

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?

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

9

Warm-Up: Back to Elementary School

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

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

Warm-Up: Back to Elementary School

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

$$\begin{array}{r} 11 \\ 26577 \\ + \quad 952 \\ \hline 529 \end{array}$$

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?

$$\begin{array}{r} 11 \\ 26577 \\ + \quad 952 \\ \hline 27529 \end{array}$$

Warm-Up: Back to Elementary School

Without using a calculator add:

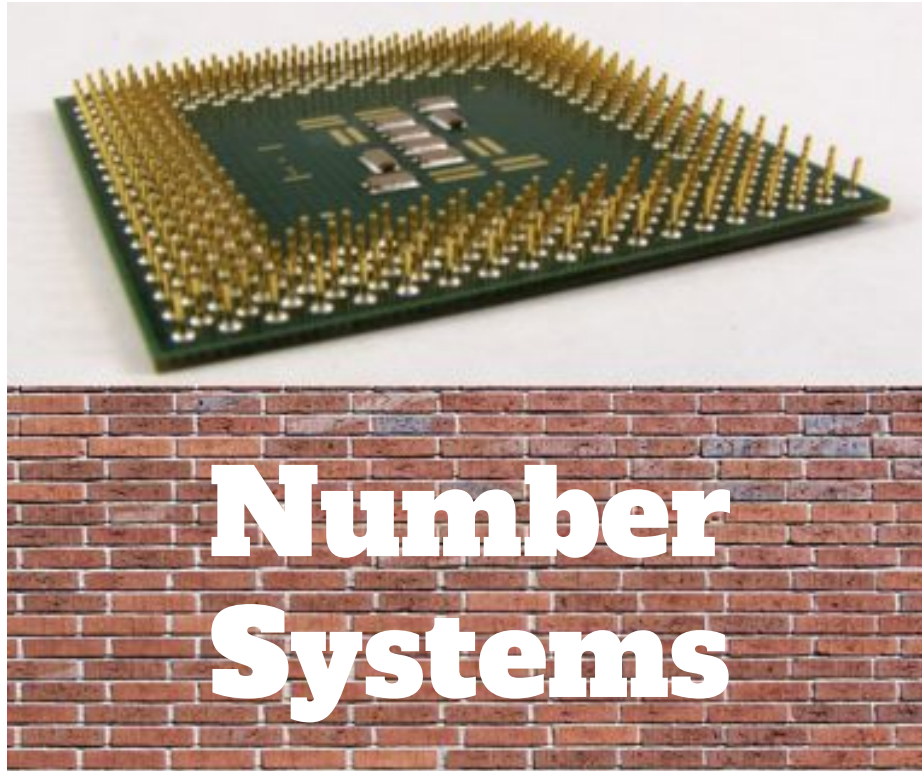
- 26577 and 952
- 99645 and 468
- 637 and 203
- 99 and 80



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



Number Systems

*The foundation upon which
Computer Architecture is built*

Unsigned Number Systems



For HWO 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

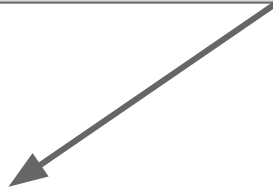
20,000



INPUT A

$B = A + A$

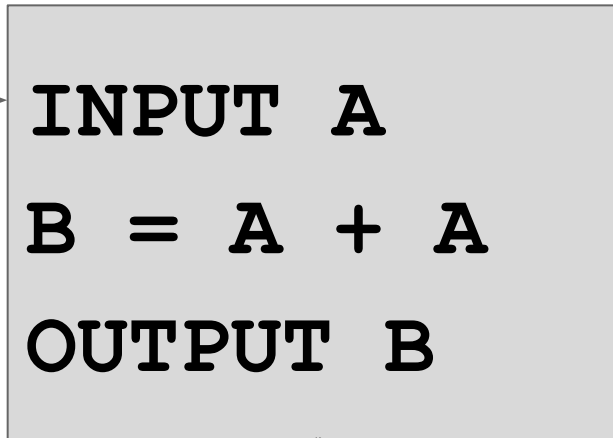
OUTPUT B



Expected:

40,000

20,000



INPUT A

$B = A + A$

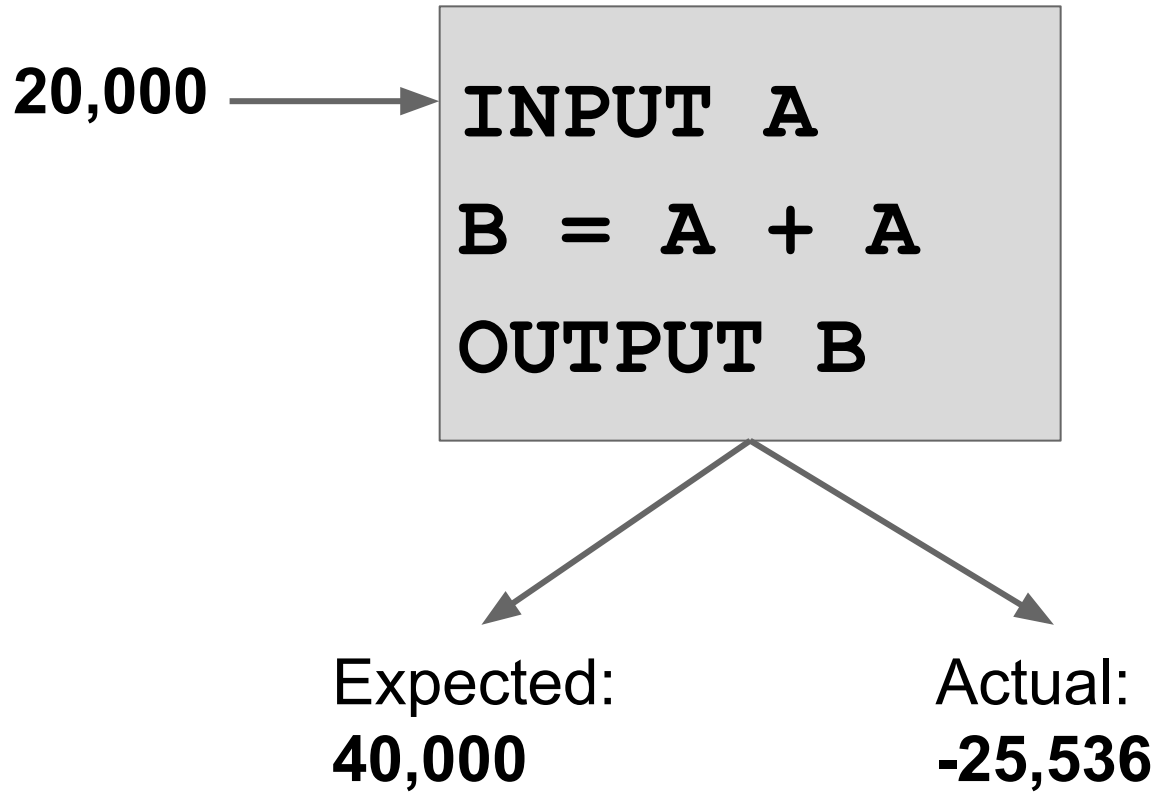
OUTPUT B

Expected:

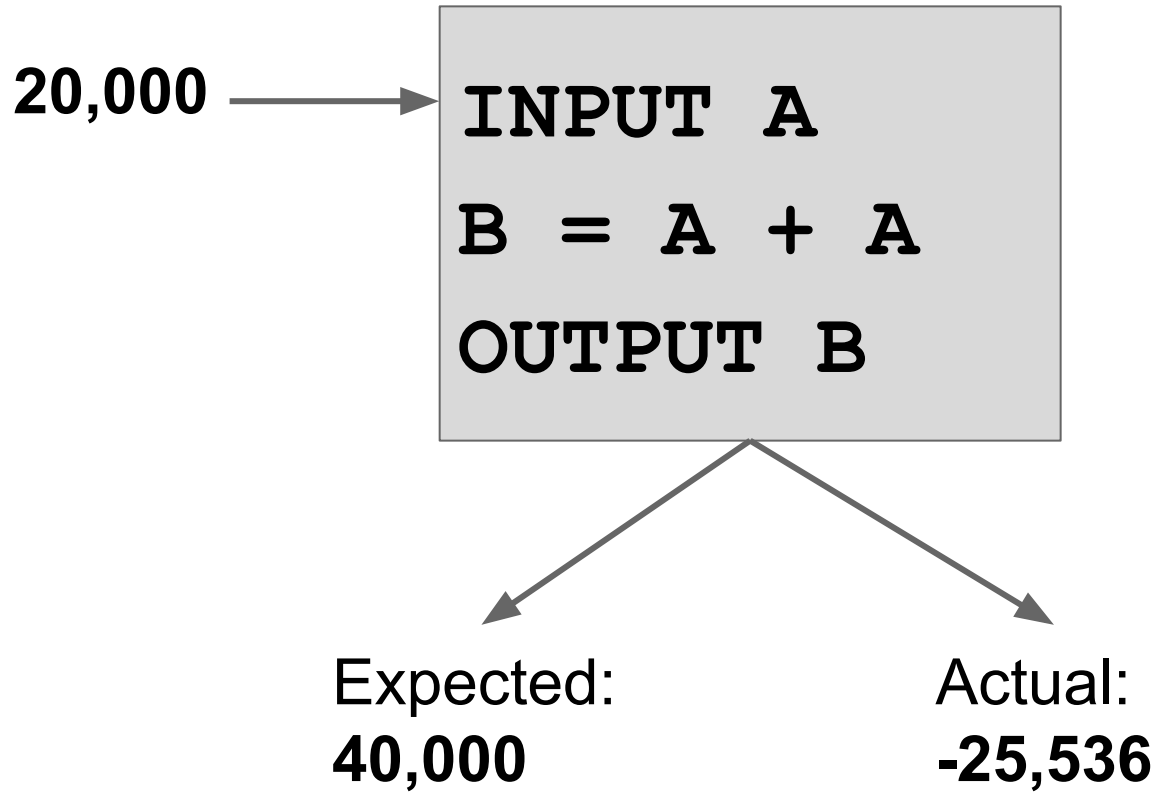
40,000

Actual:

-25,536

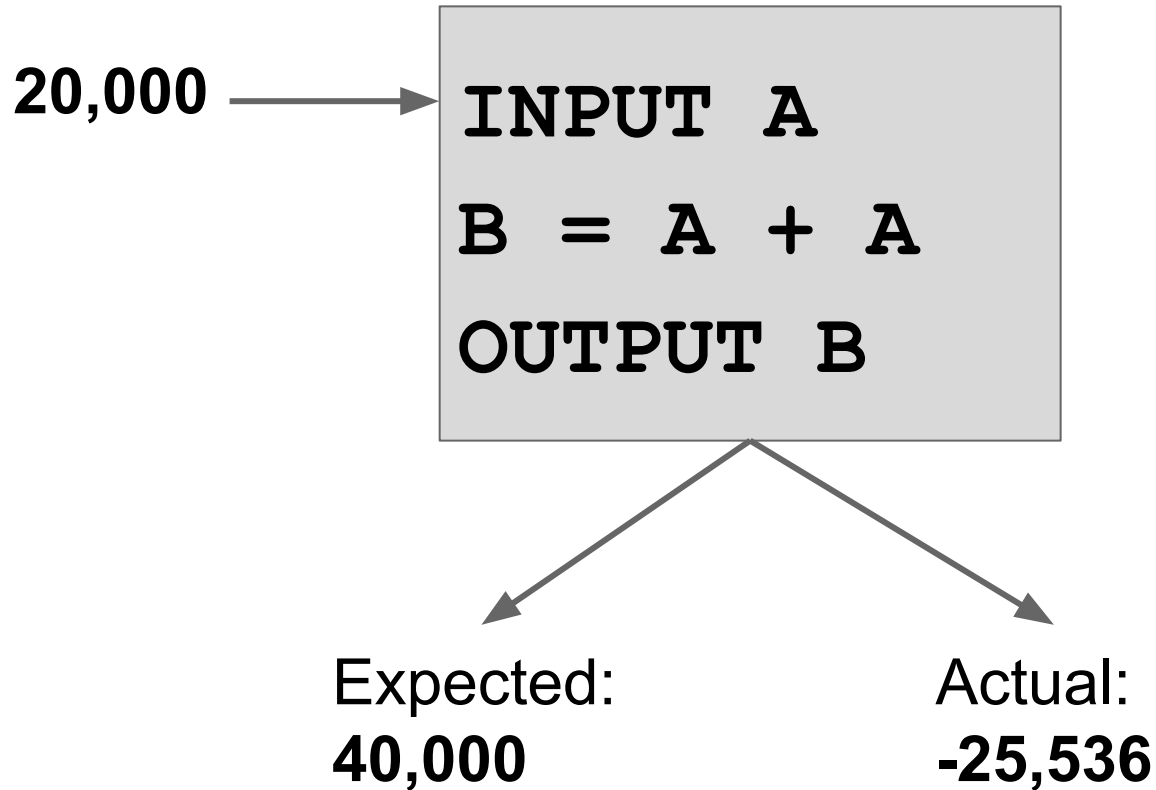


Most knew result too big for data size



Most knew result too big for data size

Which is fine

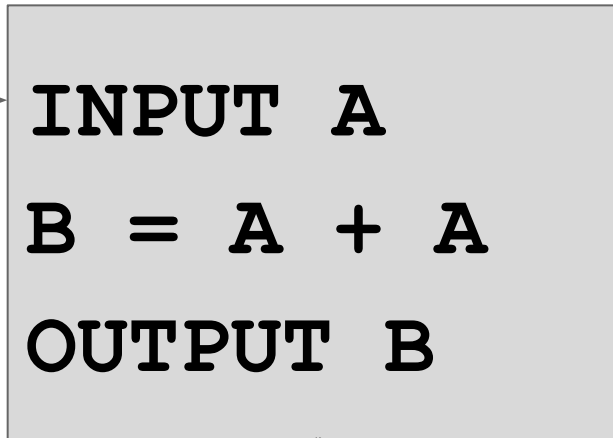


Most knew result too big for data size

Which is fine

BUT ...

20,000

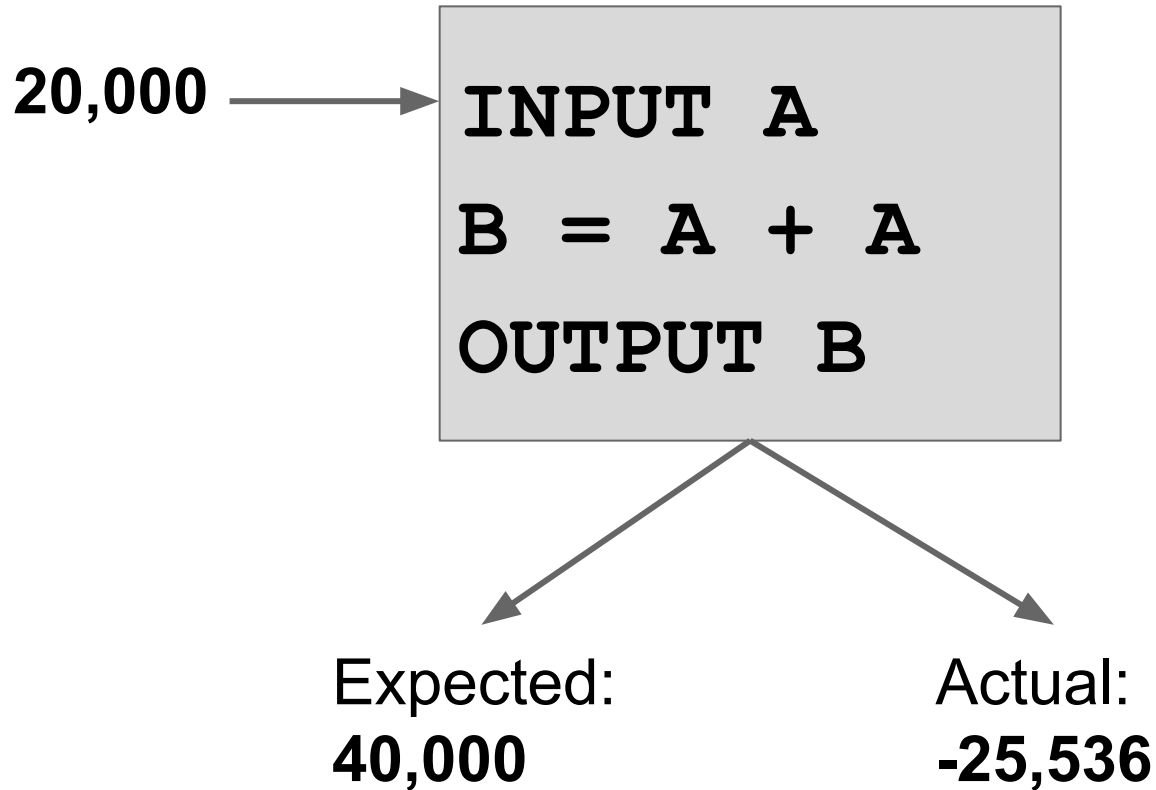


Expected:
40,000

Actual:
-25,536



*That is not the
significant part of the
problem!*

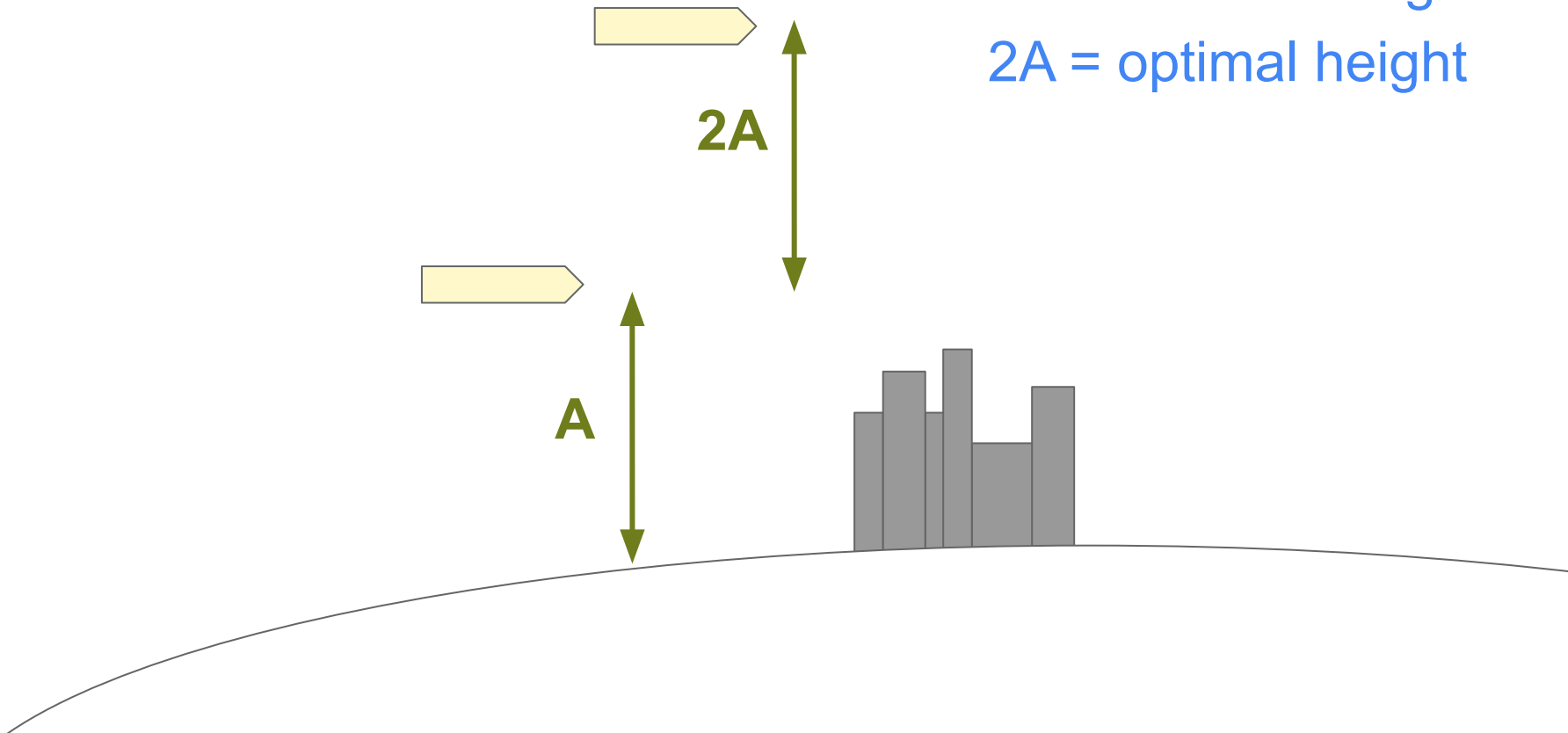


Runtime Systems did not provide a warning or error, they just output the wrong answer.

The application

Missile Control

A = minimum height
 $2A$ = optimal height



The application

Missile Control

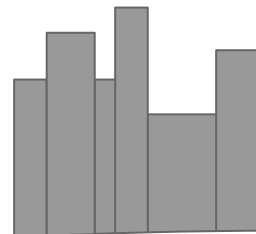
Should be
40,000 feet

Missile is launched.

Reaches height A (20,000 ft)

Software says go to 2A

20,000



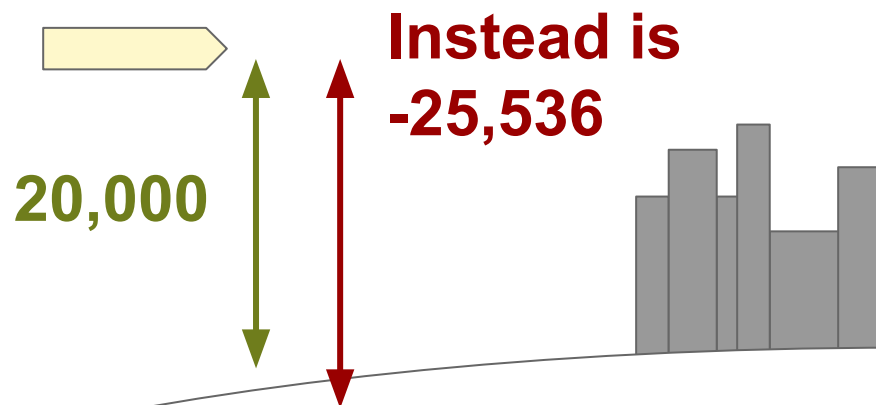
The application

Missile Control

Missile is launched.

Reaches height A (20,000 ft)

Software says go to 2A



The application

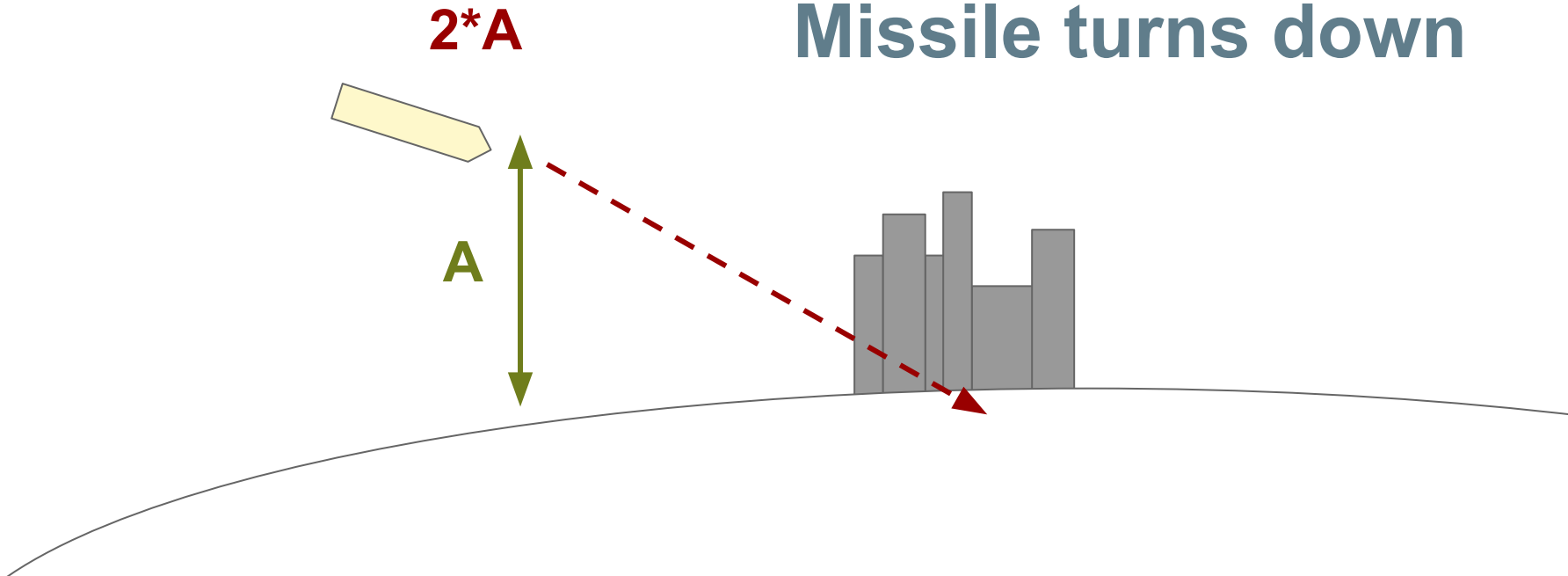
Missile Control

Missile is launched.

Reaches height A (20,000 ft)

Software says go to $2A$

Missile turns down



The application

Missile Control

Missile is launched.

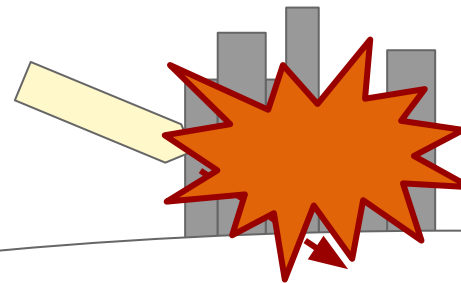
Reaches height A (20,000 ft)

Software says go to $2A$

$2 \cdot A$

Missile turns down

...



**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

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?

$$\begin{array}{r} \text{II} \\ + \text{I} \\ \hline \text{III} \end{array}$$

$$\begin{array}{r} \text{III} \\ + \text{I} \\ \hline \text{IV} \end{array}$$

$$\begin{array}{r} \text{XI} \\ + \text{V} \\ \hline \text{XVI} \end{array}$$

Positional Number Systems

Designed to allow straightforward rules when performing arithmetic

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
-------------	-------------	-------------------------------



Common Set of Characteristics

Positional Number Systems

Designed to allow straightforward rules when performing arithmetic

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
Decimal	10	0-9

Common Set of Characteristics



Positional Number Systems

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
Decimal	10	0-9

Humans tend to use base 10. *Why?*

Positional Number Systems

Humans tend to use base 10. *Why?*

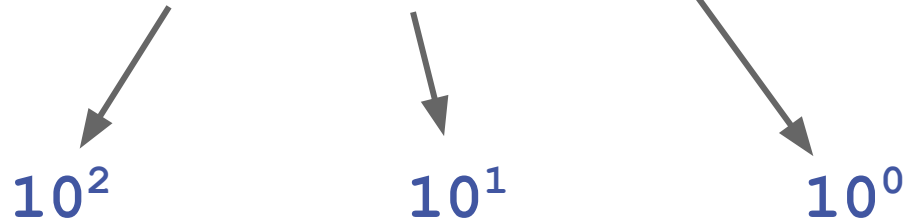


https://commons.wikimedia.org/wiki/File:Two_hand_ten_fingers.jpg

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



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

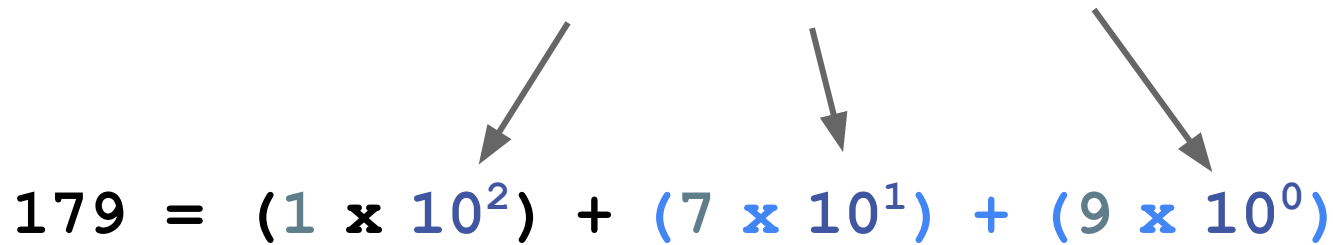


Diagram illustrating the positional expansion of the number 179. Three arrows point from the text above to the terms in the equation below: the first arrow points to 10^2 , the second to 10^1 , and the third to 10^0 .

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



Diagram illustrating the positional expansion of the number 179 into its numerical components. Three blue arrows point from the text below to the terms in the equation above: the first arrow points from 'value' to 100, the second from 'digit' to 70, and the third from 'position value' to 9.

$$179 = 100 + 70 + 9$$

value = digit * position value

Positional Number Systems

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
Decimal	10	0-9

What do computers use?

Two Types of Computer

Digital
Computer



Two Types of Computer

Digital
Computer

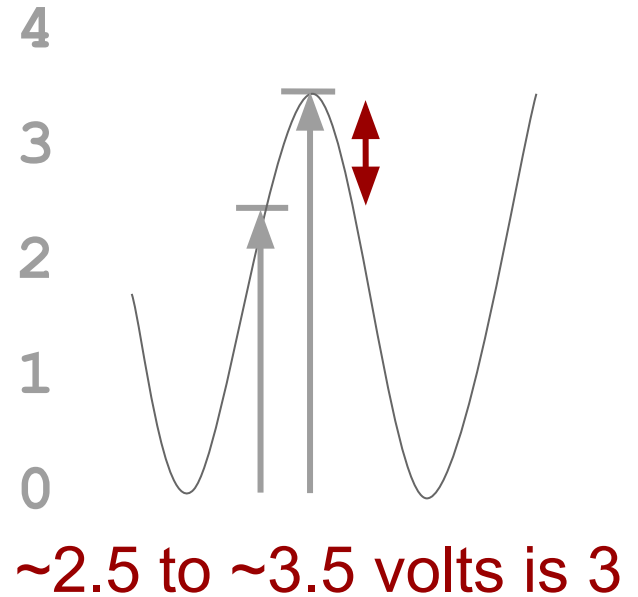


Heathkit **Analog**
Computer circa 1956



Two Types of Computer

Analog Computer



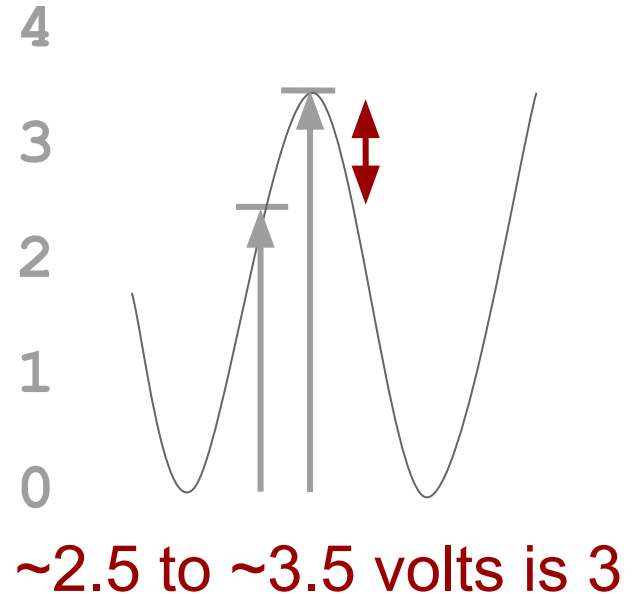
Uses Voltage Level to
represent number

Two Types of Computer

Analog Computer

Error Prone

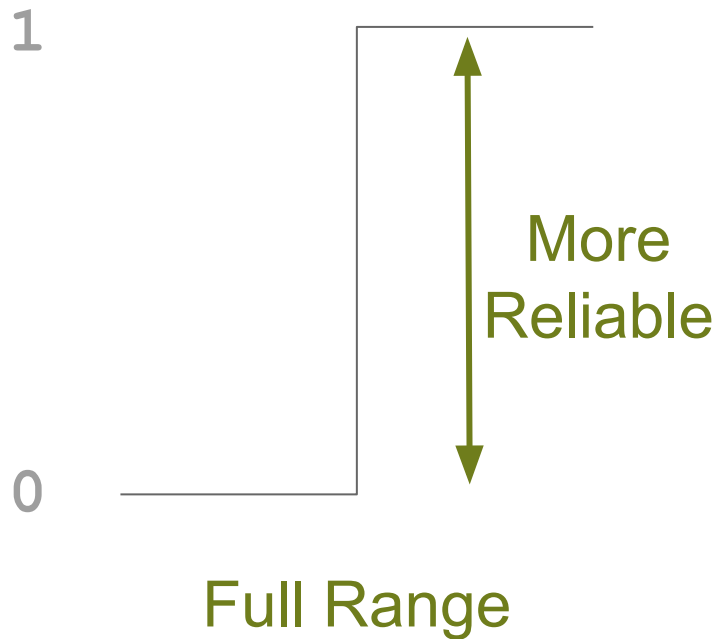
Being a little bit off
gives the wrong
number



Uses Voltage Level to
represent number

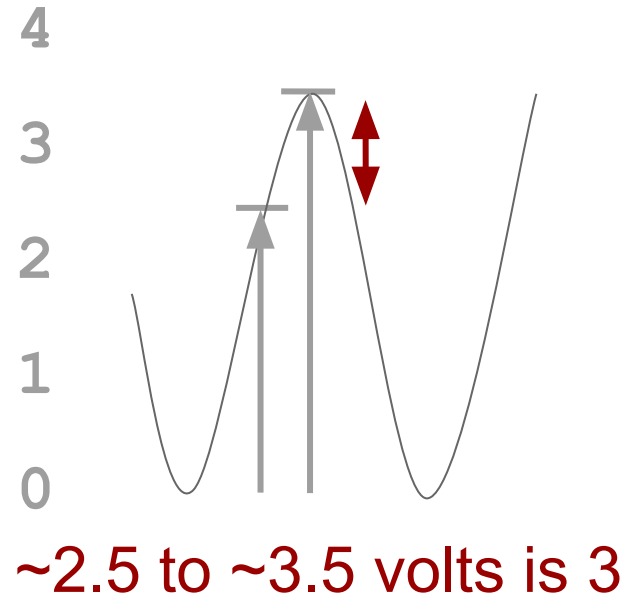
Two Types of Computer

Digital Computer



Uses Voltage On/Off to represent a number

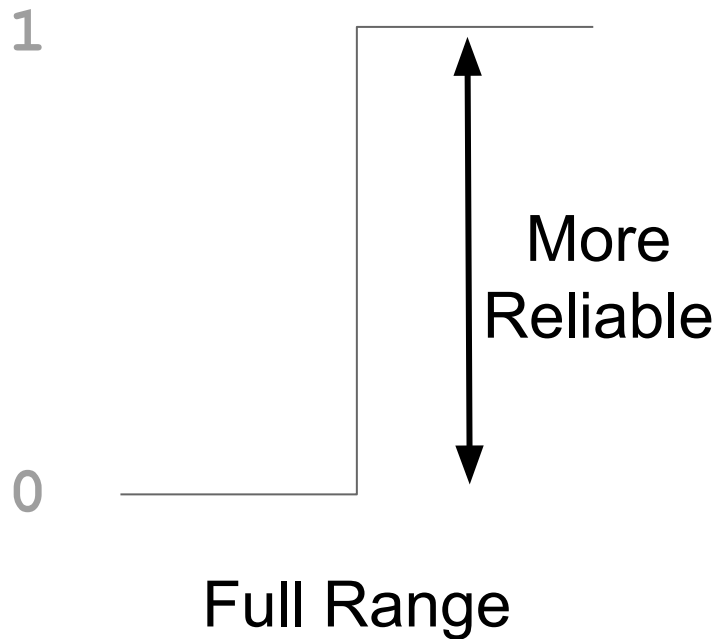
Analog Computer



Uses Voltage Level to represent number

Two Types of Computer

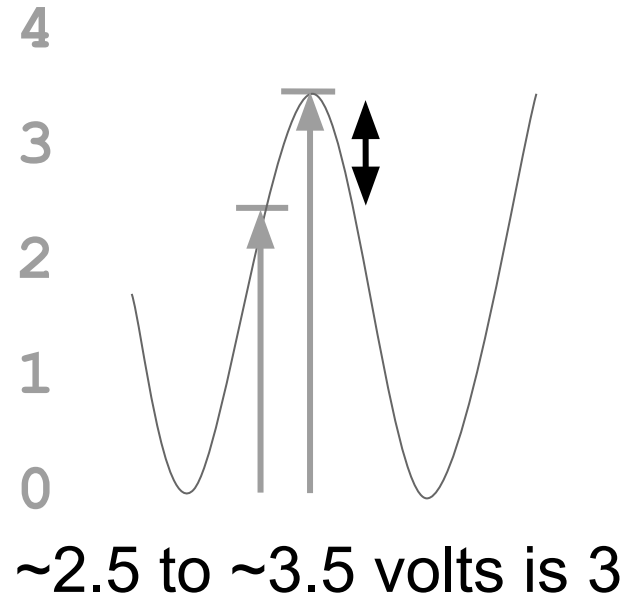
Digital Computer



Uses Voltage On/Off to represent a number

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

Analog Computer



Uses Voltage Level to represent number

Base 2

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
Decimal	10	0-9
Binary	2	0-1

Digital computers use base 2

Base 2

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
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

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
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

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
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 single digit



Else (the sum is greater than the base) the result is reduced by the base and is 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

$$\begin{array}{r} 11 \\ 26577 \\ + \quad 952 \\ \hline 27529 \end{array}$$

Else (the result is greater than the base then the result is reduced by a value equal to 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 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

**This rule is very basic for
unsigned numbers**

Else (the result is greater than the base) the result is reduced by a value equal to the base to 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

This rule

becomes critical

when we address signed
numbers

Else if the sum is greater than the base, the result is reduced by a value equal to 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

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

Else (the result is greater than the base) the result is reduced by a multiple of the base to 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

*source of many security
vulnerabilities*

Else (the result is greater than the base) the result is reduced by a value equal to the base to the next higher position

Decimal

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

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

Binary

$$\begin{array}{r} 0 \\ + 0 \\ \hline \end{array}$$

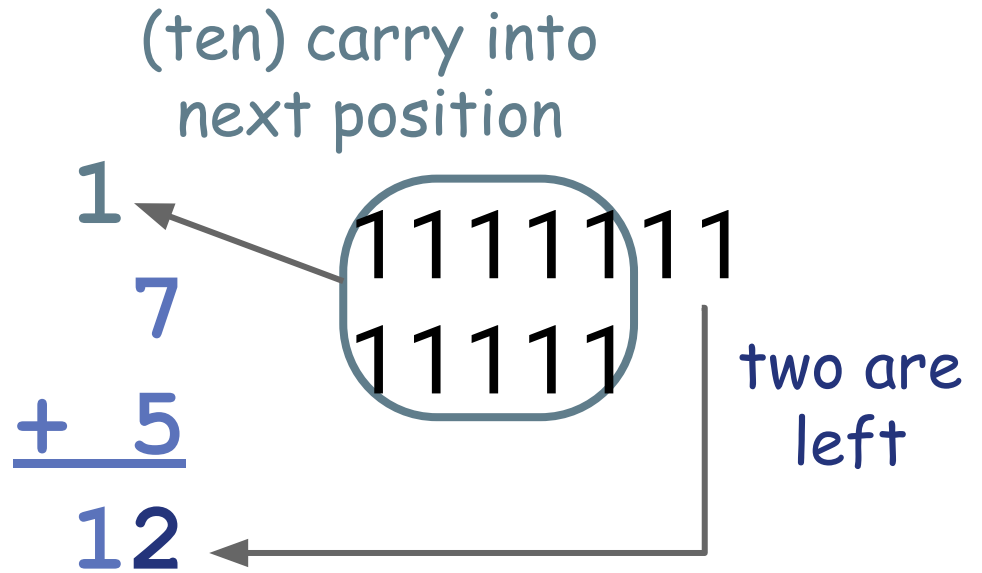
$$\begin{array}{r} 0 \\ + 1 \\ \hline \end{array}$$

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

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

Decimal

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



Binary

$$\begin{array}{r} 0 \\ + 0 \\ \hline \end{array}$$

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

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

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

Decimal

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

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

Binary

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

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

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

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

Decimal

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

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

Binary

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

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

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

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

carry into
next
position

1

Apply the Rule to Multiple Digits

Did this in the warm-up with Base-10

$$\begin{array}{r} 11 \\ 26577 \\ + \quad 952 \\ \hline 27529 \end{array}$$

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

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

Apply the Rule to Multiple Digits

11

$$1011_2 = 11_{10}$$

$$+ \underline{1011}_2 = \underline{11}_{10}$$

$$10_2$$

$$22_{10}$$

Apply the Rule to Multiple Digits

11

$$1011_2 = 11_{10}$$

$$+ \underline{1011}_2 = \underline{11}_{10}$$

110₂

22₁₀

Apply the Rule to Multiple Digits

1 11

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

Apply the Rule to Multiple Digits

1 11

$$1011_2 = 11_{10}$$

$$+ \underline{1011}_2 = \underline{11}_{10}$$

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

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

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

$$10110_2 = 16 \qquad \qquad \qquad + \quad 4 \quad + \quad 2$$

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

Full Expansion

Conversion of Binary to Decimal

Expansion of Powers

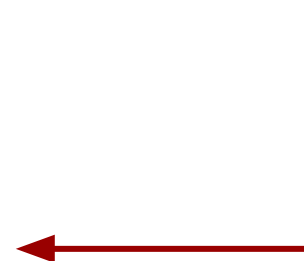
$$10110_2 = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$

$$10110_2 = 16 \qquad \qquad \qquad + \quad 4 \quad + \quad 2$$

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

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

1	0	1	1	0
16	8	4	2	1



Conversion of Binary to Decimal

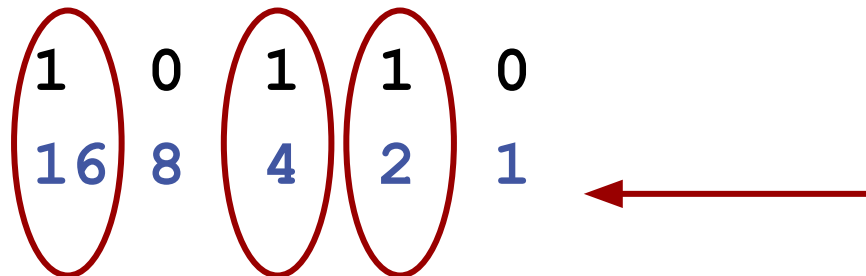
Expansion of Powers

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

$$10110_2 = 16 \qquad \qquad \qquad + \quad 4 \quad + \quad 2$$

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

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



Conversion of Binary to Decimal

Expansion of Powers

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

$$10110_2 = 16 \qquad \qquad \qquad + \quad 4 \quad + \quad 2$$

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

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

$$22_{10} =$$

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

$$2^5 = 32$$

$$32 > 22$$

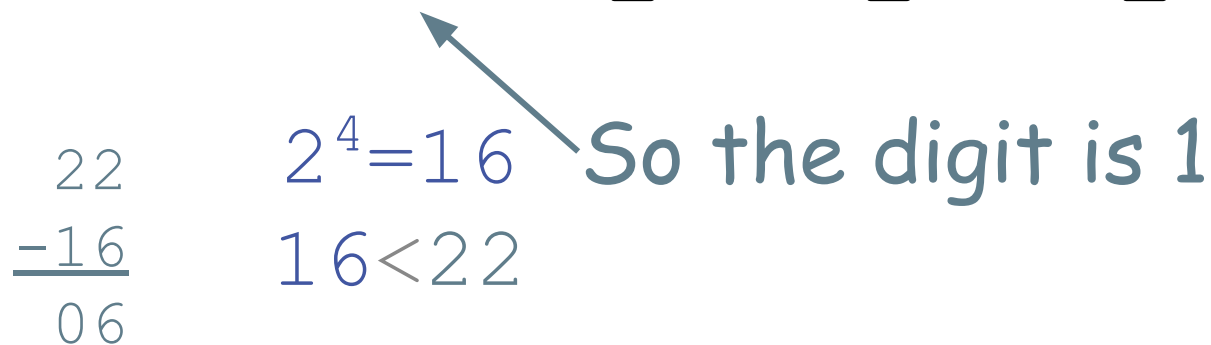
So the digit is 0

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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



22

-16

06

$2^4 = 16$

$16 < 22$

So the digit is 1

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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

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

$$2^3 = 8$$

$$8 > 6$$

So the digit is 0



Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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

$$2^2 = 4$$

$$4 < 6$$

So the digit is 1

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

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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

$$2^1 = 2$$

$$2 = 2$$

So the digit is 1

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

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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

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

0 left, so any remaining
digits are zero

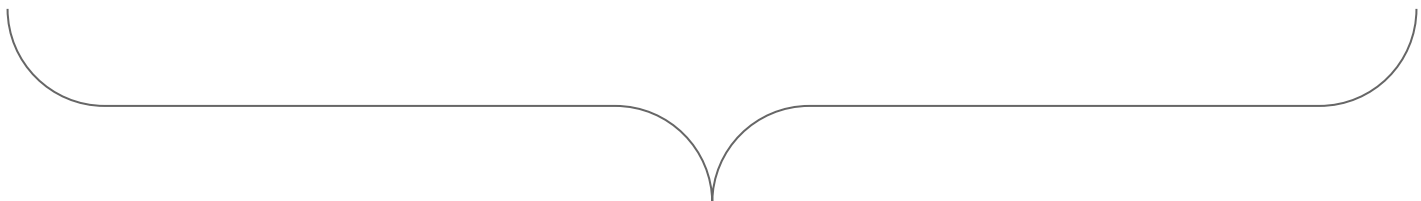


Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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

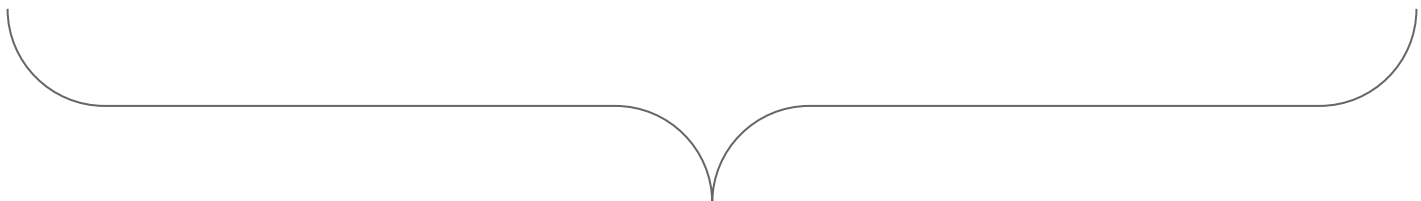

$$22_{10} = 010110_2$$

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} =$$

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


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



In unsigned arithmetic, leading zeros are not significant

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} = (0 \times 2^5) +$$



Are leading zeros significant in signed numbers?

In unsigned arithmetic, leading zeros are not significant

Conversion of Decimal to Binary

Reduction of Powers

$$22_{10} = (0 \times 2^5) +$$



Yes!

We will see that
next time

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++)

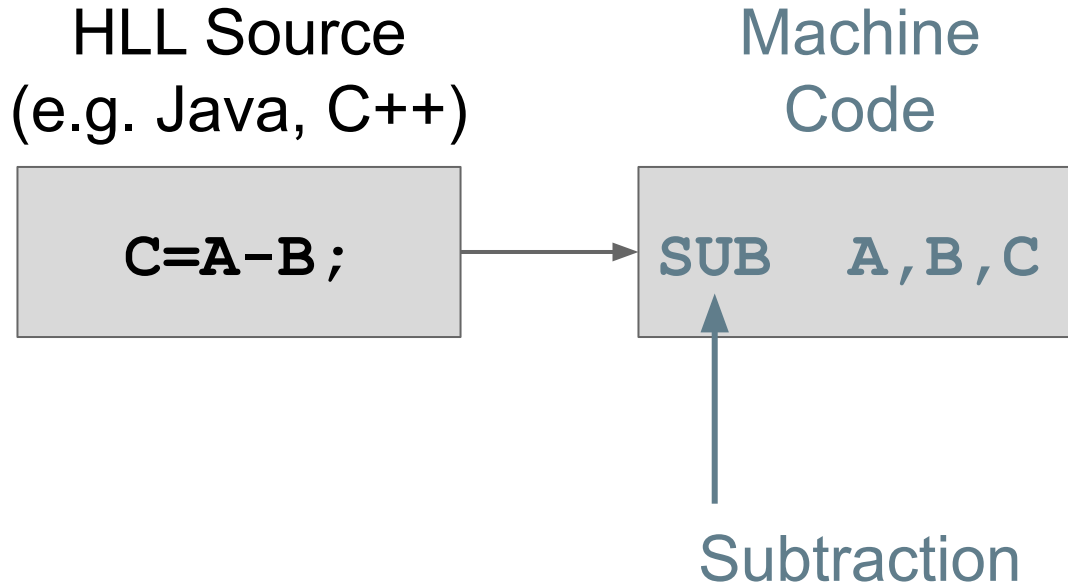


```
C=A-B;
```

Subtraction

Subtraction

How do computers subtract?



Subtraction

How do computers subtract?

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

$C = A - B;$

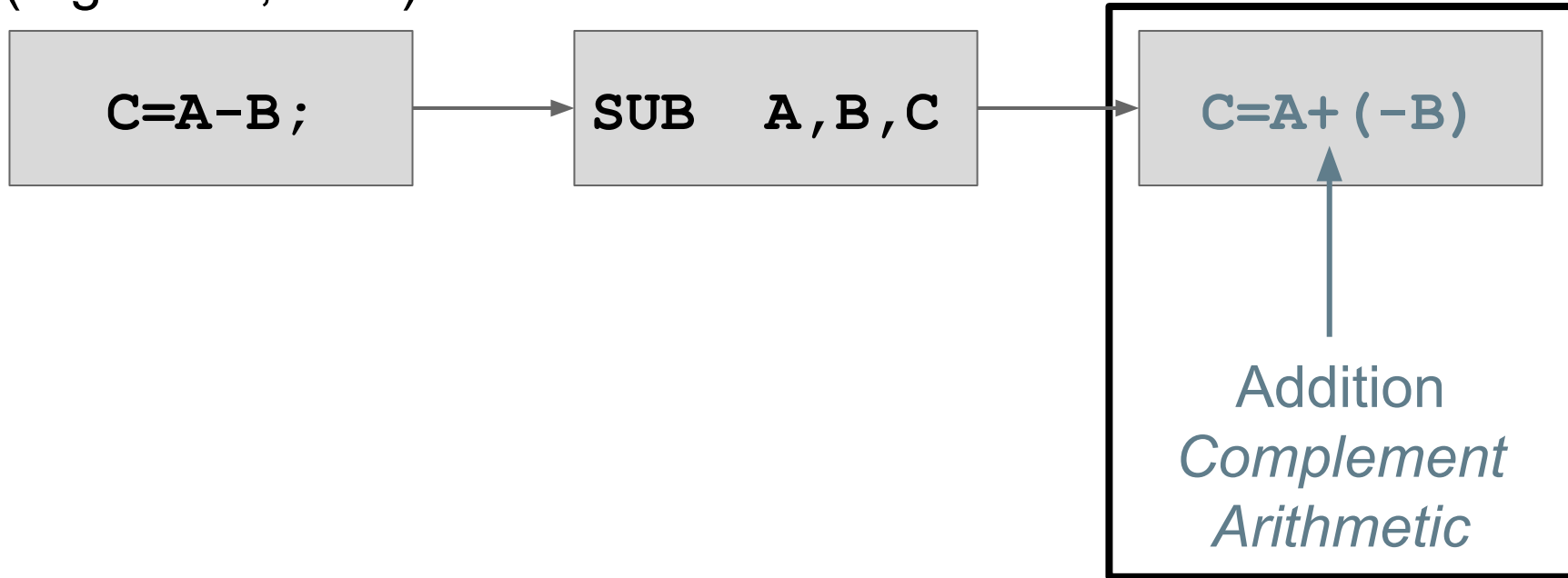
Machine
Code

SUB A, B, C

Hardware

$C = A + (-B)$

↑
Addition
*Complement
Arithmetic*



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++)

$C = A - B;$

Machine
Code

`SUB A, B, C`

Hardware

$C = A + (-B)$

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

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

Re-write in true positional form

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

Decimal Subtraction

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



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

Decimal Subtraction

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

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

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

Decimal Subtraction

We will *borrow* from
the next column

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



Decimal Subtraction

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



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

Each 10^1 equals 10×10^0

Decimal Subtraction

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



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

Decimal Subtraction


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



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

Decimal Subtraction

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


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

Decimal Subtraction


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



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

Decimal Subtraction

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


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

78

Binary Subtraction


Binary Subtraction

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

Binary Subtraction

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


Re-write in true positional form


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

Binary Subtraction

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

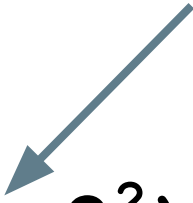
Need to redistribute
data to continue


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

Binary Subtraction

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

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

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

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

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

Binary Subtraction

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

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

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

Binary Subtraction

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

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

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

Binary Subtraction

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

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

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

+ (1×2^0)
+ (1×2^0)

Binary Subtraction

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

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

(Note: In the original image, the terms (1×2^0) and (1×2^0) in the second column are circled together, and the term (0×2^0) in the third column is crossed out.)

Binary Subtraction

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

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

The diagram illustrates the binary subtraction process using positional weights. The minuend is 100_2 (4 in decimal) and the subtrahend is 001_2 (1 in decimal). The result is 011_2 (3 in decimal). The diagram shows the subtraction of the subtrahend from the minuend, with the result being 011_2 . The terms are arranged in a grid-like fashion, with the minuend terms on the left, the subtrahend terms in the middle, and the result terms on the right. The terms are grouped by their positional weights: 2^2 , 2^1 , and 2^0 . The terms are also grouped by their sign: positive terms are on the left, and negative terms are on the right. The terms are also grouped by their value: 1×2^2 is 4, 1×2^1 is 2, and 1×2^0 is 1. The terms are also grouped by their position: the first row contains the minuend terms, the second row contains the subtrahend terms, and the third row contains the result terms. The terms are also grouped by their color: the minuend terms are blue, the subtrahend terms are black, and the result terms are black. The terms are also grouped by their font size: the minuend terms are larger than the subtrahend terms, and the result terms are the same size as the subtrahend terms. The terms are also grouped by their alignment: the minuend terms are aligned to the left, the subtrahend terms are aligned to the left, and the result terms are aligned to the left. The terms are also grouped by their spacing: the minuend terms are spaced out, the subtrahend terms are spaced out, and the result terms are spaced out. The terms are also grouped by their background: the minuend terms have a light blue background, the subtrahend terms have a light blue background, and the result terms have a light blue background. The terms are also grouped by their border: the minuend terms have a light blue border, the subtrahend terms have a light blue border, and the result terms have a light blue border. The terms are also grouped by their shadow: the minuend terms have a light blue shadow, the subtrahend terms have a light blue shadow, and the result terms have a light blue shadow. The terms are also grouped by their opacity: the minuend terms are semi-transparent, the subtrahend terms are semi-transparent, and the result terms are semi-transparent. The terms are also grouped by their rotation: the minuend terms are rotated 0 degrees, the subtrahend terms are rotated 0 degrees, and the result terms are rotated 0 degrees. The terms are also grouped by their skew: the minuend terms are skewed 0 degrees, the subtrahend terms are skewed 0 degrees, and the result terms are skewed 0 degrees. The terms are also grouped by their stretch: the minuend terms are stretched 0%, the subtrahend terms are stretched 0%, and the result terms are stretched 0%. The terms are also grouped by their weight: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their height: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their width: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their area: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their perimeter: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their circumference: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their diameter: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their radius: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their area: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their perimeter: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their circumference: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their diameter: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%. The terms are also grouped by their radius: the minuend terms are 100%, the subtrahend terms are 100%, and the result terms are 100%.

Binary Subtraction

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

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

The diagram illustrates the binary subtraction process using positional weights. The minuend is 100_2 (4 in decimal) and the subtrahend is 001_2 (1 in decimal). The result is 011_2 (3 in decimal). The diagram shows the subtraction of the subtrahend from the minuend, with the result being 011_2 . The terms are arranged in a grid, with the minuend terms in the first row, the subtrahend terms in the second row, and the result terms in the third row. The terms are grouped by positional weight: 2^2 , 2^1 , and 2^0 . The terms are also grouped by sign: positive terms are in the first column, and negative terms are in the second column. The terms are also grouped by value: the terms with a value of 4 are in the first column, the terms with a value of 2 are in the second column, and the terms with a value of 1 are in the third column. The terms are also grouped by the number of terms: the terms with a value of 4 are in the first column, the terms with a value of 2 are in the second column, and the terms with a value of 1 are in the third column. The terms are also grouped by the number of terms: the terms with a value of 4 are in the first column, the terms with a value of 2 are in the second column, and the terms with a value of 1 are in the third column.

Binary Subtraction

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

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

The diagram illustrates the binary subtraction process using positional weights. The minuend is 100_2 (4 in decimal) and the subtrahend is 001_2 (1 in decimal). The result is 011_2 (3 in decimal). The expansion shows the subtraction of the subtrahend from the minuend, with borrowing indicated by the blue annotations. The final result is $(0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$.

Binary Subtraction

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

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

Diagram illustrating the binary subtraction process using positional weights. The minuend is 100_2 (4 in decimal) and the subtrahend is 001_2 (1 in decimal). The result is 011_2 (3 in decimal). The diagram shows the subtraction of the subtrahend from the minuend, with the result being the sum of the remaining terms: $(0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$.

**Can we speed up
subtraction?**

Can we speed up subtraction?

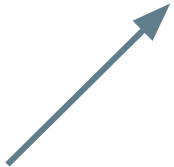
1000_{10}



Re-write the 1
followed by 0s

Can we speed up subtraction?

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



Re-write the 1
followed by 0s

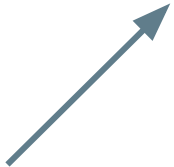


As 0 followed
by 9s plus 1

Can we speed up subtraction?

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

Re-write the 1
followed by 0s



As 0 followed
by 9s plus 1



How does this help?

Can we speed up subtraction?

$$\begin{array}{r} 1000 \\ - \underline{0625} \end{array}$$

Can we speed up subtraction?

$$\begin{array}{r} 1000 \\ - \underline{0625} \end{array}$$

 This requires multiple
re-distributions of data

Can we speed up subtraction?

$$\begin{array}{r} 1 \\ 1000 \\ - \underline{0625} \end{array} = \begin{array}{r} 0999 \\ - \underline{0625} \end{array}$$

This requires multiple
re-distributions of data



Can we speed up subtraction?

$$\begin{array}{r} 1 \\ 1000 \\ - \underline{0625} \end{array} = \begin{array}{r} 0999 \\ - \underline{0625} \end{array}$$

This requires multiple
re-distributions of data



This requires no
re-distributions




Can we speed up subtraction?

$$\begin{array}{r} 1000 \\ -0625 \\ \hline \end{array} = \begin{array}{r} 1 \\ 0999 \\ -0625 \\ \hline \end{array}$$

This requires multiple
re-distributions of data



5
This requires no
re-distributions



Can we speed up subtraction?

$$\begin{array}{r} 1000 \\ -0625 \\ \hline \end{array} = \begin{array}{r} 1 \\ 0999 \\ -0625 \\ \hline \end{array}$$

This requires multiple
re-distributions of data

75
This requires no
re-distributions

Can we speed up subtraction?

$$\begin{array}{r} 1000 \\ -0625 \\ \hline \end{array} = \begin{array}{r} 1 \\ 0999 \\ -0625 \\ \hline \end{array}$$

375

This requires multiple
re-distributions of data

This requires no
re-distributions

Can we speed up subtraction?

$$\begin{array}{r} 1000 \\ -0625 \\ \hline \end{array} = \begin{array}{r} 1 \\ 0999 \\ -0625 \\ \hline 0375 \end{array}$$

This requires multiple
re-distributions of data

This requires no
re-distributions

Can we speed up subtraction?

$$\begin{array}{r} 1 \\ 1000 \\ - \underline{0625} \\ \hline \end{array} = \begin{array}{r} 0999 \\ - \underline{0625} \\ \hline 0375 \end{array}$$

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?

$$\begin{array}{r} 22_{10} \\ -11_{10} \\ \hline \end{array} = \begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \quad 0_2 \\ -0 \quad 1 \quad 0 \quad 1 \quad 1_2 \\ \hline \end{array}$$

Can we speed up subtraction?

$$\begin{array}{r} 22_{10} \\ -11_{10} \\ \hline \end{array} = \begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \quad 0_2 \\ -0 \quad 1 \quad 0 \quad 1 \quad 1_2 \\ \hline \end{array}$$

This requires
redistribution
of data

Can we speed up subtraction?

$$\begin{array}{r} 22_{10} \\ -11_{10} \\ \hline \end{array} = \begin{array}{r} 1 \quad 0 \quad 1 \quad \cancel{1} \quad \cancel{0} \\ -0 \quad 1 \quad 0 \quad 1 \quad 1 \\ \hline \end{array} \begin{array}{r} 1 \\ 1 \\ 2 \end{array}$$

Can we speed up subtraction?

$$\begin{array}{r} 22_{10} \\ -11_{10} \\ \hline \end{array} = \begin{array}{r} 1 \quad 0 \quad 1 \quad 1 \quad 0 \\ -0 \quad 1 \quad 0 \quad 1 \quad 1 \\ \hline 1 \end{array}$$

Can we speed up subtraction?

$$\begin{array}{r}
 22_{10} \\
 -11_{10} \\
 \hline
 \end{array}
 =
 \begin{array}{r}
 1 \quad 0 \quad 1 \quad 1 \quad 0_2 \\
 -0 \quad 1 \quad 0 \quad 1 \quad 1_2 \\
 \hline
 1
 \end{array}$$

Diagram illustrating a subtraction operation in binary. The top row shows the minuend 22_{10} converted to binary 10110_2 . The bottom row shows the subtrahend -11_{10} converted to binary -01011_2 . The result of the subtraction is 11011_2 . A blue line connects the 1 in the third column of the top row to the 0 in the fourth column of the top row, indicating a borrow operation.

Can we speed up subtraction?

$$\begin{array}{r}
 22_{10} \\
 -11_{10} \\
 \hline
 \end{array}
 =
 \begin{array}{r}
 1 \quad 0 \quad 1 \quad 1 \quad 0_2 \\
 -0 \quad 1 \quad 0 \quad 1 \quad 1_2 \\
 \hline
 1 \quad 1
 \end{array}$$

Diagram illustrating the subtraction of 11 from 22 in binary. The top row shows the minuend 22 in decimal (10110 in binary) and the subtrahend 11 in decimal (01011 in binary). The bottom row shows the result of the subtraction, which is 11 in decimal (101 in binary). A blue line connects the 1 in the 4th column of the top row to the 0 in the 4th column of the bottom row, indicating a borrow operation.

Can we speed up subtraction?

$$\begin{array}{r}
 22_{10} \\
 -11_{10} \\
 \hline
 \end{array}
 =
 \begin{array}{r}
 1 \quad 0 \quad 1 \quad 1 \quad 0_2 \\
 -0 \quad 1 \quad 0 \quad 1 \quad 1_2 \\
 \hline
 0 \quad 1 \quad 1
 \end{array}$$

Diagram illustrating the subtraction of 11 from 22 in binary. The top row shows the binary representation of 22 (10110) and the bottom row shows the binary representation of 11 (01011). The result of the subtraction is 011 (3 in decimal). A blue '0' is highlighted above the third column, and a line connects it to the '1' in the fourth column of the top row, indicating a borrow operation.

Can we speed up subtraction?

$$\begin{array}{r} 22_{10} \\ -11_{10} \\ \hline \end{array} = \begin{array}{r} 1 \\ 1 1 \\ 0 1 0 \\ \underline{1 0 1} \\ 0 1 1 \end{array}$$

Can we speed up subtraction?

22_{10}
 -11_{10}
 \hline

$=$
 $=$

1
 1 1
 0 1 0 0 1
 1 0 1 1 0
 \hline
 1 0 1 1 0

Can we speed up subtraction?

$$\begin{array}{r}
 22_{10} \\
 -11_{10} \\
 \hline
 \end{array}
 =
 \begin{array}{r}
 \\
 \\
 \\
 \\
 \\
 \hline

 \end{array}$$

Diagram illustrating the subtraction of 11 from 22 in binary, showing the borrowing process. The result is 11 in binary.

Can we speed up subtraction?

$$\begin{array}{r}
 22_{10} \\
 -11_{10} \\
 \hline
 11_{10}
 \end{array}
 =
 \begin{array}{r}
 111 \\
 01001 \\
 \underline{10101} \\
 01011_2
 \end{array}$$





Diagram illustrating the subtraction of 11 from 22 in binary, showing the borrowing process. The result is 11 in decimal, which is 1011 in binary.

Can we speed up subtraction?

$$\begin{array}{r}
 22_{10} \\
 -11_{10} \\
 \hline
 11_{10}
 \end{array}
 =
 \begin{array}{r}
 1 \\
 1 \\
 010 \\
 \hline
 101
 \end{array}
 =
 \begin{array}{r}
 1 \\
 11 \\
 010 \\
 \hline
 010
 \end{array}
 =
 \begin{array}{r}
 1 \\
 11 \\
 010 \\
 \hline
 010
 \end{array}
 =
 \begin{array}{r}
 1 \\
 11 \\
 010 \\
 \hline
 010
 \end{array}$$

Diagram illustrating the subtraction of 11 from 22 in binary, showing the borrowing process and the resulting binary result 11.

Concepts so far

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

Long binary numbers are hard for
humans to remember

The binary
value of PI is
11.0100110101
01010101001...



Long binary numbers are hard for *humans* to remember

Darn ... I can
only remember
7 digits

The binary
value of PI is
11.0100110101
01010101001...



Shortcut to working in binary

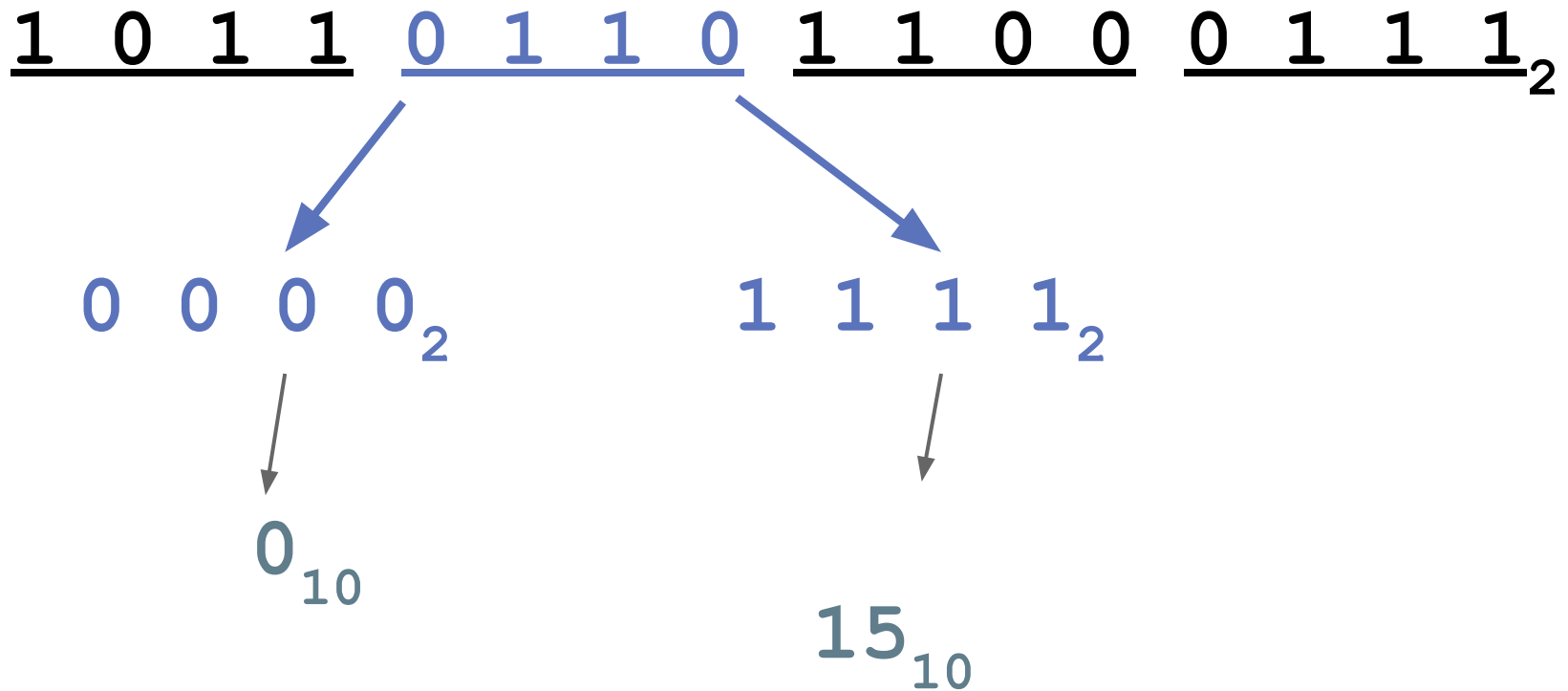
1 0 1 1 0 1 1 0 1 1 0 0 0 1 1 1₂

Shortcut to working in binary

1 0 1 1 0 1 1 0 1 1 0 0 0 1 1 1₂

Break a binary number into
groups of 4 binary digits

Shortcut to working in binary



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

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

Proper term would be senidenary

Hexadecimal - base 16

<u>Name</u>	<u>Base</u>	<u>Digits (0 thru base-1)</u>
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 \leftrightarrow Binary

... lookup in table

1 1 0 0 0 1 0 1 1 1 1 0₂

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 0₂



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 0₂

↕

C

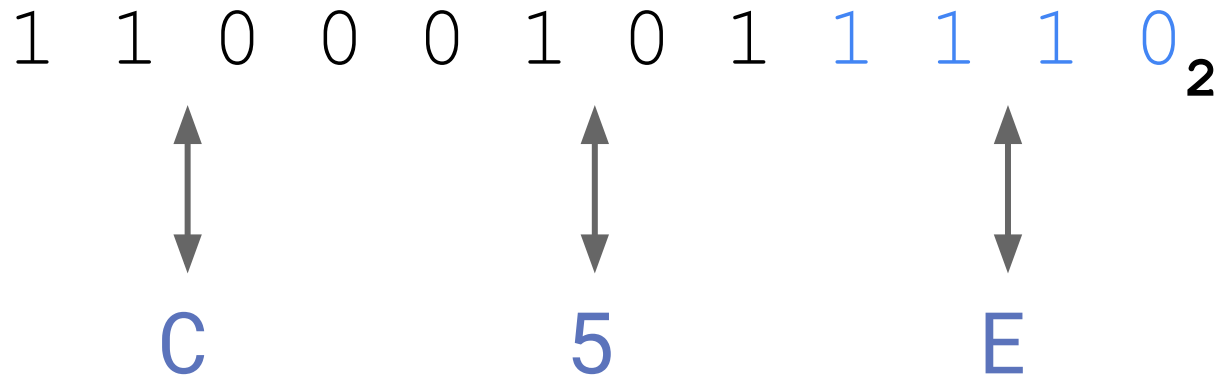
↕

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



Hex → Decimal

Expansion of Powers to get Decimal

$$A7_{16} = (Ax16^1) + (7x16^0)$$

Hex → Decimal

Expansion of Powers to get Decimal

$$A7_{16} = (Ax16^1) + (7x16^0)$$

$$A7_{16} = (10x16^1) + (7x16^0)$$

Do math in decimal
when converting to
decimal

Hex → Decimal

Expansion of Powers to get Decimal

$$A7_{16} = (Ax16^1) + (7x16^0)$$

$$A7_{16} = (10x16^1) + (7x16^0)$$

$$A7_{16} = 160 + 7$$

Do math in decimal
when converting to
decimal

Hex → Decimal

Expansion of Powers to get Decimal

$$A7_{16} = (A \times 16^1) + (7 \times 16^0)$$

$$A7_{16} = (10 \times 16^1) + (7 \times 16^0)$$

$$A7_{16} = 160 + 7$$

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

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

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

$16^2 = 256$ So Digit is 0

$$256 > 167$$

Decimal → Hex

Reduction of Powers to get Hexadecimal

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

$$16^1 = 16$$

$$10 \times 16 = 160$$

Ten 16 in 167

$$11 \times 16 = 176$$

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 in 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

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

Addition in Hex

Work col by col

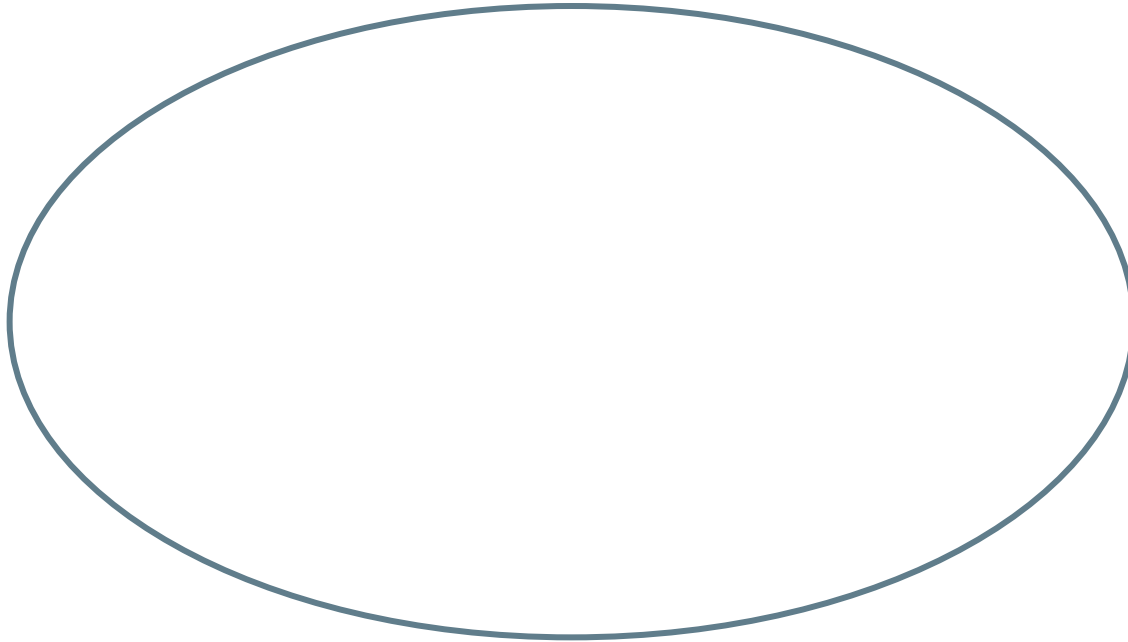
- convert to decimal
- perform the operation
- convert result to hex

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

Addition in Hex

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

Addition in Hex



$$\begin{array}{r} 7 \text{ A}_{16} \\ + \underline{5 \text{ E}_{16}} \\ \hline \end{array}$$

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

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

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

Addition in Hex

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

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

Addition in Hex

$$\begin{array}{r} A_{16} = 10_{10} \\ + E_{16} = \underline{14}_{10} \\ \hline 24 = \end{array}$$

Greater than
the base (16)

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

Addition in Hex

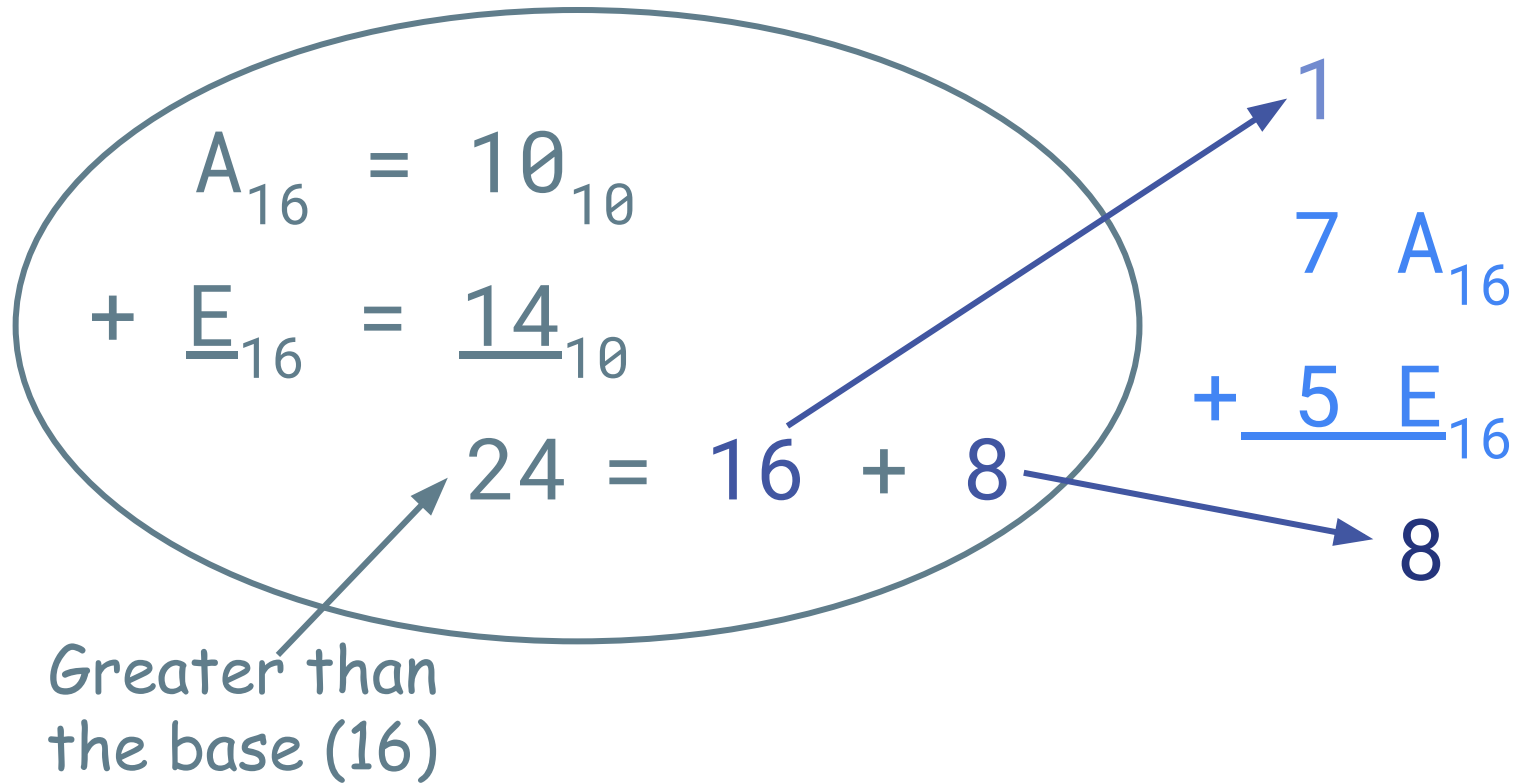
$$\begin{array}{r} A_{16} = 10_{10} \\ + E_{16} = \underline{14}_{10} \end{array}$$

$$24 = 16 + 8$$

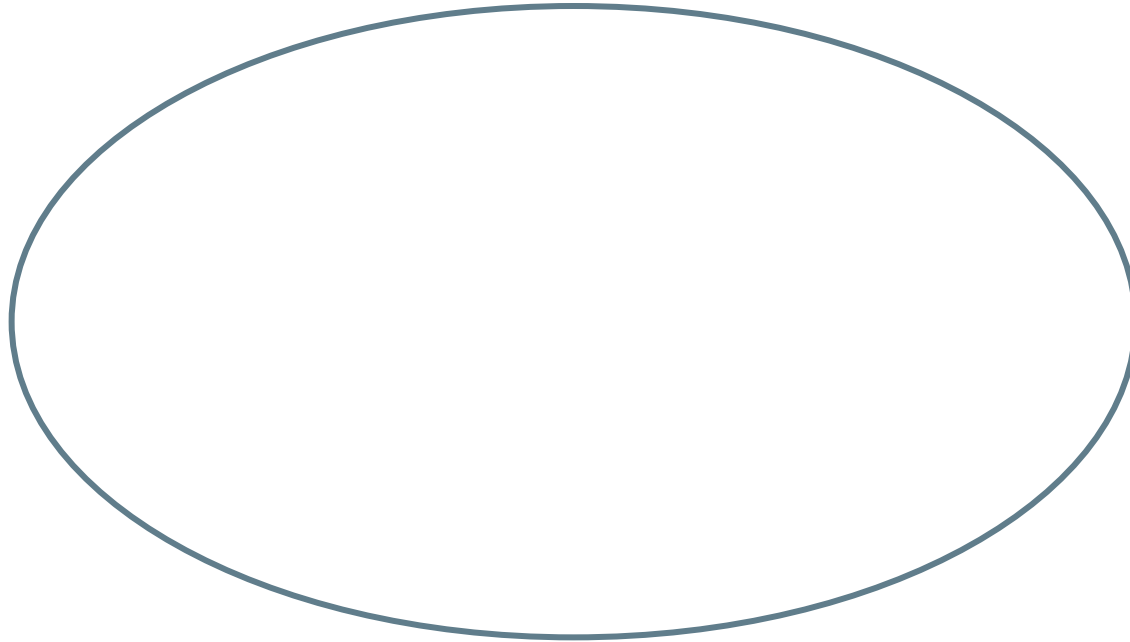
Greater than
the base (16)

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

Addition in Hex

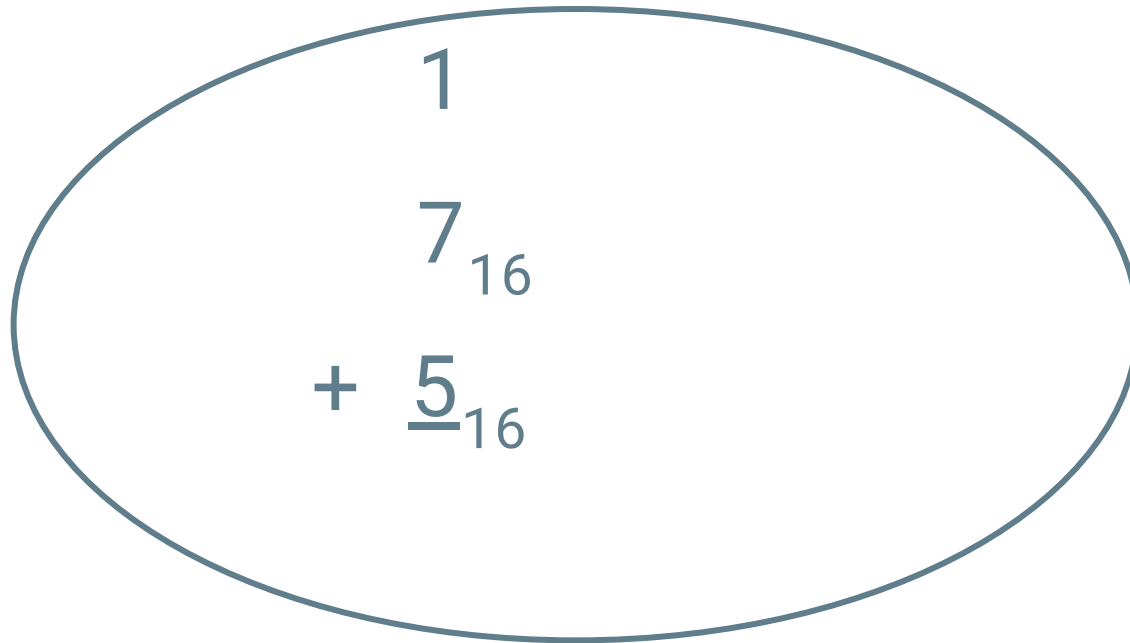


Addition in Hex



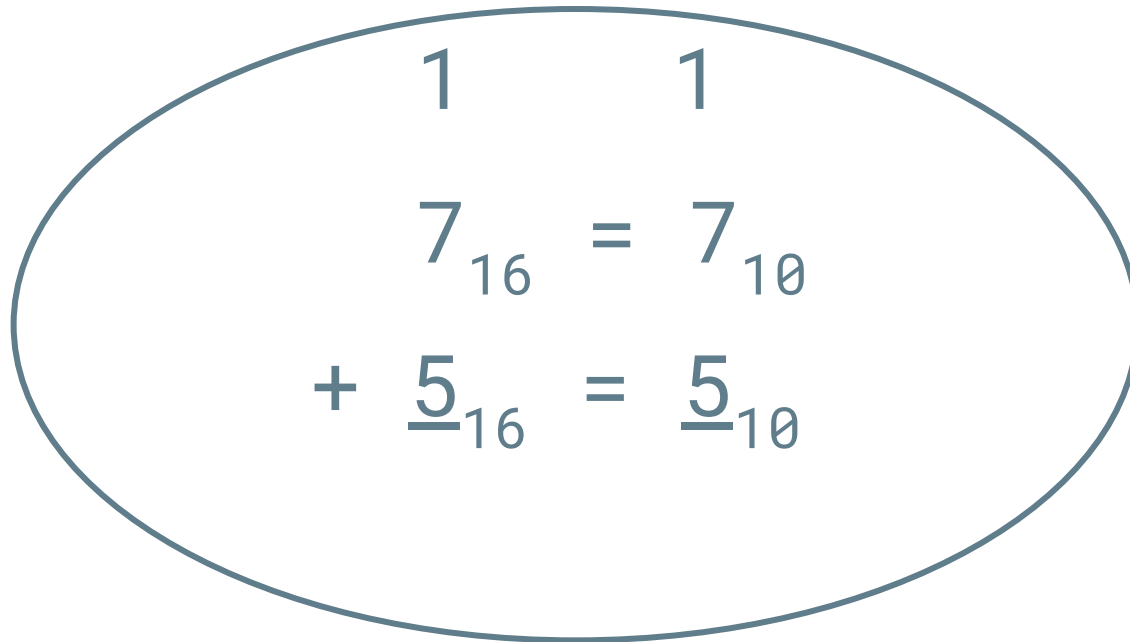
$$\begin{array}{r} 1 \\ 7 \text{ A}_{16} \\ + \underline{5 \text{ E}_{16}} \\ 8 \end{array}$$

Addition in Hex


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

$$\begin{array}{r} 1 \\ 7 \text{ A}_{16} \\ + \underline{5 \text{ E}_{16}} \\ 8 \end{array}$$

Addition in Hex



1 1

$$7_{16} = 7_{10}$$
$$+ \underline{5}_{16} = \underline{5}_{10}$$

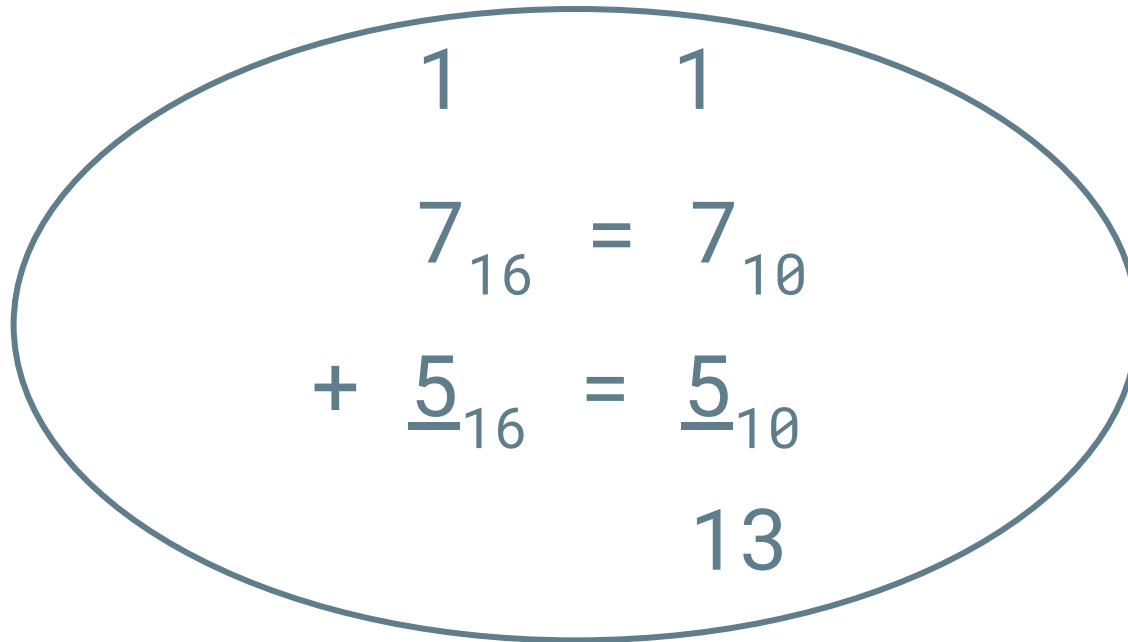
1

7 A₁₆

$$+ \underline{5 \ E}_{16}$$

8

Addition in Hex


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

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

Addition in Hex

Diagram illustrating the addition of two hexadecimal numbers, 17 and 5, resulting in 13. The result 13 is noted as being less than the base (16).

$$\begin{array}{r} 1 \quad 1 \\ 7_{16} = 7_{10} \\ + \quad 5_{16} = 5_{10} \\ \hline 13 \end{array}$$

Less than
the base (16)

Diagram illustrating the addition of two hexadecimal numbers, 1A and 5, resulting in 1F.

$$\begin{array}{r} 1 \quad A_{16} \\ + \quad 5 \quad E_{16} \\ \hline 1 \quad F \end{array}$$

Addition in Hex

The diagram illustrates the addition of two hexadecimal numbers, 17₁₆ and 15₁₆, resulting in 1D₁₆. The addition is shown in two columns. The first column contains 17₁₆ and 15₁₆, with a plus sign and an equals sign. The second column contains 17₁₆ and 15₁₆, with a plus sign and an equals sign. The result of the addition is 1D₁₆. A blue oval encloses the first column of the addition. An arrow points from the text "Less than the base (16)" to the number 13, which is the decimal equivalent of the sum 7 + 5. Another arrow points from the number 13 to the digit D in the result 1D₁₆.

$$\begin{array}{r} 1 \quad 1 \\ 7_{16} = 7_{10} \\ + \quad 5_{16} = 5_{10} \\ \hline 13 \end{array}$$

Less than the base (16)

$$\begin{array}{r} 1 \\ 7 \quad A_{16} \\ + \quad 5 \quad E_{16} \\ \hline D \quad 8 \end{array}$$

Addition in Hex

1

7 A₁₆

+ 5 E₁₆

D 8₁₆

Subtraction in Hex

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

Subtraction in Hex

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

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

Subtraction in Hex

$$\begin{array}{r} 8_{16} = 8_{10} \\ - E_{16} = \underline{14}_{10} \end{array} \quad \begin{array}{l} \text{Requires} \\ \text{redistribution} \end{array}$$

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

Subtraction in Hex

$$\begin{array}{r} \text{C } 10 \\ \text{D } 8_{16} \\ - \text{5 E}_{16} \\ \hline \end{array}$$

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

Subtraction in Hex

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

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

Subtraction in Hex

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

$$\begin{array}{r} C \quad 10 \\ D \quad 8_{16} \\ - \underline{5 \quad E}_{16} \end{array}$$

Subtraction in Hex

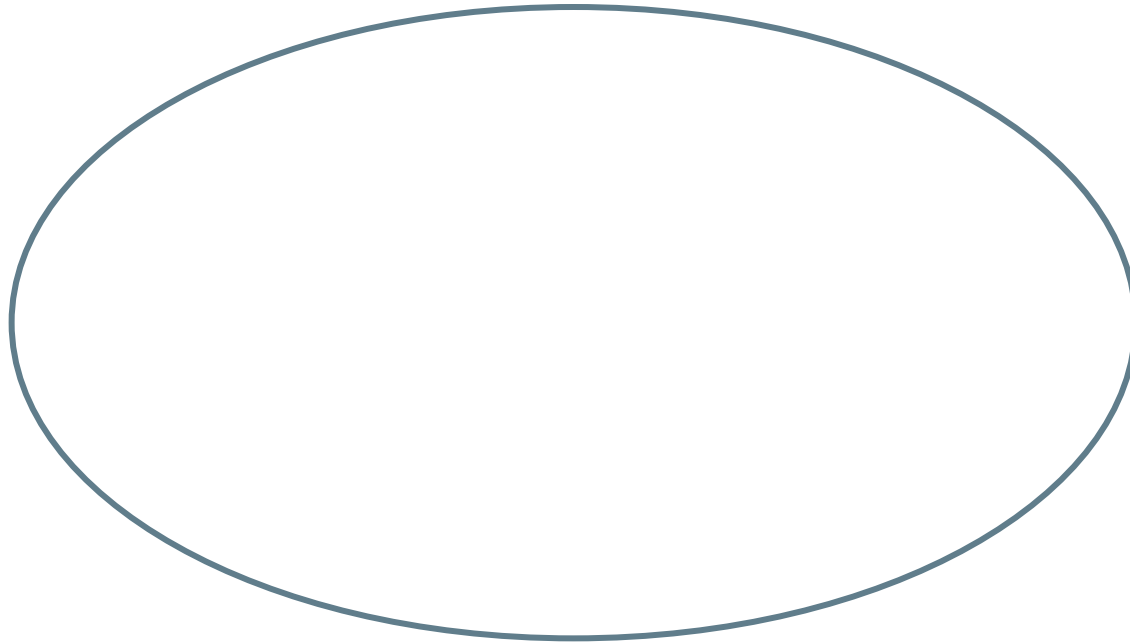
The diagram illustrates the subtraction of hexadecimal numbers. A large blue oval encloses the first two steps of the calculation. To the right, the third step is shown with a blue oval around the '10' in the minuend and a blue arrow pointing from this '10' to the letter 'A'.

$$\begin{array}{r} 10_{16} \\ + 8_{16} \\ - \underline{E_{16}} \\ \hline \end{array} = \begin{array}{r} 16_{10} \\ + 8_{10} \\ - \underline{14_{10}} \\ \hline \end{array}$$

10 → A

$$\begin{array}{r} C \ 10 \\ D \ 8_{16} \\ - \underline{5 \ E_{16}} \\ \hline \end{array}$$

Subtraction in Hex



$$\begin{array}{r} \text{C } 10 \\ \text{D } 8_{16} \\ - \text{5 E}_{16} \\ \hline \text{A} \end{array}$$

Subtraction in Hex

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

$$\begin{array}{r} C \quad 10 \\ D \quad 8_{16} \\ - 5 \quad E_{16} \\ \hline A \end{array}$$

Subtraction in Hex

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

$$\begin{array}{r} C \quad 10 \\ D \quad 8_{16} \\ - \underline{5 \quad E}_{16} \\ A \end{array}$$

Subtraction in Hex

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

$$\begin{array}{r} C \quad 10 \\ D \quad 8_{16} \\ - \underline{5 \quad E}_{16} \\ \hline 7 \quad A \end{array}$$

Subtraction in Hex

$$\begin{array}{r} \text{D } 8_{16} \\ - \text{5 E}_{16} \\ \hline \text{7 A} \end{array}$$

Concepts so far

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



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

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

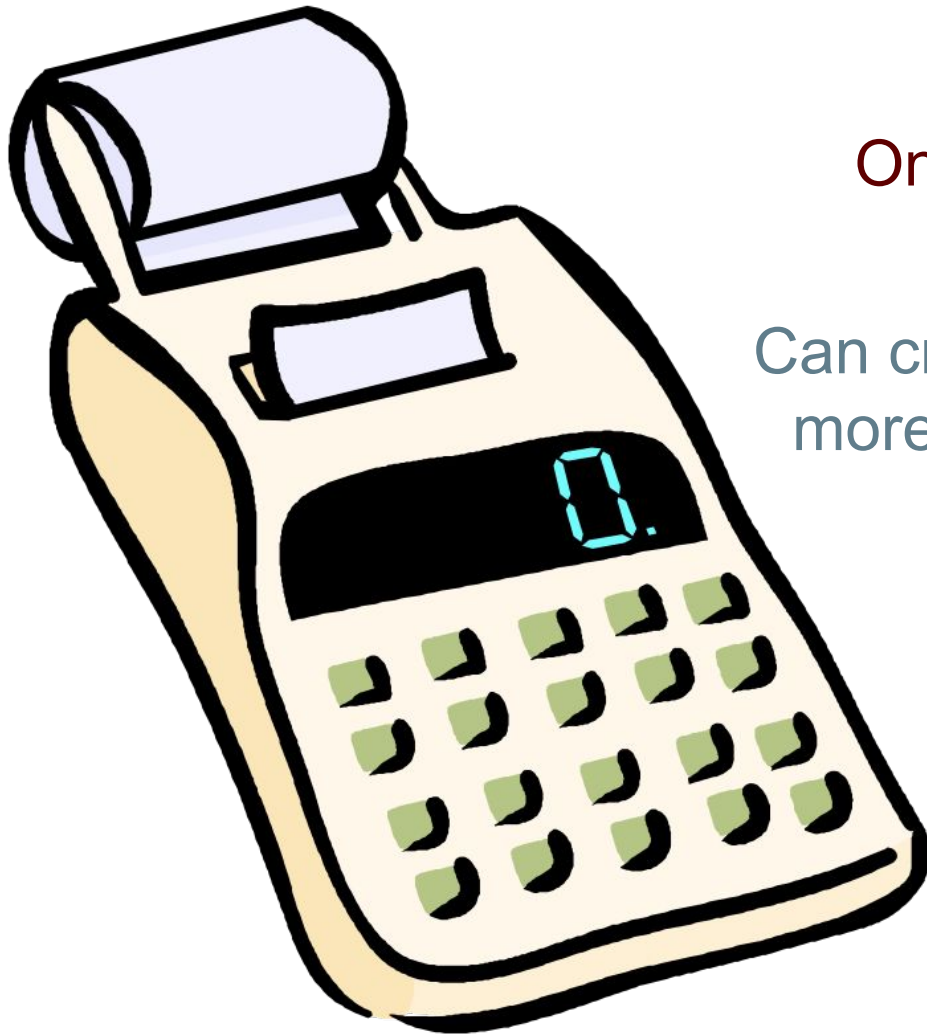
Digits are fixed by the Manufacturer



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

$$\begin{array}{r} 1101 \\ + 0100 \\ \hline \end{array}$$

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 1101 \\ + 0100 \\ \hline \end{array}$$

$$13 + 4$$

Desired result = 17

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 1101 \\ + 0100 \\ \hline \end{array}$$

$$13 + 4$$

Desired result = 17

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

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 1101 \\ + 0100 \\ \hline \end{array}$$

$$13 + 4$$

Desired result = 17

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

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

$$1 - 4$$

Desired result = -3

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 1101 \\ + 0100 \\ \hline \end{array}$$

$$13 + 4$$

Desired result = 17

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

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

$$1 - 4$$

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

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

Subtraction

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

Result
presented

Examples using 4 bit unsigned numbers

Addition

Subtraction

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

$$\begin{array}{r} 0001 \\ - 0100 \\ \hline \end{array}$$

Result
presented

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

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

Subtraction

$$\begin{array}{r} 1 \\ 011 \\ \cancel{1000}1 \\ - 0100 \\ \hline 1101 \end{array}$$

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

Subtraction

$$\begin{array}{r} 1 \\ 011 \\ \cancel{1000}1 \\ - 0100 \\ \hline 1101 \end{array}$$

Result presented

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

Subtraction

$$\begin{array}{r} 1 \\ 011 \\ 10001 \\ - 0100 \\ \hline 1101 \end{array}$$

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

Result
presented

Examples using 4 bit unsigned numbers

Addition

$$\begin{array}{r} 11 \\ 1101 \\ + 0100 \\ \hline 0001 \end{array}$$

Subtraction

$$\begin{array}{r} 1 \\ 011 \\ \cancel{10001} \\ - 0100 \\ \hline 1101 \end{array}$$