10110_2 = 22_{10}$

or ... just write the positional values below the digits

1	0	1	1	0
16	8	4	2	1

Conversion of Binary to Decimal

Expansion of Powers

$$10110_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$

$$10110_2 = 16 + 4 + 2$$

$$10110_2 = 22_{10}$$

or ... just write the positional values below the digits and do the math

1	0	1	1	0
16	8	4	2	1

Conversion of Binary to Decimal

Expansion of Powers

$10110_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

$10110_2 = 16 + 4 + 2$

$10110_2 = 22_{10}$

or ... just write the positional values below the digits and do the math

1	0	1	1	0	
16	8	4	2	1	= 22

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(\_x2^5)+(\_x2^4)+(\_x2^3)+(\_x2^2)+(\_x2^1)+(\_x2^0)$

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(\_x2^5)+(\_x2^4)+(\_x2^3)+(\_x2^2)+(\_x2^1)+(\_x2^0)$

Need to know the
largest power of 2 in 22

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0x2^5)+(\_x2^4)+(\_x2^3)+(\_x2^2)+(\_x2^1)+(\_x2^0)$

$2^5=32$

$32>22$

So the digit is 0

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0 \times 2^5) + (1 \times 2^4) + (\_ \times 2^3) + (\_ \times 2^2) + (\_ \times 2^1) + (\_ \times 2^0)$

$2^4 = 16$ So the digit is 1

$16 < 22$

$$\begin{array}{r} 22 \\ \underline{-16} \\ 06 \end{array}$$

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0 \mathrm{x} 2^5)+(1 \mathrm{x} 2^4)+(0 \mathrm{x} 2^3)+(\_\mathrm{x} 2^2)+(\_\mathrm{x} 2^1)+(\_\mathrm{x} 2^0)$

$$\begin{array}{r} 22 \\ \underline{-16} \\ 06 \end{array}$$

$2^3=8$

$8>6$

So the digit is 0

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0x2^5)+(1x2^4)+(0x2^3)+(1x2^2)+(\_x2^1)+(\_x2^0)$

$2^2=4$

$4<6$

So the digit is 1

$$\begin{array}{r} 22 \\ \underline{-16} \\ 06 \\ \underline{-04} \\ 02 \end{array}$$

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0x2^5)+(1x2^4)+(0x2^3)+(1x2^2)+(1x2^1)+(\_x2^0)$

$2^1=2$

$2=2$

So the digit is 1

$$
\begin{array}{r}
22 \\
\underline{-16} \\
06 \\
\underline{-04} \\
02 \\
\underline{-02} \\
00
\end{array}
$$

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$

$$\begin{array}{r} 22 \\ \underline{-16} \\ 06 \\ \underline{-04} \\ 02 \\ \underline{-02} \\ 00 \end{array}$$

0 left, so any remaining digits are zero

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0x2^5)+(1x2^4)+(0x2^3)+(1x2^2)+(1x2^1)+(0x2^0)$

$$22_{10} = 010110_2$$

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

$$\underbrace{(0\text{x}2^5)+(1\text{x}2^4)+(0\text{x}2^3)+(1\text{x}2^2)+(1\text{x}2^1)+(0\text{x}2^0)}$$

$$22_{10} = 010110_2$$

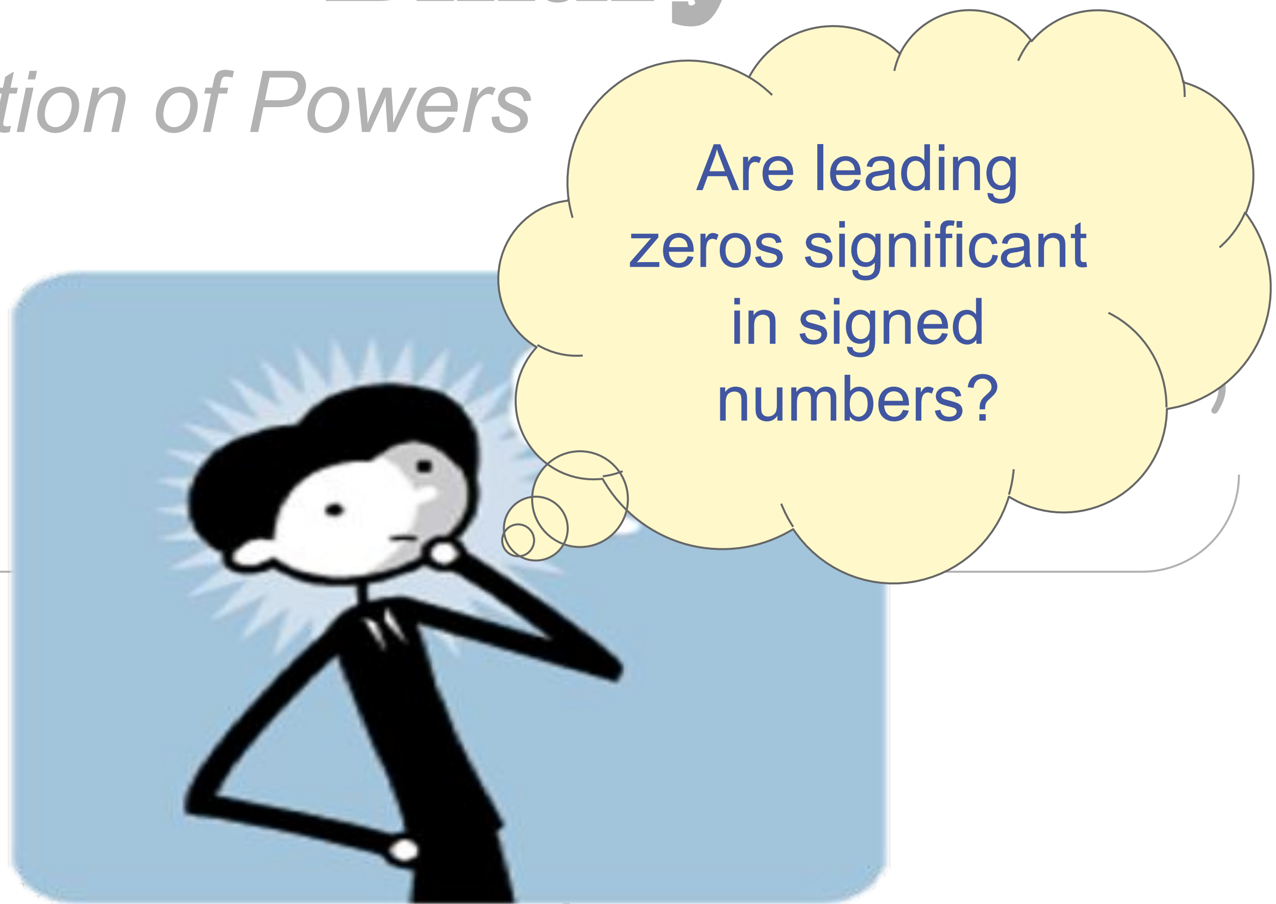
In unsigned arithmetic, leading zeros are not significant

Conversion of Decimal to Binary

Reduction of Powers

$22_{10} =$

$(0x2^5) +$



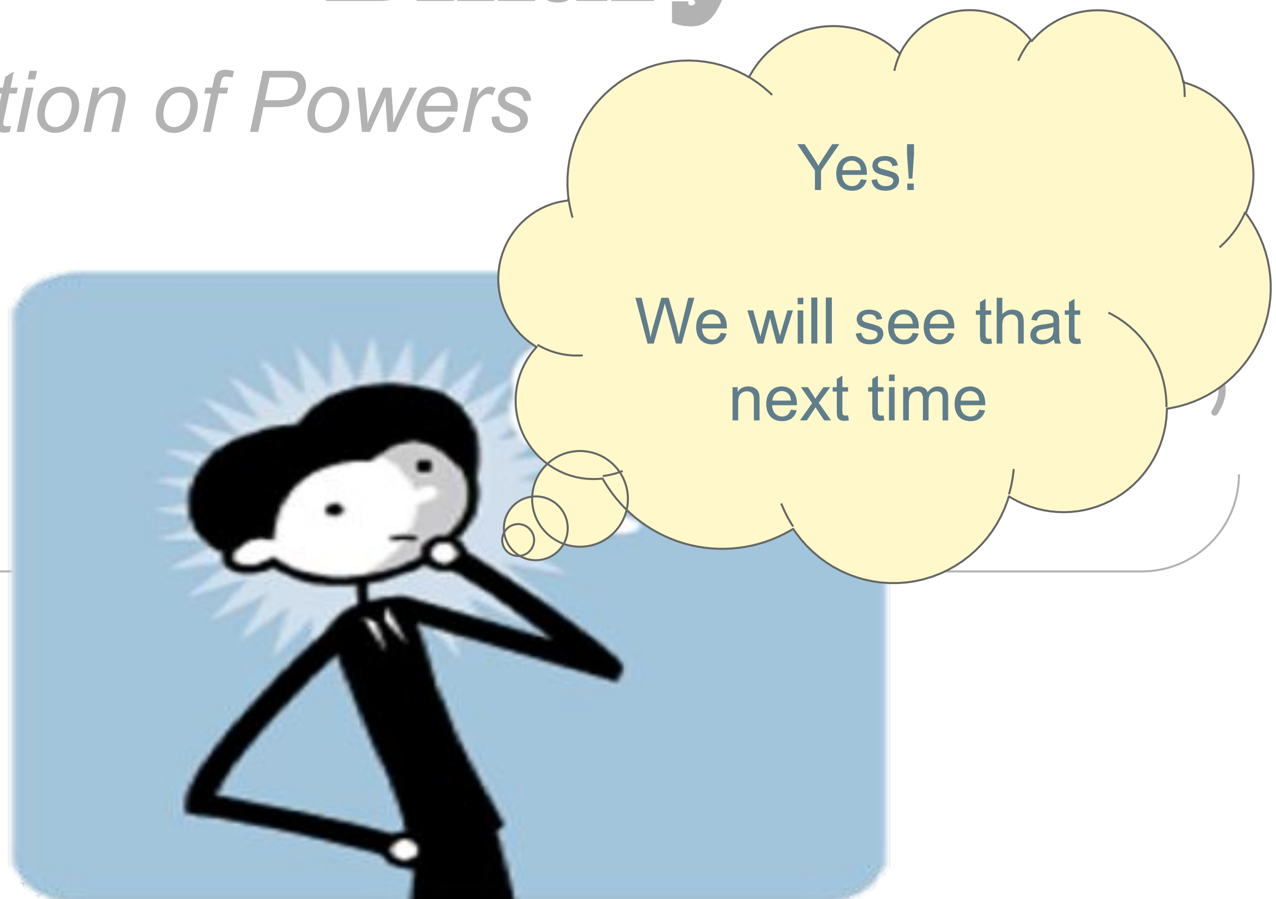
In unsigned arithmetic, leading zeros are not significant

Conversion of Decimal to Binary

Reduction of Powers

22_{10} =

$(0x2^5)+$



In unsigned arithmetic, leading zeros are not significant

Concepts so far

- ❑✓Addition
- ❑✓Positional Number Systems
- ❑✓Conversion Among Bases
- ❑ Subtraction

Subtraction

How do computers subtract?

Subtraction

How do computers subtract?

HLL Source
(e.g. Java, C++)

```
C=A-B;
```

Subtraction

Subtraction

How do computers subtract?

HLL Source
(e.g. Java, C++)

```
C=A-B;
```

Machine
Code

```
SUB  A,B,C
```

Subtraction

Subtraction

How do computers subtract?

HLL Source (e.g. Java, C++)		Machine Code		Hardware
`C=A-B;`	→	`SUB A,B,C`	→	`C=A+(-B)`

Addition *Complement Arithmetic* → `C=A+(-B)`

Subtraction

How do computers subtract?

HLL Source
(e.g. Java, C++)

```
C=A-B;
```

Machine
Code

```
SUB   A,B,C
```

Hardware

```
C=A+(-B)
```

Addition
Complement
Arithmetic

Why?

Subtraction

How do computers subtract?

HLL Source (e.g. Java, C++)		Machine Code		Hardware
`C=A-B;`	→	`SUB A,B,C`	→	`C=A+(-B)`

Addition *Complement Arithmetic*

Why?
Do not need special subtract hardware

Subtraction

How do computers subtract?

HLL Source
(e.g. Java, C++)

`C=A-B;`

Machine
Code

`SUB  A,B,C`

Hardware

`C=A+(-B)`

Addition
Complement
Arithmetic

For completeness we will cover real Subtraction

Subtraction

How do computers subtract?

HLL Source
(e.g. Java, C++)

`C=A-B;`

→

Machine
Code

`SUB  A,B,C`

→

Hardware

`C=A+(-B)`

↑

Addition
Complement
Arithmetic

Some evil person may ask you to subtract in Base 7

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

Long way
All the steps

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

Long way
All the steps

Then we will see a clever shortcut

Decimal Subtraction

```
  85
 -07
```

Re-write in true positional form

$$
\begin{array}{r}
(8\times10^1) + (5\times10^0) \\
-(0\times10^1) + (7\times10^0) \\
\hline
\end{array}
$$

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

$$\begin{array}{r} (8 \times 10^1) + (5 \times 10^0) \\ -(0 \times 10^1) + (7 \times 10^0) \\ \hline \end{array}$$

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

Since $5<7$ we must take action to proceed

$$\begin{array}{rl} & (8\times10^1) + (5\times10^0) \\ - & (0\times10^1) + (7\times10^0) \\ \hline \end{array}$$

Decimal Subtraction

We will *borrow* from the next column

85

-07

https://flickr.com/photos/67193564@N03/6173548853

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

$$\begin{array}{rl} & (8\times10^{1}) + (5\times10^{0}) \\ - & (0\times10^{1}) + (7\times10^{0}) \\ \hline \end{array}$$

Each 10^{1} equals 10×10^{0}

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

$$\begin{array}{rl} & (\cancel{8\times10}^1) + (5\times10^0) \\ - & (0\times10^1) + (7\times10^0) \\ \hline \end{array}$$

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

$$\begin{array}{rcl}
(7\times10^{1}) & + & (10\times10^{0}) \\
(\cancel{8\times10}^{1}) & + & (5\times10^{0}) \\
-(0\times10^{1}) & + & (7\times10^{0}) \\ \hline
\end{array}$$

Decimal Subtraction

$$
\begin{array}{r}
85 \\
-07 \\
\hline
\end{array}
$$

$$
\begin{array}{rcl}
(7\times10^{1}) & + & (10\times10^{0}) \\
(\cancel{8\times10}^{1}) & + & (5\times10^{0}) \\
-(0\times10^{1}) & + & (7\times10^{0}) \\
\hline
 & & (8\times10^{0})
\end{array}
$$

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

$$\begin{array}{rl} & (7\times10^{1}) + (10\times10^{0}) \\ & (\cancel{8\times10}^{1}) + (5\times10^{0}) \\ - & (0\times10^{1}) + (7\times10^{0}) \\ \hline & (7\times10^{1}) + (8\times10^{0}) \end{array}$$

Decimal Subtraction

$$\begin{array}{r} 85 \\ -07 \\ \hline \end{array}$$

$$\begin{array}{rccc} & (\ 7\times10^{1}) & + & (10\times10^{0}) \\ & (\ \cancel{8\times10}^{1}) & + & (\ 5\times10^{0}) \\ - & (\ 0\times10^{1}) & + & (\ 7\times10^{0}) \\ \hline & (\ 7\times10^{1}) & + & (\ 8\times10^{0}) \end{array}$$

78

Binary Subtraction

Binary Subtraction

$$\begin{array}{rcl} 100_2 & = & 4_{10} \\ -\ 001_2 & = & 1_{10} \\ \hline \end{array}$$

Binary Subtraction

$$
\begin{array}{rcl}
100_2 & = & 4_{10} \\
-\ 001_2 & = & 1_{10} \\
\hline
\end{array}
$$

Re-write in true positional form

$$
\begin{array}{rl}
 & (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) \\
- & (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\
\hline
\end{array}
$$

Binary Subtraction

$$
\begin{array}{rcl}
 & 100_2 & = 4_{10} \\
- & 001_2 & = 1_{10} \\
\hline
\end{array}
$$

Need to redistribute data to continue

$$
\begin{array}{rl}
 & (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) \\
- & (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\
\hline
\end{array}
$$

Binary Subtraction

$$\begin{array}{rcl} 100_2 & = & 4_{10} \\ -\ 001_2 & = & 1_{10} \\ \hline \end{array}$$

This is the first position that is not 0

$$\begin{array}{r} (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) \\ -\ (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ \hline \end{array}$$

Binary Subtraction

$$
\begin{array}{rrcl}
 & 100_2 & = & 4_{10} \\
- & 001_2 & = & 1_{10} \\
\hline
\end{array}
$$

Each 2^2 is the same as two 2^1

$$
\begin{array}{rl}
 & (0 \times 2^2) + (2 \times 2^1) \\
 & (\cancel{1 \times 2}^2) + (0 \times 2^1) + (0 \times 2^0) \\
- & (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\
\hline
\end{array}
$$

Binary Subtraction

$$\begin{array}{rcl} 100_2 & = & 4_{10} \\ -\ 001_2 & = & 1_{10} \\ \hline \end{array}$$

Each 2^2 is the same as two 2^1

$$\begin{array}{rccccc} & (0 \times 2^2) & + (10 \times 2^1) & & \\ & (\cancel{1 \times 2}^2) & + (0 \times 2^1) & + & (0 \times 2^0) \\ - & (0 \times 2^2) & + (0 \times 2^1) & + & (1 \times 2^0) \\ \hline \end{array}$$

Binary Subtraction

$$
\begin{array}{rcl}
100_2 & = & 4_{10} \\
-\ 001_2 & = & 1_{10} \\
\hline
\end{array}
$$

Each 2^2 is the same as two 2^1

$$
\begin{array}{lcccccc}
 & & & & (1 \times 2^1) & & \\
 & (0 \times 2^2) & + & & (1 \times 2^1) & & \\
 & (\cancel{1 \times 2}^2) & + & & (0 \times 2^1) & + & (0 \times 2^0) \\
- & (0 \times 2^2) & + & & (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
\end{array}
$$

Binary Subtraction

$$\begin{array}{rcl} 100_2 & = & 4_{10} \\ -\ 001_2 & = & 1_{10} \\ \hline \end{array}$$

Each 2^1 is the same as two 2^0

$$\begin{array}{rrrrr}
 & & & + & (1 \times 2^0) \\
 & & \cancel{(1 \times 2^1)} & + & (1 \times 2^0) \\
 & (0 \times 2^2) & + \ (1 \times 2^1) & & \\
 & \cancel{(1 \times 2^2)} & + \ (0 \times 2^1) & + & (0 \times 2^0) \\
- & (0 \times 2^2) & + \ (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
\end{array}$$

Binary Subtraction

$$\begin{array}{rcl} 100_2 & = & 4_{10} \\ -\ 001_2 & = & 1_{10} \\ \hline \end{array}$$

$$\begin{array}{rccccc}
 & & & & + & (1 \times 2^0) \\
 & & & \cancel{(1 \times 2^1)} & + & (1 \times 2^0) \\
 & (0 \times 2^2) & + & (1 \times 2^1) & & \\
 & \cancel{(1 \times 2^2)} & + & (0 \times 2^1) & + & (0 \times 2^0) \\
- & (0 \times 2^2) & + & (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
 & & & & & (1 \times 2^0)
\end{array}$$

Binary Subtraction

$$
\begin{array}{rcl}
 100_2 & = & 4_{10} \\
- \ 001_2 & = & 1_{10} \\
\hline
\end{array}
$$

$$
\begin{array}{rccccc}
 & & & & + & (1 \times 2^0) \\
 & & & \cancel{(1 \times 2^1)} & + & (1 \times 2^0) \\
 & (0 \times 2^2) & + & (1 \times 2^1) & & \\
 & \cancel{(1 \times 2^2)} & + & (0 \times 2^1) & + & (0 \times 2^0) \\
- & (0 \times 2^2) & + & (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
 & & & (1 \times 2^1) & + & (1 \times 2^0)
\end{array}
$$

Binary Subtraction

$$
\begin{array}{rcl}
100_2 & = & 4_{10} \\
-\ 001_2 & = & 1_{10} \\
\hline
\end{array}
$$

$$
\begin{array}{rlllll}
 & & & & + & (1 \times 2^0) \\
 & & & (\cancel{1 \times 2^1}) & + & (1 \times 2^0) \\
 & (0 \times 2^2) & + & (1 \times 2^1) & & \\
 & (\cancel{1 \times 2^2}) & + & (0 \times 2^1) & + & (0 \times 2^0) \\
- & (0 \times 2^2) & + & (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
 & (0 \times 2^2) & + & (1 \times 2^1) & + & (1 \times 2^0)
\end{array}
$$

Binary Subtraction

$$
\begin{array}{rlll}
 & 100_2 & = & 4_{10} \\
- & 001_2 & = & 1_{10} \\
\hline
 & 011_2 & &
\end{array}
$$

$$
\begin{array}{rccccc}
 & & & & + & (1 \times 2^0) \\
 & & & \cancel{(1 \times 2^1)} & + & (1 \times 2^0) \\
 & (0 \times 2^2) & + & (1 \times 2^1) & & \\
 & \cancel{(1 \times 2^2)} & + & (0 \times 2^1) & + & (0 \times 2^0) \\
- & (0 \times 2^2) & + & (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
 & (0 \times 2^2) & + & (1 \times 2^1) & + & (1 \times 2^0)
\end{array}
$$

Binary Subtraction

$$
\begin{array}{rcl}
100_2 & = & 4_{10} \\
-\ 001_2 & = & 1_{10} \\
\hline
011_2 & = & 3_{10}
\end{array}
$$

$$
\begin{array}{lcccc}
 & & & + & (1 \times 2^0) \\
 & & \cancel{(1 \times 2^1)} & + & (1 \times 2^0) \\
(0 \times 2^2) & + & (1 \times 2^1) & & \\
\cancel{(1 \times 2^2)} & + & (0 \times 2^1) & + & (0 \times 2^0) \\
-\ (0 \times 2^2) & + & (0 \times 2^1) & + & (1 \times 2^0) \\
\hline
(0 \times 2^2) & + & (1 \times 2^1) & + & (1 \times 2^0)
\end{array}
$$

Can we speed up subtraction?

Can we speed up subtraction?

1000_{10}

Re-write the `1`
followed by `0`s

Can we speed up subtraction?

$$1000_{10} = \overset{1}{0999}$$

Re-write the `1` followed by `0`s

As `0` followed by `9`s plus `1`

Can we speed up subtraction?

$$
\begin{array}{r}
1 \\
1000_{10} \;=\; 0999
\end{array}
$$

Re-write the 1 followed by 0s

As 0 followed by 9s plus 1

How does this help?

Can we speed up subtraction?

```
 1000
-0625
```

Can we speed up subtraction?

```
 1000
-0625
```

This requires multiple re-distributions of data

Can we speed up subtraction?

```
                 1
 1000   =   0999
-0625   =  -0625
```

This requires multiple re-distributions of data

Can we speed up subtraction?

```
                1
 1000   =   0999
-0625   =  -0625
```

This requires multiple re-distributions of data

This requires no re-distributions

Can we speed up subtraction?

```
                 1
 1000   =   0999
-0625   =  -0625
                 5
```

This requires multiple re-distributions of data

This requires no re-distributions

Can we speed up subtraction?

```
                  1
 1000   =   0999
-0625   =  -0625
              75
```

This requires multiple re-distributions of data

This requires no re-distributions

Can we speed up subtraction?

```
                   1
  1000   =   0999
 -0625   =  -0625
              375
```

This requires multiple re-distributions of data

This requires no re-distributions

Can we speed up subtraction?

```
                     1
  1000   =   0999
 -0625   =  -0625
             0375
```

This requires multiple re-distributions of data

This requires no re-distributions

Can we speed up subtraction?

```
                  1
 1000   =   0999
-0625   =  -0625
            0375
```

This requires multiple re-distributions of data

This requires no re-distributions

Helps avoid little (human) mistakes
(especially when working in different bases)

Can we speed up subtraction?

$$
\begin{array}{rcrrrrr}
22_{10} & = & 1 & 0 & 1 & 1 & 0_2 \\
\underline{-11}_{10} & = & \underline{-0} & \underline{1} & \underline{0} & \underline{1} & \underline{1}_2
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcccccc}
22_{10} & = & 1 & 0 & 1 & 1 & 0_2 \\
\underline{-11_{10}} & = & \underline{-0} & \underline{1} & \underline{0} & \underline{1} & \underline{1_2}
\end{array}
$$

This requires redistribution of data

Can we speed up subtraction?

$$
\begin{array}{rcrrrrr}
 & & & & & & 1 \\
 & & & & & 0 & 1 \\
22_{10} & = & 1 & 0 & 1 & \cancel{1} & \cancel{0}_2 \\
\underline{-11}_{10} & = & -0 & 1 & 0 & 1 & 1_2 \\
\hline
\end{array}
$$

Can we speed up subtraction?

$$\begin{array}{rcrrrrr} & & & & & & 1 \\ & & & & & 0 & 1 \\ 22_{10} & = & 1 & 0 & 1 & \cancel{1} & \cancel{0}_2 \\ -11_{10} & = & -0 & 1 & 0 & 1 & 1_2 \\ \hline & & & & & & 1 \end{array}$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrr}
 & & & & & 1 & \\
 & & & & & 1 & 1 \\
 & & & & 0 & \cancel{0} & 1 \\
22_{10} & = & 1 & 0 & \cancel{1} & \cancel{1} & \cancel{0}_2 \\
\underline{-11}_{10} & = & \underline{-0} & \underline{1} & \underline{0} & \underline{1} & \underline{1}_2 \\
 & & & & & & 1
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrr}
 & & & & & 1 & \\
 & & & & & 1 & 1 \\
 & & & & 0 & \cancel{0} & 1 \\
22_{10} & = & 1 & 0 & \cancel{1} & \cancel{1} & \cancel{0}_2 \\
\underline{-11}_{10} & = & -0 & 1 & 0 & 1 & 1_2 \\
\hline
 & & & & & 1 & 1
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrr}
 & & & & & 1 & \\
 & & & & & 1 & 1 \\
 & & & & 0 & \cancel{0} & 1 \\
22_{10} & = & 1 & 0 & \cancel{1} & \cancel{1} & \cancel{0}_2 \\
\underline{-11_{10}} & = & \underline{-0} & \underline{1} & \underline{0} & \underline{1} & \underline{1}_2 \\
 & & & & 0 & 1 & 1
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrr}
 & & & & & 1 & \\
 & & & 1 & & 1 & 1 \\
 & & 0 & 1 & 0 & \cancel{0} & 1 \\
22_{10} & = & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0}_2 \\
-11_{10} & = & -0 & 1 & 0 & 1 & 1_2 \\
\hline
 & & & & 0 & 1 & 1
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrrr}
 & & & & & & 1 & \\
 & & & 1 & & 1 & 1 & \\
 & & 0 & 1 & 0 & \cancel{0} & 1 & \\
22_{10} & = & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & _2 \\
-11_{10} & = & -0 & 1 & 0 & 1 & 1 & _2 \\
\hline
 & & & 1 & 0 & 1 & 1 &
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrrl}
 & & & & & 1 & & \\
 & & & 1 & & 1 & 1 & \\
 & & 0 & 1 & 0 & \cancel{0} & 1 & \\
22_{10} & = & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & _2 \\
-11_{10} & = & -0 & 1 & 0 & 1 & 1 & _2 \\
\hline
 & & 0 & 1 & 0 & 1 & 1 &
\end{array}
$$

Can we speed up subtraction?

$$
\begin{array}{rcrrrrrl}
 & & & & & 1 & & \\
 & & & 1 & & 1 & 1 & \\
 & & 0 & 1 & 0 & \cancel{0} & 1 & \\
22_{10} & = & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0} & _2 \\
\underline{-11}_{10} & = & \underline{-0} & \underline{1} & \underline{0} & \underline{1} & \underline{1} & _2 \\
11_{10} & = & 0 & 1 & 0 & 1 & 1 & _2
\end{array}
$$

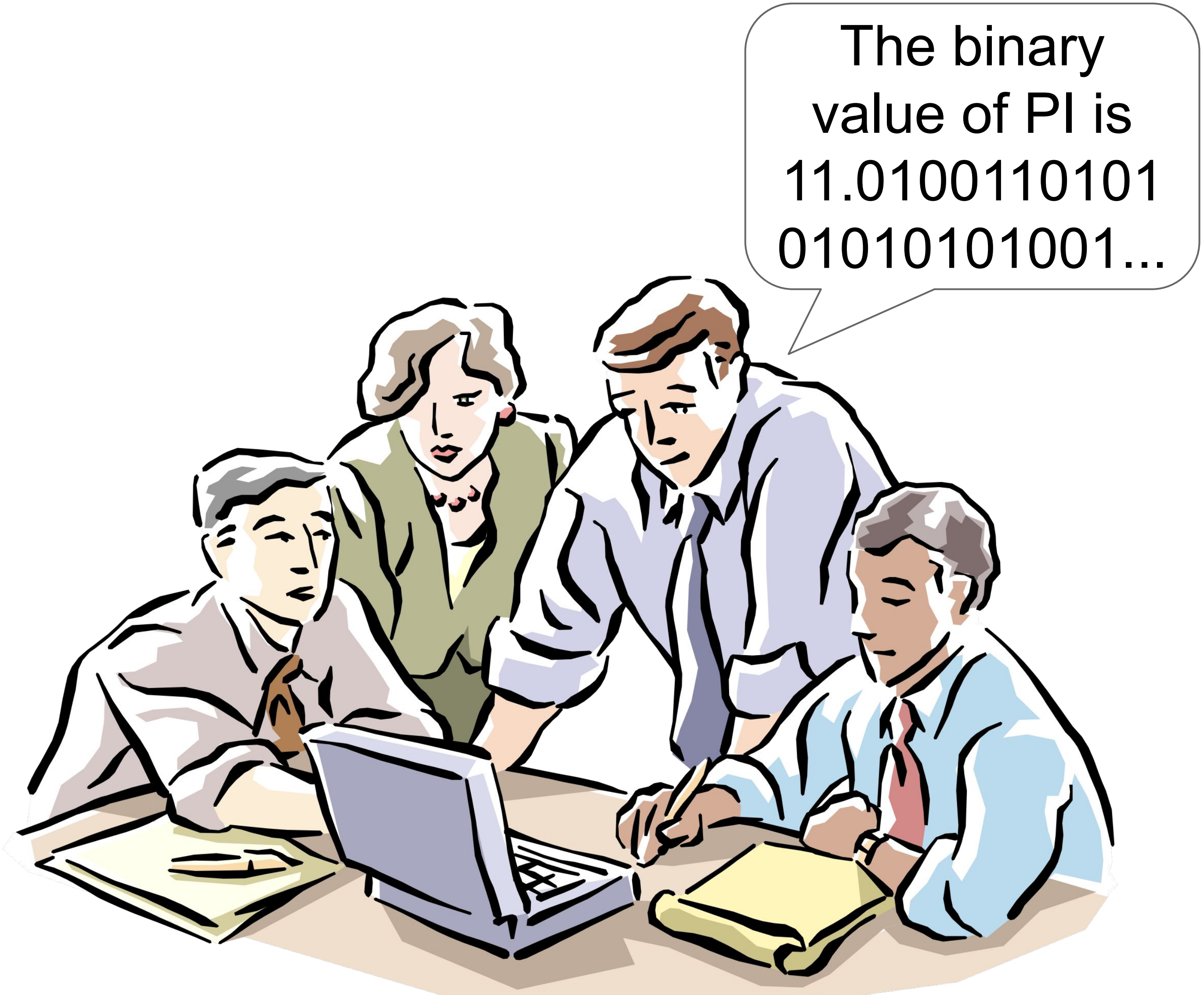
Can we speed up subtraction?

$$
\begin{array}{rcccccc}
 & & & & & 1 & \\
 & & & 1 & & 1 & 1 \\
 & & 0 & 1 & 0 & \cancel{0} & 1 \\
22_{10} & = & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{0}_2 \\
\underline{-11}_{10} & = & \underline{-0} & \underline{1} & \underline{0} & \underline{1} & \underline{1}_2 \\
11_{10} & = & 0 & 1 & 0 & 1 & 1_2
\end{array}
$$

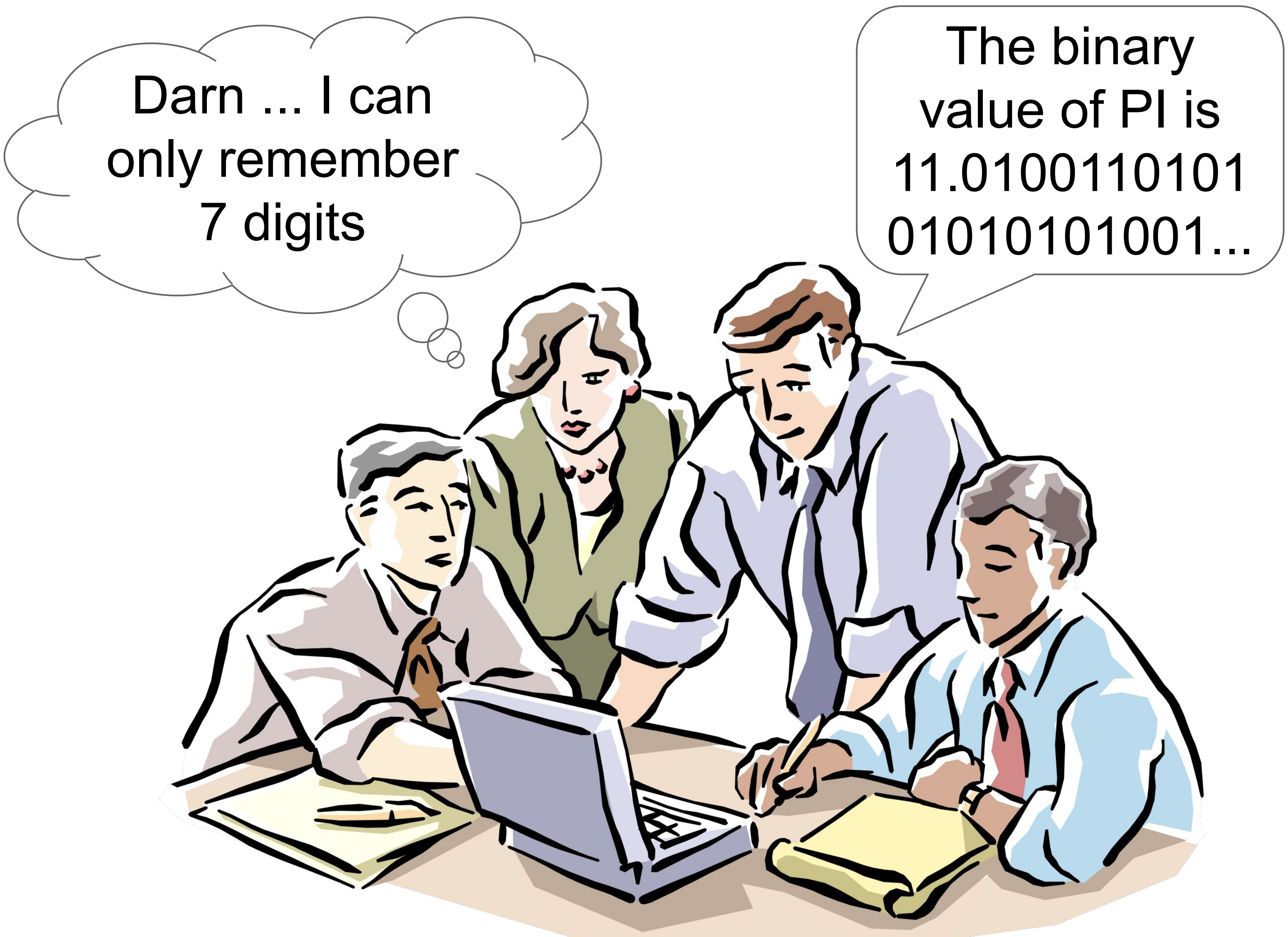
Concepts so far

- ☑ Addition
- ☑ Positional Number Systems
- ☑ Conversion Among Bases
- ☑ Subtraction

Long binary numbers are hard for ***humans*** to remember



Long binary numbers are hard for ***humans*** to remember



Shortcut to working in binary

$1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1_2$

Shortcut to working in binary

$$\underline{1\ 0\ 1\ 1}\ \ \underline{0\ 1\ 1\ 0}\ \ \underline{1\ 1\ 0\ 0}\ \ \underline{0\ 1\ 1\ 1}_2$$

Break a binary number into groups of 4 binary digits

Shortcut to working in binary

$\underline{1\ 0\ 1\ 1}\ \underline{0\ 1\ 1\ 0}\ \underline{1\ 1\ 0\ 0}\ \underline{0\ 1\ 1\ 1}_2$

$0\ 0\ 0\ 0_2$ $\quad$ $1\ 1\ 1\ 1_2$

0_{10} $\quad$ 15_{10}

Digits run 0 to (base - 1)

Digits 0 - 15 lead to a base of 16

Concepts so far

- ☑ Addition
- ☑ Positional Number Systems
- ☑ Conversion Among Bases
- ☑ Subtraction
- ❑ Hexadecimal

Hexadecimal - base 16

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9
Binary	2	0-1
Hexadecimal	16	0-(15)

Proper term would be senidenary

Hexadecimal - base 16

Name	Base	Digits (0 thru base-1)
Decimal	10	0-9
Binary	2	0-1
Hexadecimal (Hex)	16	0-(15)

we need something for digits above 9

Decimal	Binary	Hexadecimal
	0000	
	0001	
	0010	
	0011	
	0100	
	0101	
	0110	
	0111	
	1000	
	1001	
	1010	
	1011	
	1100	
	1101	
	1110	
	1111	

Decimal	Binary	Hexadecimal
00	0000	
01	0001	
02	0010	
03	0011	
04	0100	
05	0101	
06	0110	
07	0111	
08	1000	
09	1001	
10	1010	
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

Decimal	Binary	Hexadecimal
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

Decimal	Binary	Hexadecimal
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hex ↔ Binary

… lookup in table

$1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0_2$

Decimal	Binary	Hexadecimal
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hex ↔ Binary

… lookup in table

$1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0_2$

$\updownarrow$

C

Decimal	Binary	Hexadecimal
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hex ↔ Binary

… lookup in table

$1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0_2$

C 5

Decimal	Binary	Hexadecimal
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hex ↔ Binary

… lookup in table

$1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0_2$

C 5 E

Hex → Decimal

Expansion of Powers to get Decimal

$$A\ 7_{16} = (\ A \times 16^{1}) + (\ 7 \times 16^{0})$$

Hex → Decimal

Expansion of Powers to get Decimal

$$A\,7_{16} = (A \times 16^1) + (7 \times 16^0)$$

$$A\,7_{16} = (10 \times 16^1) + (7 \times 16^0)$$

Do math in decimal when converting to decimal

Hex → Decimal

Expansion of Powers to get Decimal

$$A\ 7_{16} = (\ A \times 16^{1}) + (\ 7 \times 16^{0})$$

$$A\ 7_{16} = (10 \times 16^{1}) + (\ 7 \times 16^{0})$$

$$A\ 7_{16} = 160 + 7$$

Do math in decimal when converting to decimal

Hex → Decimal

Expansion of Powers to get Decimal

$$A\ 7_{16} = (\ Ax16^{1}) + (\ 7x16^{0})$$

$$A\ 7_{16} = (10x16^{1}) + (\ 7x16^{0})$$

$$A\ 7_{16} = 160 + 7$$

$$A\ 7_{16} = 167_{10}$$

Do math in decimal when converting to decimal

Decimal → Hex

Reduction of Powers to get Hexadecimal

$$167_{10} = (\_\_\_ \times 16^2) + (\_\_\_ \times 16^1) + (\_\_\_ \times 16^0)$$

Decimal → Hex

Reduction of Powers to get Hexadecimal

$$167_{10} = (\ 0 \times 16^2) + (\_\_ \times 16^1) + (\_\_ \times 16^0)$$

So Digit is 0

$16^2 = 256$

$256 > 167$

Decimal → Hex

Reduction of Powers to get Hexadecimal

$$167_{10}=(\ 0\text{x}16^2) + (\ A\text{x}16^1) + (\_\_\text{x}16^0)$$

$16^1=16$

$10\text{x}16=160$

Ten 16 in 167

$11\text{x}16=176$

So digit is "A"

$160<167<176$

We have 167-160=7 left

Decimal → Hex

Reduction of Powers to get Hexadecimal

$$167_{10} = (\ 0 \times 16^2) + (\ A \times 16^1) + (\ 7 \times 16^0)$$

$16^0 = 1$ Seven 1 in 7

$7 \times 1 = 7$ So digit is "7"

Decimal → Hex

Reduction of Powers to get Hexadecimal

$$167_{10} = (\ 0 \times 16^2) + (\ A \times 16^1) + (\ 7 \times 16^0)$$

$$167_{10} = A7_{16}$$

Conversion Alternatives

- Reduction of Powers works from high digit to low.
- It's also easy to work from low digit to high.
  - We saw how to do this is C and Software Tools:

```
Given some value v
While v is non-zero
    Next low-order hex digit is v % 16
    Let v = v / 16 (shift all hex digits to the right)
```

Addition in Hex

$$\begin{array}{r} 7\;A_{16} \\ +\;5\;E_{16} \\ \hline \end{array}$$

Addition in Hex

Work col by col
- convert to decimal
- perform the operation
- convert result to hex

$$\begin{array}{rcc} & 7 & A_{16} \\ + & 5 & E_{16} \\ \hline \end{array}$$

Addition in Hex

$$\begin{array}{r} 7\ A_{16} \\ +\ 5\ E_{16} \\ \hline \end{array}$$

Addition in Hex

$$
\begin{array}{rcc}
 & 7 & A_{16} \\
+ & 5 & \underline{E}_{16} \\
\hline
\end{array}
$$

Addition in Hex

$$\begin{array}{r} A_{16} \\ +\ \underline{E}_{16} \end{array}$$

$$\begin{array}{r} 7\ A_{16} \\ +\ \underline{5\ E}_{16} \end{array}$$

Addition in Hex

$$
\begin{array}{rcl}
A_{16} & = & 10_{10} \\
+\ \underline{E}_{16} & = & \underline{14}_{10}
\end{array}
$$

$$
\begin{array}{r}
7\ A_{16} \\
+\ \underline{5\ E}_{16}
\end{array}
$$

Addition in Hex

$$A_{16} = 10_{10}$$
$$+\ \underline{E_{16}} = \underline{14}_{10}$$
$$24$$

$$\begin{array}{rcc} & 7 & A_{16} \\ + & 5 & E_{16} \\ \hline \end{array}$$

Addition in Hex

$$\begin{array}{rcl} A_{16} & = & 10_{10} \\ + \underline{E}_{16} & = & \underline{14}_{10} \\ & & 24 = \end{array}$$

Greater than
the base (16)

$$\begin{array}{r} 7\ A_{16} \\ +\underline{\ 5\ E}_{16} \end{array}$$

Addition in Hex

$$\begin{array}{rcl} & A_{16} & = 10_{10} \\ + & \underline{E_{16}} & = \underline{14_{10}} \\ & & 24 = 16 + 8 \end{array}$$

Greater than
the base (16)

$$\begin{array}{rcc} & 7 & A_{16} \\ + & \underline{5} & \underline{E_{16}} \end{array}$$

Addition in Hex

$$
\begin{array}{rcl}
A_{16} & = & 10_{10} \\
+\ \underline{E}_{16} & = & \underline{14}_{10} \\
24 & = & 16 + 8
\end{array}
$$

Greater than the base (16)

$$
\begin{array}{rrr}
 & 1 & \\
 & 7 & A_{16} \\
+ & \underline{5} & \underline{E}_{16} \\
 & & 8
\end{array}
$$

Addition in Hex

$$
\begin{array}{rcc}
 & 1 & \\
 & 7 & A_{16} \\
+ & 5 & E_{16} \\
\hline
 & & 8
\end{array}
$$

Addition in Hex

$$\begin{array}{rl} & 1 \\ & 7_{16} \\ + & \underline{5}_{16} \end{array}$$

$$\begin{array}{rcc} & 1 & \\ & 7 & A_{16} \\ + & \underline{5} & \underline{E}_{16} \\ & & 8 \end{array}$$

Addition in Hex

$$
\begin{array}{rcl}
1 & & 1 \\
7_{16} & = & 7_{10} \\
+\ \underline{5}_{16} & = & \underline{5}_{10}
\end{array}
$$

$$
\begin{array}{rcc}
 & 1 & \\
 & 7 & A_{16} \\
+ & \underline{5} & \underline{E}_{16} \\
 & & 8
\end{array}
$$

Addition in Hex

$$
\begin{array}{rcl}
1 & & 1 \\
7_{16} & = & 7_{10} \\
+\ \underline{5}_{16} & = & \underline{5}_{10} \\
 & & 13
\end{array}
\qquad\qquad
\begin{array}{rcc}
 & 1 & \\
 & 7 & A_{16} \\
+ & \underline{5} & \underline{E}_{16} \\
 & & 8
\end{array}
$$

Addition in Hex

$$\begin{array}{rcl} & 1 & & 1 \\ & 7_{16} & = & 7_{10} \\ + & \underline{5}_{16} & = & \underline{5}_{10} \\ & & & 13 \end{array}$$

Less than
the base (16)

$$\begin{array}{rrr} & 1 & \\ & 7 & A_{16} \\ + & 5 & E_{16} \\ \hline & & 8 \end{array}$$

Addition in Hex

$$\begin{array}{rcl} 1 & & 1 \\ 7_{16} & = & 7_{10} \\ +\ \underline{5}_{16} & = & \underline{5}_{10} \\ & & 13 \end{array}$$

$$\begin{array}{rr} 1 & \\ 7 & A_{16} \\ +\ \underline{5} & \underline{E}_{16} \\ D & 8 \end{array}$$

Less than
the base (16)

Addition in Hex

$$
\begin{array}{rcc}
 & 1 & \\
 & 7 & A_{16} \\
+ & 5 & E_{16} \\
\hline
 & D & 8_{16}
\end{array}
$$

Subtraction in Hex

$$\begin{array}{r} D\ 8_{16} \\ -\ \underline{5\ E}_{16} \end{array}$$

Subtraction in Hex

$$\begin{array}{rcl} 8_{16} & = & 8_{10} \\ -\ \underline{E}_{16} & = & \underline{14}_{10} \end{array}$$

$$\begin{array}{rcc} & D & 8_{16} \\ - & \underline{5} & \underline{E}_{16} \end{array}$$

Subtraction in Hex

$$8_{16} = 8_{10}$$

$$- \underline{E}_{16} = \underline{14}_{10}$$

Requires redistribution

$$\begin{array}{r} D\ 8_{16} \\ - \underline{5\ E}_{16} \end{array}$$

Subtraction in Hex

$$
\begin{array}{rcc}
 & C & 10 \\
 & \cancel{D} & 8_{16} \\
- & \underline{5} & \underline{E_{16}}
\end{array}
$$

Each $16^1=16*16^0$

Subtraction in Hex

$$
\begin{array}{rr}
 & 10_{16} \\
+ & 8_{16} \\
- & \underline{E}_{16}
\end{array}
$$

$$
\begin{array}{rcc}
 & C & 10 \\
 & \cancel{D} & 8_{16} \\
- & \underline{5} & \underline{E}_{16}
\end{array}
$$

Subtraction in Hex

$$
\begin{array}{rrcrr}
 & 10_{16} & & & 16_{10} \\
+ & 8_{16} & = & + & 8_{10} \\
- & \underline{E}_{16} & = & - & \underline{14}_{10}
\end{array}
$$

$$
\begin{array}{rcc}
 & C & 10 \\
 & \cancel{D} & 8_{16} \\
- & \underline{5} & \underline{E}_{16}
\end{array}
$$

Subtraction in Hex

$$\begin{array}{rrcrr} & 10_{16} & & & 16_{10} \\ + & 8_{16} & = & + & 8_{10} \\ - & \underline{E}_{16} & = & - & \underline{14}_{10} \\ & & & & 10 \end{array}$$

$$\begin{array}{rcc} & C & 10 \\ & \cancel{D} & 8_{16} \\ - & 5 & E_{16} \\ \hline & & A \end{array}$$

Subtraction in Hex

$$
\begin{array}{rcc}
 & C & 10 \\
 & \cancel{D} & 8_{16} \\
- & 5 & E_{16} \\
\hline
 & & A
\end{array}
$$

Subtraction in Hex

$$\begin{array}{r} C_{16} \\ -\ \underline{5}_{16} \end{array}$$

$$\begin{array}{rr} C & 10 \\ \cancel{D} & 8_{16} \\ -\ \underline{5} & \underline{E}_{16} \\ & A \end{array}$$

Subtraction in Hex

$$\begin{array}{rcr} C_{16} & = & 12_{10} \\ -\ \underline{\quad 5}_{16} & = & -\ \underline{\quad 5}_{10} \end{array}$$

$$\begin{array}{rrr} & C & 10 \\ & \cancel{D} & 8_{16} \\ - & 5 & E_{16} \\ \hline & & A \end{array}$$

Subtraction in Hex

$$\begin{array}{rcr} C_{16} & = & 12_{10} \\ -\ \underline{\ 5}_{16} & = & -\ \underline{\ 5}_{10} \\ & & 7_{10} \end{array}$$

$$\begin{array}{rr} C & 10 \\ \cancel{D} & 8_{16} \\ -\ 5 & E_{16} \\ \hline 7 & A \end{array}$$

Subtraction in Hex

$$
\begin{array}{rcc}
 & D & 8_{16} \\
- & 5 & E_{16} \\
\hline
 & 7 & A
\end{array}
$$

Concepts so far

- [x] Addition
- [x] Positional Number Systems
- [x] Conversion Among Bases
- [x] Subtraction
- [x] Hexadecimal
- [ ] Overflow

Overflow

$$\begin{array}{r} 9 \\ + \ 2 \\ \hline \end{array}$$

Working with pencil and paper,
dry-erase markers and whiteboard,
etc.

Create Digits as needed

$$\begin{array}{r} 1 \\ 9 \\ +\ 2 \\ \hline 1 \end{array}$$

Working with pencil and paper,
dry-erase markers and whiteboard,
etc.

Create Digits as needed

$$\begin{array}{r} 1 \\ 9 \\ +\ \ 2 \\ \hline 11 \end{array}$$

Working with pencil and paper,
dry-erase markers and whiteboard,
etc.

Infinite Precision Arithmetic

Create Digits as needed

```
  1
   9
+  2
----
  11
```

Working with pencil and paper,
dry-erase markers and whiteboard,
etc.

Once you build hardware

Digits are fixed by the Manufacturer

Once you build hardware

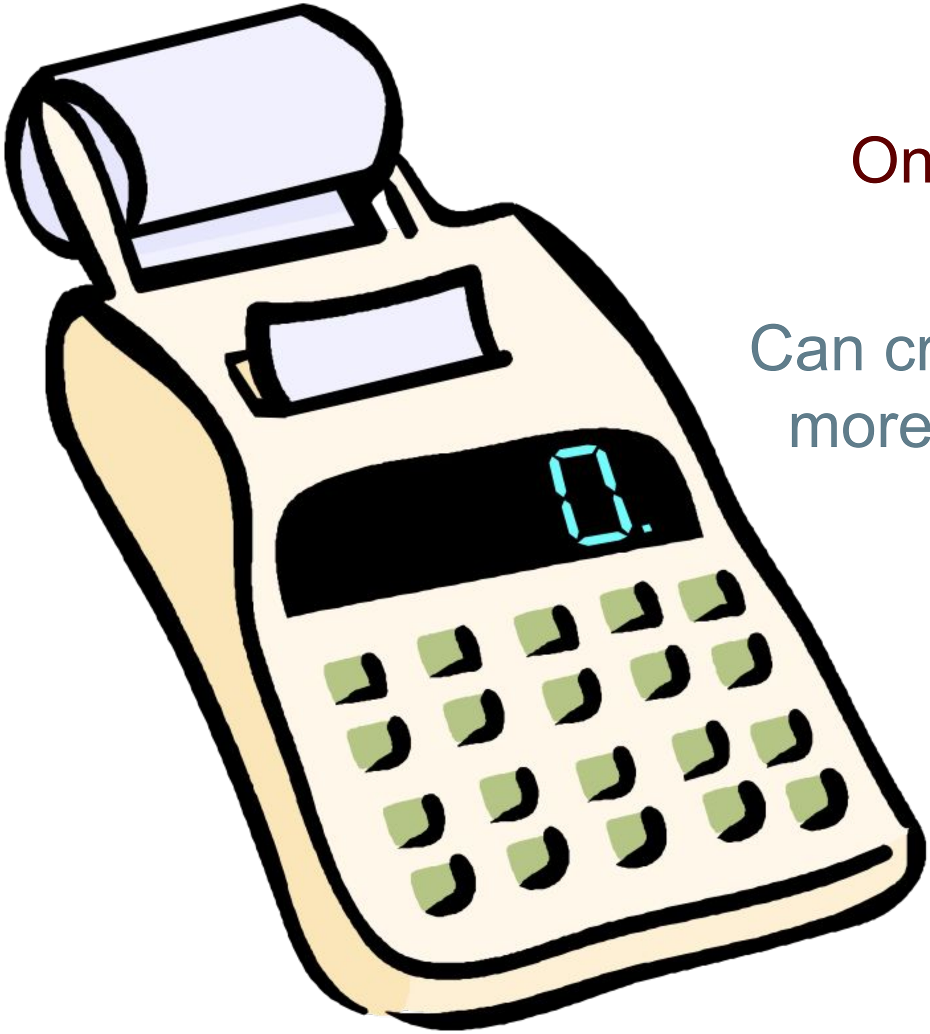
Finite Precision Arithmetic

Digits are fixed by the Manufacturer

Once you build hardware

Finite Precision Arithmetic

Digits are fixed by the Manufacturer



Once you build hardware

Can create a result that requires more digits than are available

Overflow

the generation of a result that does not fit in the data size being used

Examples using 4 bit unsigned numbers

Addition

```
  1101
+ 0100
------
```

Subtraction

```
  0001
- 0100
------
```

Examples using 4 bit unsigned numbers

Addition

```
  1101
+ 0100
```

13 + 4

Desired result = 17

Subtraction

```
  0001
- 0100
```

Examples using 4 bit unsigned numbers

Addition

```
   1101
+  0100
```

13 + 4

Desired result = 17

17 > 15 which is the largest value that fits in a 4 bit unsigned number

Subtraction

```
   0001
-  0100
```

Examples using 4 bit unsigned numbers

Addition

```
   1101
+  0100
```

13 + 4

Desired result = 17

17 > 15 which is the largest value that fits in a 4 bit unsigned number

Subtraction

```
   0001
-  0100
```

1 - 4

Desired result = -3

Examples using 4 bit unsigned numbers

Addition

```
   1101
+  0100
```

13 + 4

Desired result = 17

17 > 15 which is the largest value that fits in a 4 bit unsigned number

Subtraction

```
   0001
-  0100
```

1 - 4

Desired result = -3

-3 < 0 which is the smallest value that fits in a 4 bit unsigned number

Examples using 4 bit unsigned numbers

Addition

```
  11
   1101
 + 0100
 ------
   0001
```

Subtraction

```
   0001
 - 0100
 ------
```

Examples using 4 bit unsigned numbers

Addition

```
 11
   1101
+  0100
   0001
```

Result presented

Subtraction

```
   0001
-  0100
```

Examples using 4 bit unsigned numbers

Addition

```
 11
   1101
 + 0100
   ----
   0001
```

Result presented

Overflow detected by carry out of the Most Significant Digit (MSD)

Subtraction

```
   0001
 - 0100
   ----
```

Examples using 4 bit unsigned numbers

Addition

11

1101

+ 0100

0001

Subtraction

1

011

~~100~~01

\- 0100

1101

Examples using 4 bit unsigned numbers

Addition

11

1101

+ 0100

0001

Subtraction

1

011

~~1~~~~00~~01

- 0100

1101

Result presented

Examples using 4 bit unsigned numbers

Addition

```
  11
   1101
+  0100
   0001
```

Subtraction

```
    1
  011
 10001
-  0100
   1101
```

Result presented

Overflow detected by borrow from the Most Significant Digit (MSD)

Examples using 4 bit unsigned numbers

Addition

```
  11
    1101
+   0100
--------
    0001
```

Subtraction

```
      1
    011
   10001
-   0100
--------
    1101
```