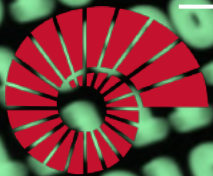


# COMP201

## Computer Systems & Programming

### Lecture #05 – Floating Point

---



**KOÇ  
UNIVERSITY**

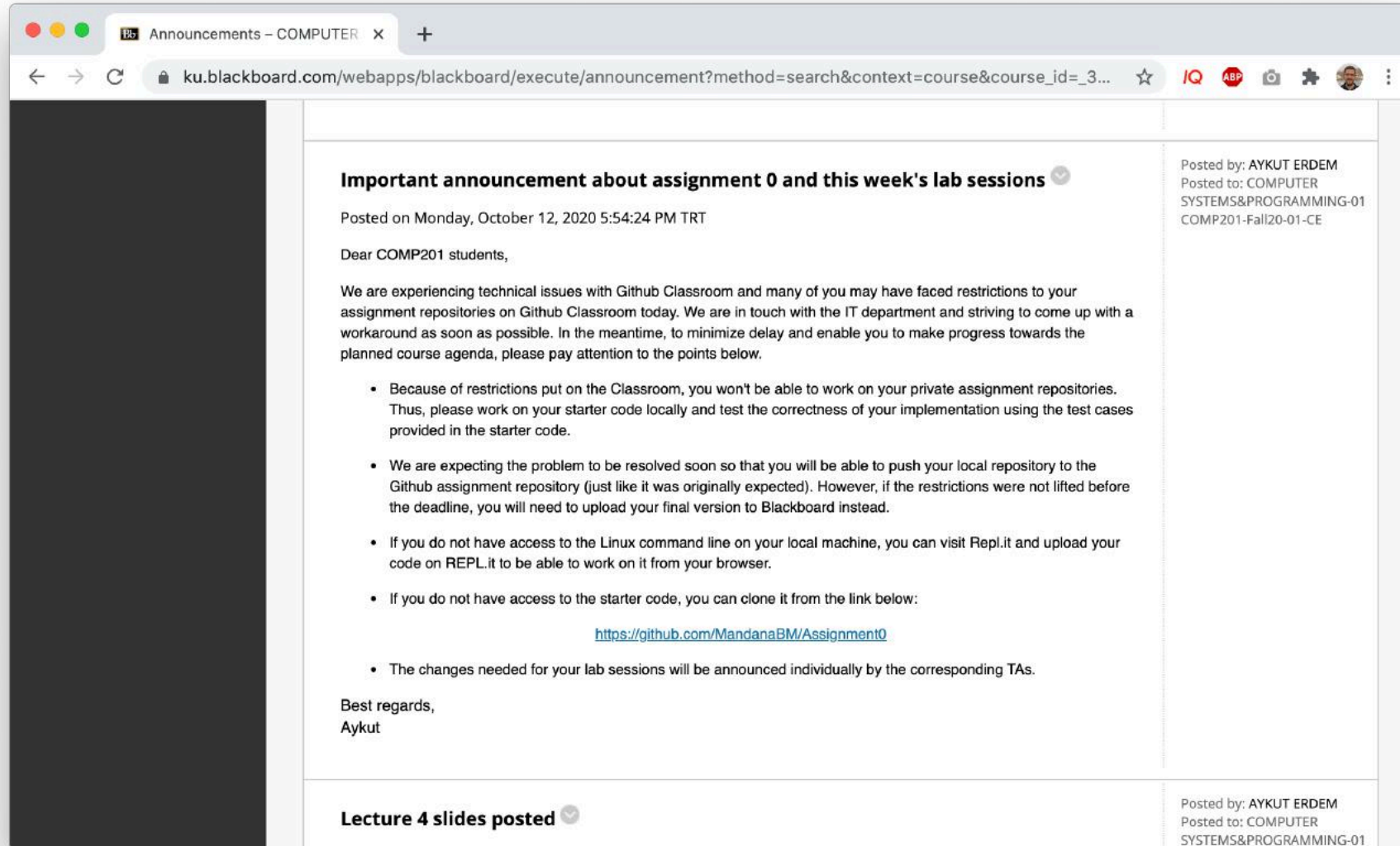
Aykut Erdem // Koç University // Fall 2020



# Recap: Bitwise Operators

- You're already familiar with many operators in C:
  - **Arithmetic operators:** +, -, \*, /, %
  - **Comparison operators:** ==, !=, <, >, <=, >=
  - **Logical Operators:** &&, ||, !
- **Bitwise operators:**
  - Logical operators: &, |, ~, ^,
  - Bit shift operators: <<, >>

# GitHub Classroom – What happened?



The screenshot shows a web browser window with the address bar displaying `ku.blackboard.com/webapps/blackboard/execute/announcement?method=search&context=course&course_id=_3...`. The page title is "Announcements – COMPUTER". The main content area features an announcement titled "Important announcement about assignment 0 and this week's lab sessions" with a checkmark icon. The announcement is dated "Monday, October 12, 2020 5:54:24 PM TRT" and is addressed to "COMP201 students". The text explains technical issues with GitHub Classroom and provides a list of instructions for students to work around these issues. The instructions include working locally, pushing to the GitHub repository, using Repl.it for Linux command line access, and cloning the starter code from a provided link. The announcement is signed "Best regards, Aykut". To the right of the announcement, the posting information is displayed: "Posted by: AYKUT ERDEM", "Posted to: COMPUTER SYSTEMS&PROGRAMMING-01", and "COMP201-Fall20-01-CE". Below the main announcement, there is a section titled "Lecture 4 slides posted" with a checkmark icon. To the right of this section, the posting information is displayed: "Posted by: AYKUT ERDEM", "Posted to: COMPUTER SYSTEMS&PROGRAMMING-01", and "COMP201-Fall20-01-CE".

**Important announcement about assignment 0 and this week's lab sessions** ✓

Posted on Monday, October 12, 2020 5:54:24 PM TRT

Dear COMP201 students,

We are experiencing technical issues with Github Classroom and many of you may have faced restrictions to your assignment repositories on Github Classroom today. We are in touch with the IT department and striving to come up with a workaround as soon as possible. In the meantime, to minimize delay and enable you to make progress towards the planned course agenda, please pay attention to the points below.

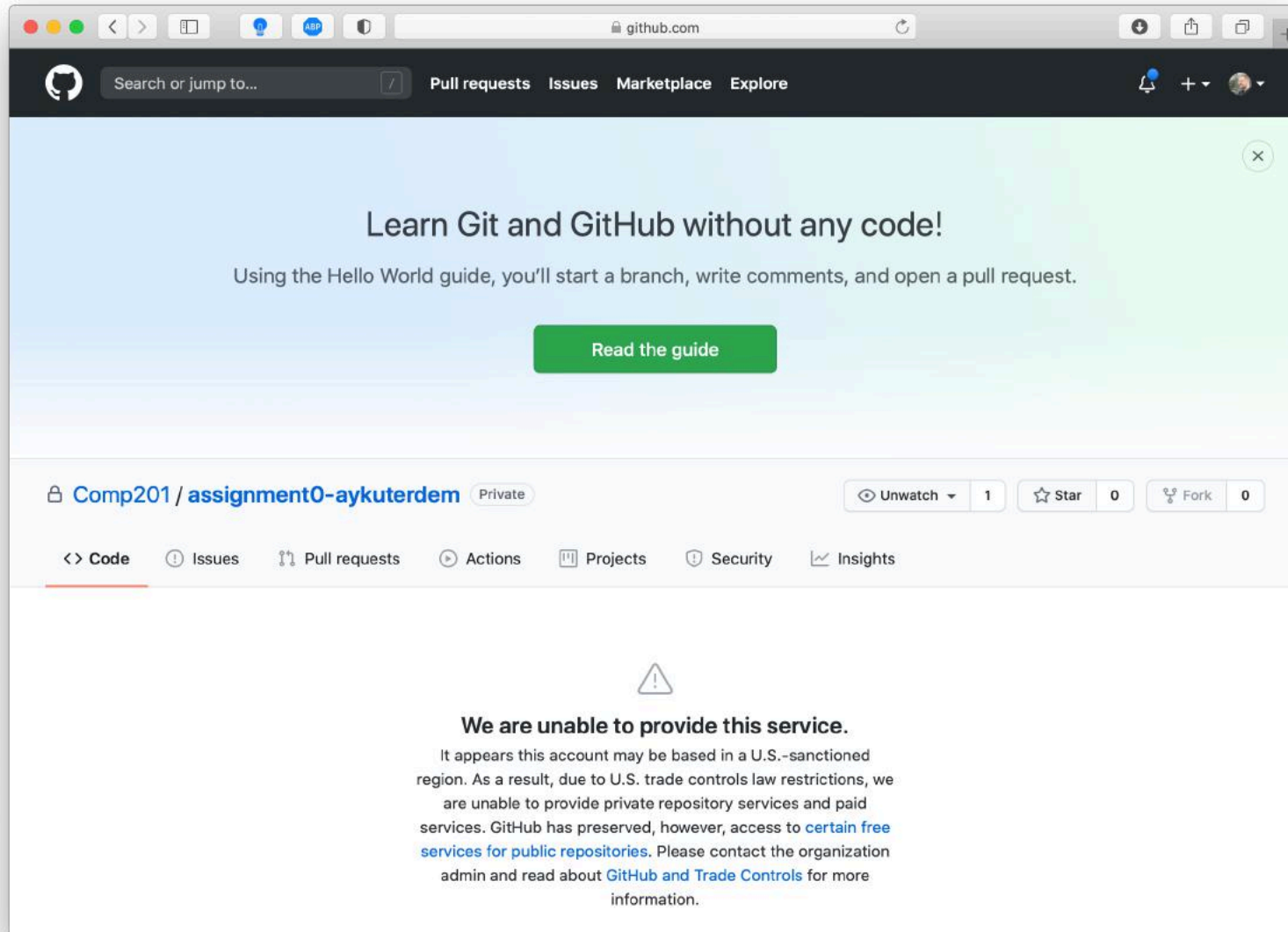
- Because of restrictions put on the Classroom, you won't be able to work on your private assignment repositories. Thus, please work on your starter code locally and test the correctness of your implementation using the test cases provided in the starter code.
- We are expecting the problem to be resolved soon so that you will be able to push your local repository to the Github assignment repository (just like it was originally expected). However, if the restrictions were not lifted before the deadline, you will need to upload your final version to Blackboard instead.
- If you do not have access to the Linux command line on your local machine, you can visit Repl.it and upload your code on REPL.it to be able to work on it from your browser.
- If you do not have access to the starter code, you can clone it from the link below:  
<https://github.com/MandanaBM/Assignment0>
- The changes needed for your lab sessions will be announced individually by the corresponding TAs.

Best regards,  
Aykut

**Lecture 4 slides posted** ✓

Posted by: AYKUT ERDEM  
Posted to: COMPUTER SYSTEMS&PROGRAMMING-01  
COMP201-Fall20-01-CE

# GitHub Classroom – What happened?



- Someone accessed GitHub Classroom repository from abroad.
- **Please use VPN if you're accessing GitHub from outside.**
- We will inform you about the new submission procedure soon.

# New Lab Sections Added to KUSIS!

The screenshot shows the KUSIS web interface for Koç University. The 'My Teaching Schedule' for Fall 2020 is displayed. A blue box highlights the newly added lab sections:

Class	Class Title	Enrolled	Days & Times	Room	Class Dates
COMP 201-01 (2449)	COMPUTER SYSTEMS&PROGRAMMING (Lecture)	142	MoWeFr 2:00PM - 2:50PM	TBA	Oct 5, 2020-Jan 8, 2021
COMP 201-LABA (2523)	COMPUTER SYSTEMS&PROGRAMMING (Laboratory)	33	Mo 7:00PM - 7:50PM	TBA	Oct 5, 2020-Jan 8, 2021
COMP 201-LABB (2524)	COMPUTER SYSTEMS&PROGRAMMING (Laboratory)	67	We 7:00PM - 7:50PM	TBA	Oct 5, 2020-Jan 8, 2021
COMP 201-LABC (2525)	COMPUTER SYSTEMS&PROGRAMMING (Laboratory)	42	Th 8:00PM - 8:50PM	TBA	Oct 5, 2020-Jan 8, 2021
COMP 201-LABD (3681)	COMPUTER SYSTEMS&PROGRAMMING (Laboratory)	0	Th 8:00PM - 8:50PM	TBA	Oct 5, 2020-Jan 8, 2021
COMP 201-LABE (3682)	COMPUTER SYSTEMS&PROGRAMMING (Laboratory)	0	We 7:00PM - 7:50PM	TBA	Oct 5, 2020-Jan 8, 2021
COMP 201-LABF (3683)	COMPUTER SYSTEMS&PROGRAMMING (Laboratory)	0	We 7:00PM - 7:50PM	TBA	Oct 5, 2020-Jan 8, 2021
TEAC 500-234 (3624)	TEACHING EXPERIENCE (Lecture)	0	TBA	TBA	Oct 5, 2020-Jan 8, 2021

- Based on the lab section assignments, we have now officially
  - 2 new lab sections on Wednesday (**LAB E-F**)
  - 1 new lab section on Thursday (**LAB D**)
- **Please change your section accordingly!**
  - This is important for setting up the participants for the Zoom meetings.



# New Lab Sections Added to KUSIS!

Lab A	Lab B	Lab C	Lab D	Lab E	Lab F	Lab G	Lab H	Lab I	Lab J	Lab K	Lab L	Lab M	Lab N	Lab O	Lab P
Time: Monday, 19.00 - 19.50	Time: Wednesday, 19.00 - 19.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Wednesday, 19.00 - 19.50	Time: Wednesday, 19.00 - 19.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50	Time: Thursday, 20.00 - 20.50
Teaching Assistant: Muhammad Aditya Sasongko	Teaching Assistant: Farzin Negahbani	Teaching Assistant: Mandana Bagheri Marzjarani	Teaching Assistant: Saman	Teaching Assistant: Amir Akhlaghi	Teaching Assistant: Ahmed Imam Shah	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman	Teaching Assistant: Saman
Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID	Student ID
69054 Barış Kaplan	69387 koray tecimer	72910 Atilla Özbek	72840 Begü	71455 Alkan Akısu	68881 Tolgay Dülger	72910 Atilla Özbek	72840 Begü	69282 Berke Can Rizai	60353 Mete UZ	68842 Kadir Danışman	68910 Yusu	64365 Arda Bakır	68624 Tunahan Öztürk	68824 İrmak	68845 Hasa
72959 Farrin Marouf Sofian	72290 Eren Yenigül	68842 Kadir Danışman	68910 Yusu	72849 Burak Yıldırım	69395 Arda Tiftikçi	68842 Kadir Danışman	68910 Yusu	68770 Eren Barış Bostancı	63940 Bulut Boru	64365 Arda Bakır	68624 Tunahan Öztürk	68824 İrmak	68624 Önder Akaçık	68845 Hasa	69529 Utku
69312 Ege Seçilmiş	64795 Kutluhan Palalıoğlu	64365 Arda Bakır	68624 Tunahan Öztürk	68770 Eren Barış Bostancı	69680 Oğuzhan Hatipoğlu	64365 Arda Bakır	68624 Tunahan Öztürk	68692 Emre Düzakin	68914 emirhan erel	68624 Tunahan Öztürk	68824 İrmak	68624 Önder Akaçık	68845 Hasa	69529 Utku	68671 Onur
72368 Sarp Göl	69721 Alp Sipahi	64626 IRMAK ILGAZ	68671 Onur	64641 Ayda Kanıl	54082 BERKAY HOPALI	64626 IRMAK ILGAZ	68671 Onur	68920 Kaan Ergökçen	69716 Atahan Efe Pekcelin	64388 Melis Oktayoğlu	69224 Alp Özaslan	63876 Alp Seyhun Canoğlu	63921 Berke Çağlar	68859 Doğa	60167 Ahmet
72269 Yiğit Yakar	68692 Emre Düzakin	64388 Melis Oktayoğlu	69224 Alp Özaslan	70192 Doruk Özer	69705 Onur Mavitaş	64388 Melis Oktayoğlu	69224 Alp Özaslan	53777 Can Arda Okçuoğlu	69337 Oya Suran	69224 Alp Özaslan	63876 Alp Seyhun Canoğlu	63921 Berke Çağlar	63921 Berke Çağlar	68859 Doğa	64220 Fulya
71456 Muzaffer Mert Akkan	69003 Koray Hafizoğlu	69224 Alp Özaslan	63876 Alp Seyhun Canoğlu	64240 Kemal Batuhan Başkaya	68262 Ardi Deniz Kıratlı	69224 Alp Özaslan	63876 Alp Seyhun Canoğlu	69003 Koray Hafizoğlu	69374 Atahan Tap	69224 Alp Özaslan	63876 Alp Seyhun Canoğlu	63921 Berke Çağlar	63921 Berke Çağlar	68859 Doğa	60167 Ahmet
71739 Efe Ertim	63923 Özgün Ozan Nacitarhan	63876 Alp Seyhun Canoğlu	63921 Berke Çağlar	71753 Duha Emir Ganioglu	68756 Mehmet Mert Bezirgan	63876 Alp Seyhun Canoğlu	63921 Berke Çağlar	69579 Lütfü Mustafa Kemal Ato	69574 Yağmur Akarken	64060 Bora Berke Sahin	60281 Umutcan Türkmen	60233 Canberk Eker	60233 Canberk Eker	68859 Doğa	60167 Ahmet
68631 Ahmet Talha Akgül	71956 Doğa Kukul	60281 Umutcan Türkmen	60233 Canberk Eker	64570 Doruk Örnekcı	68678 Ceyhan Aslan	60281 Umutcan Türkmen	60233 Canberk Eker	64570 Doruk Örnekcı	69222 Ufuk Özalp	64017 Elvin Altıntaş	64845 Gizem Kaya	71801 beyza gündoğan	71801 beyza gündoğan	64220 Fulya	64139 Batül
54521 İbrahim Durul Koca	64013 Melodi Ezgi Keskin	64845 Gizem Kaya	64220 Fulya	64763 Beyzanur Çoban	54306 Hazal Mengüaslan	64845 Gizem Kaya	64220 Fulya	64507 Talha Enes Güler	68835 Mehmet ÇUHADAR	72213 Kaan Turkmene	71868 Celal Kaplan	71469 Arda Aliz	71469 Arda Aliz	72049 Furki	60198 Yiğit
71766 Asu Tutku Gökçek	64763 Beyzanur Çoban	71868 Celal Kaplan	64139 Batül	69450 Mehmet Ukaş Uysal	68904 Pınar Erbil	71868 Celal Kaplan	64139 Batül	72414 Emir Şahin	72851 Damla Yıldız	72116 Baris Soke	72577 Andrew Bond	69584 Gizem Güneş	69584 Gizem Güneş	61253 Hasa	60610 Ertan
72414 Emir Şahin	68776 İrem Nur Bulut	72577 Andrew Bond	60198 Yiğit	65204 Ayşe Elif Satır	69498 Ege Berk Yıldırım	72577 Andrew Bond	60198 Yiğit	63982 Melih Özpelik	59721 Ekrem Yiğiter	69707 Çisem Özden	69584 Gizem Güneş	73075 Altun Hasanlı	73075 Altun Hasanlı	60610 Ertan	68168 Ezgi
63982 Melih Özpelik	69002 Ece Güz	73075 Altun Hasanlı	60610 Ertan	68648 Can Berk Alakır	59721 Ekrem Yiğiter	73075 Altun Hasanlı	60610 Ertan	60571 Başak çörtük	69101 Zeynep Sila Kaya	60035 Taluhan Öneş	73075 Altun Hasanlı	69388 Doğa Tekkaya	69388 Doğa Tekkaya	68168 Ezgi	59998 Ayca
69572 Deniz Erdoğan	63948 Mertcan Aşgün	69388 Doğa Tekkaya	68168 Ezgi	64586 Ogeday YILDIZ	69074 Meryem Karakaş	69388 Doğa Tekkaya	68168 Ezgi	50103 Melodi Kütük	64245 Ece Bahtışen	68669 Burak Genç	60768 Hüseyin Berk Kılıç	59871 İrem Sahin	60768 Hüseyin Berk Kılıç	59998 Ayca	71478 Mert
71715 Evrim Enüstün	64610 aras yılmaz	59871 İrem Sahin	71478 Mert		68147 Özcan Adnan Çobanoğlu	59871 İrem Sahin	71478 Mert		64509 Ahmet Kadir Zeybek						

Note that the Section IDs are not in alphabetical order!

# About Quiz 1 & 2

- Due to a series of unfortunate events, Quiz 1 hasn't been released yet.
- On Friday, you will take Quiz 1 and Quiz 2 together in one sitting.
  - 20 mins, between 14:35-14:55
- Topics covered: B&O 2.1-2.4
- There will be a mock-up quiz at the end of today's lecture (*will not be graded*)



**COMP201 Topic 2: How can a  
computer represent real numbers  
in addition to integer numbers?**



# Learning Goals

Understand the design and compromises of the floating point representation, including:

- Fixed point vs. floating point
- How a floating point number is represented in binary
- Issues with floating point imprecision
- Other potential pitfalls using floating point numbers in programs

# Plan For Today

- Representing real numbers
- Fixed Point
- Floating Point

**Disclaimer:** Slides for this lecture were borrowed from  
—Nick Troccoli's Stanford CS107 class

# Lecture Plan

- Representing real numbers
- Fixed Point
- Floating Point



# Real Numbers

- We previously discussed representing integer numbers using two's complement.
- However, this system does not represent real numbers such as  $3/5$  or  $0.25$ .
- How can we design a representation for real numbers?

# Real Numbers

**Problem:** unlike with the integer number line, where there are a finite number of values between two numbers, there are an *infinite* number of real number values between two numbers!

**Integers between 0 and 2:** 1

**Real Numbers Between 0 and 2:** 0.1, 0.01, 0.001, 0.0001, 0.00001,...

We need a fixed-width representation for real numbers. Therefore, by definition, *we will not be able to represent all numbers.*

# Real Numbers

**Problem:** every number base has un-representable real numbers.

**Base 10:**  $1/6_{10} = 0.16666666\dots_{10}$

**Base 2:**  $1/10_{10} = 0.000110011001100110011\dots_2$

Therefore, by representing in base 2, *we will not be able to represent all numbers*, even those we can exactly represent in base 10.



# Fixed Point

- **Idea:** Like in base 10, let's add binary decimal places to our existing number representation.

5 9 3 4 . 2 1 6

$10^3$

$10^2$

$10^1$

$10^0$

$10^{-1}$

$10^{-2}$

$10^{-3}$

1 0 1 1 . 0 1 1

$2^3$

$2^2$

$2^1$

$2^0$

$2^{-1}$

$2^{-2}$

$2^{-3}$

# Lecture Plan

- Representing real numbers
- Fixed Point
- Floating Point

# Fixed Point

- **Idea:** Like in base 10, let's add binary decimal places to our existing number representation.

1 0 1 1 . 0 1 1

8s    4s    2s    1s                    1/2s    1/4s    1/8s

- **Pros:** arithmetic is easy! And we know exactly how much precision we have.



# Fixed Point

- **Problem:** we have to fix where the decimal point is in our representation. What should we pick? This also fixes us to 1 place per bit.

. 0 1 1 0 0 1 1

$1/2s$   $1/4s$   $1/8s$  ...

1 0 1 1 0 . 1 1

$16s$   $8s$   $4s$   $2s$   $1s$   $1/2s$   $1/4s$

# Fixed Point

- **Problem:** we have to fix where the decimal point is in our representation. What should we pick? This also fixes us to 1 place per bit.

Base 10

Base 2

$$5.07E30 = 10 \underbrace{\dots\dots\dots}_{100 \text{ zeros}} 0.1$$

$$9.86E-32 = 0.0 \underbrace{\dots\dots\dots}_{100 \text{ zeros}} 01$$

To be able to store both these numbers using the same fixed point representation, the bitwidth of the type would need to be at least 207 bits wide!

# Let's Get Real

What would be nice to have in a real number representation?

- Represent widest range of numbers possible
- Flexible “floating” decimal point
- Represent scientific notation numbers, e.g.  $1.2 \times 10^6$
- Still be able to compare quickly
- Have more predictable over/under-flow behavior



# Lecture Plan

- Representing real numbers
- Fixed Point
- Floating Point

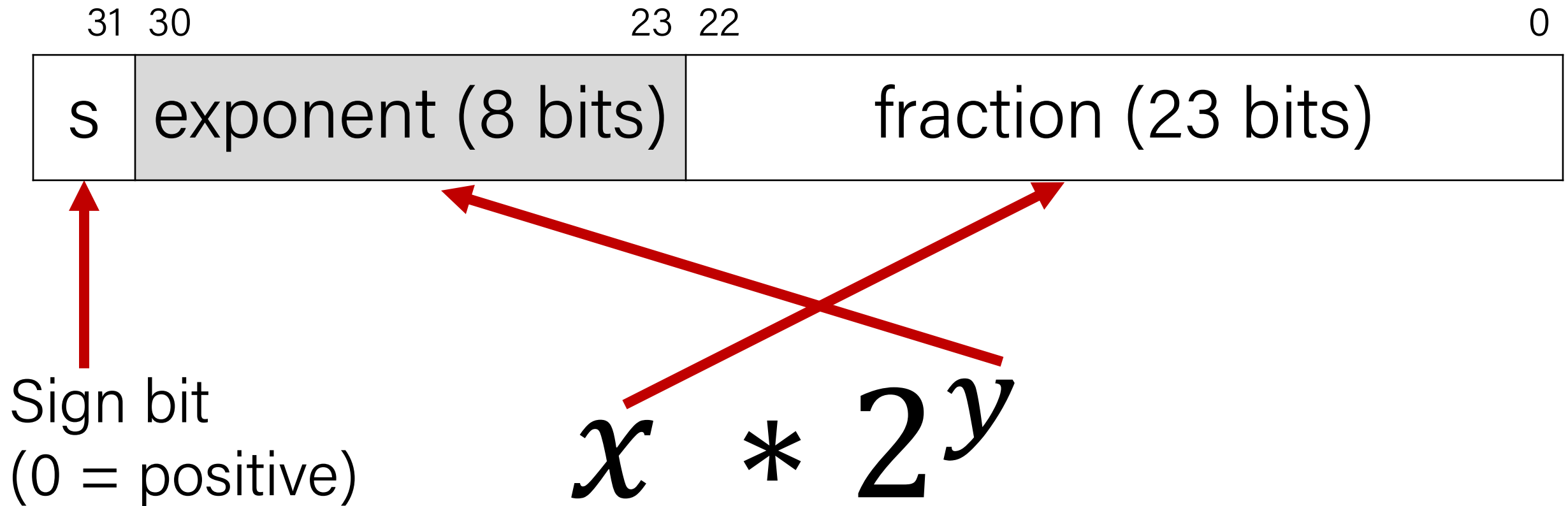
# IEEE Floating Point

Let's aim to represent numbers of the following scientific-notation-like format:

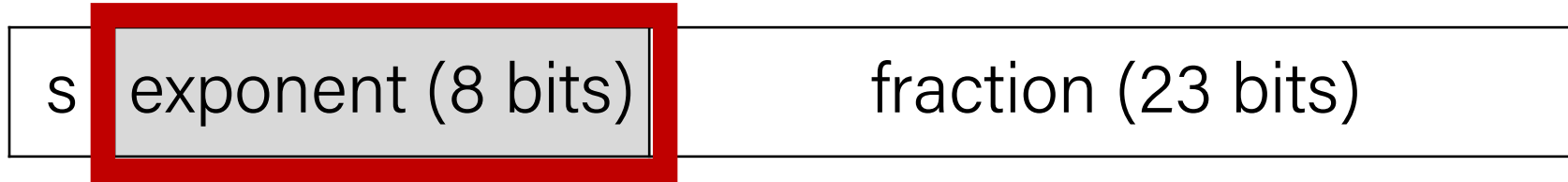
$$x * 2^y$$

With this format, 32-bit floats represent numbers in the range  $\sim 1.2 \times 10^{-38}$  to  $\sim 3.4 \times 10^{38}$ ! Is every number between those representable? **No.**

# IEEE Single Precision Floating Point



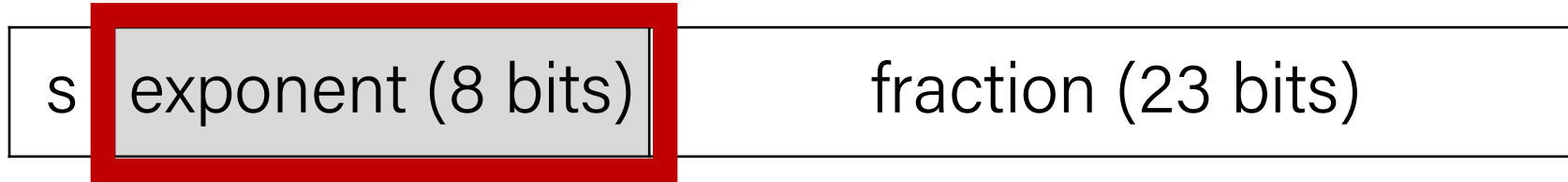
# Exponent



Exponent (Binary)	Exponent (Base 10)
11111111	?
11111110	?
11111101	?
11111100	?
...	?
00000011	?
00000010	?
00000001	?
00000000	?



# Exponent



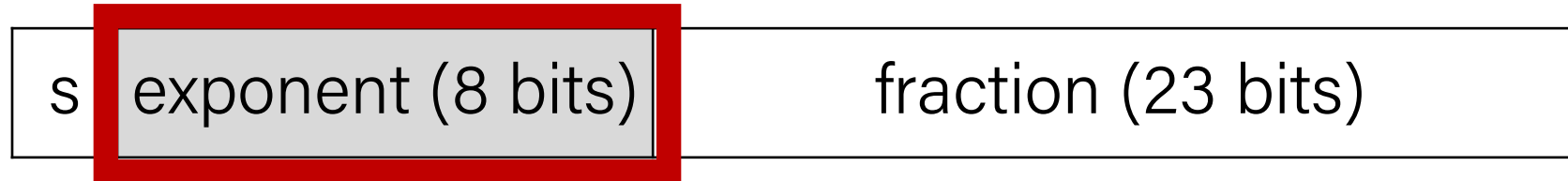
Exponent (Binary)	Exponent (Base 10)
11111111	RESERVED
11111110	?
11111101	?
11111100	?
...	?
00000011	?
00000010	?
00000001	?
00000000	RESERVED

# Exponent



Exponent (Binary)	Exponent (Base 10)
11111111	RESERVED
11111110	127
11111101	126
11111100	125
...	...
00000011	-124
00000010	-125
00000001	-126
00000000	RESERVED

# Exponent



- The exponent is **not** represented in two's complement.
- Instead, exponents are sequentially represented starting from 000...1 (most negative) to 111...10 (most positive). This makes bit-level comparison fast.
- **Actual value = binary value - 127 ("bias")**

11111110	$254 - 127 = 127$
11111101	$253 - 127 = 126$
...	...
00000010	$2 - 127 = -125$
00000001	$1 - 127 = -126$

# Fraction



$$x * 2^y$$

- We could just encode whatever  $x$  is in the fraction field. But there's a trick we can use to make the most out of the bits we have.

# An Interesting Observation

## In Base 10:

$$42.4 \times 10^5 = 4.24 \times 10^6$$

$$324.5 \times 10^5 = 3.245 \times 10^7$$

$$0.624 \times 10^5 = 6.24 \times 10^4$$

We tend to adjust the exponent until we get down to one place to the left of the decimal point.

## In Base 2:

$$10.1 \times 2^5 = 1.01 \times 2^6$$

$$1011.1 \times 2^5 = 1.0111 \times 2^8$$

$$0.110 \times 2^5 = 1.10 \times 2^4$$

**Observation:** in base 2, this means there is always a 1 to the left of the decimal point!

# Fraction



$$x * 2^y$$

- We can adjust this value to fit the format described previously. Then,  $x$  will always be in the format 1.XXXXXXXXXX...
- Therefore, in the fraction portion, we can encode just what is *to the right* of the decimal point! This means we get one more digit for precision.

**Value encoded = 1.[FRACTION BINARY DIGITS]**



# Practice

Sign	Exponent						Fraction			
0	0	...	0	0	0	1	0	1	0	...

Is this number:

**A) Greater than 0?**

**B) Less than 0?**

Is this number:

**A) Less than -1?**

**B) Between -1 and 1?**

**C) Greater than 1?**

# Skipping Numbers

- We said that it's not possible to represent *all* real numbers using a fixed-width representation. What does this look like?

## Float Converter

- <https://www.h-schmidt.net/FloatConverter/IEEE754.html>

## Floats and Graphics

- <https://www.shadertoy.com/view/4tVyDK>

# Let's Get Real

What would be nice to have in a real number representation?

- Represent widest range of numbers possible ✓
- Flexible “floating” decimal point ✓
- Represent scientific notation numbers, e.g.  $1.2 \times 10^6$  ?
- Still be able to compare quickly ✓
- Have more predictable over/under-flow behavior ?

# Representing Zero

The float representation of zero is all zeros (with any value for the sign bit)

Sign	Exponent	Fraction
any	All zeros	All zeros

- This means there are two representations for zero! ☹️

# Representing Small Numbers

If the exponent is all zeros, we switch into “denormalized” mode.

Sign	Exponent	Fraction
any	All zeros	Any

- We now treat the exponent as -126, and the fraction as *without* the leading 1.
- This allows us to represent the smallest numbers as precisely as possible.

# Representing Exceptional Values

If the exponent is all ones, and the fraction is all zeros, we have  $\pm$  infinity.

Sign	Exponent	Fraction
any	All ones	All zeros

- The sign bit indicates whether it is positive or negative infinity.
- Floats have built-in handling of over/underflow!
  - Infinity + anything = infinity
  - Negative infinity + negative anything = negative infinity
  - Etc.



# Representing Exceptional Values

If the exponent is all ones, and the fraction is nonzero, we have  
**Not a Number (NaN)**

Sign	Exponent						Fraction
any	1	...	...	...	...	1	Any nonzero

- NaN results from computations that produce an invalid mathematical result.
  - Sqrt(negative)
  - Infinity / infinity
  - Infinity + -infinity
  - Etc.

# Number Ranges

- 32-bit integer (type **int**):
  - › -2,147,483,648 to 2147483647
  - › Every integer in that range can be represented
- 64-bit integer (type **long**):
  - › -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
- 32-bit floating point (type **float**):
  - $\sim 1.2 \times 10^{-38}$  to  $\sim 3.4 \times 10^{38}$
  - Not all numbers in the range can be represented (not even all integers in the range can be represented!)
  - Gaps can get quite large! (larger the exponent, larger the gap between successive fraction values)
- 64-bit floating point (type **double**):
  - $\sim 2.2 \times 10^{-308}$  to  $\sim 1.8 \times 10^{308}$

# Precision options

- Single precision: 32 bits



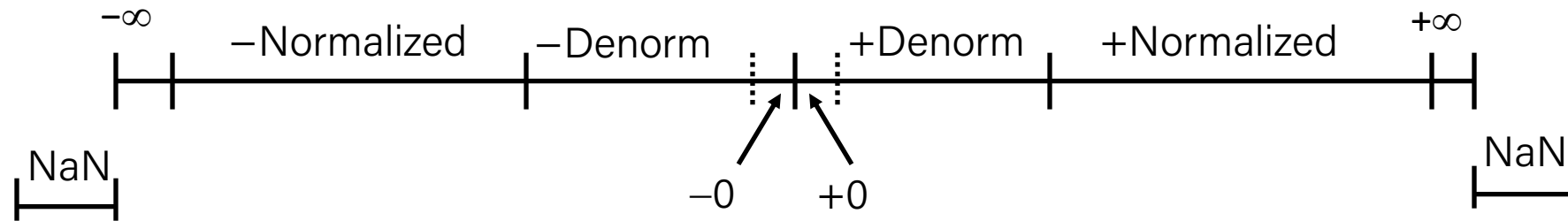
- Double precision: 64 bits



- Extended precision: 80 bits (Intel only)



# Visualization: Floating Point Encodings



# Recap

- Representing real numbers
- Fixed Point
- Floating Point

**Next time:** *More on floating points. How can a computer perform arithmetic operating on floating points?*

# Additional Reading

## What Every Computer Scientist Should Know About Floating-Point Arithmetic

DAVID GOLDBERG

*Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304*

Floating-point arithmetic is considered an esoteric subject by many people. This is rather surprising, because floating-point is ubiquitous in computer systems: Almost every language has a floating-point datatype; computers from PCs to supercomputers have floating-point accelerators; most compilers will be called upon to compile floating-point algorithms from time to time; and virtually every operating system must respond to floating-point exceptions such as overflow. This paper presents a tutorial on the aspects of floating-point that have a direct impact on designers of computer systems. It begins with background on floating-point representation and rounding error, continues with a discussion of the IEEE floating-point standard, and concludes with examples of how computer system builders can better support floating point.

Categories and Subject Descriptors: (Primary) C.0 [Computer Systems Organization]: General—*instruction set design*; D.3.4 [Programming Languages]: Processors—*compilers, optimization*; G.1.0 [Numerical Analysis]: General—*computer arithmetic, error analysis, numerical algorithms* (Secondary) D.2.1 [Software Engineering]: Requirements/Specifications—*languages*; D.3.1 [Programming Languages]: Formal Definitions and Theory—*semantics* D.4.1 [Operating Systems]: Process Management—*synchronization*

General Terms: Algorithms, Design, Languages

Additional Key Words and Phrases: denormalized number, exception, floating-point, floating-point standard, gradual underflow, guard digit, NaN, overflow, relative error, rounding error, rounding mode, ulp, underflow

[What Every Computer Scientist Should Know About Floating-Point Arithmetic,](#)

David Goldberg, ACM Computing Surveys, 23(1), 1991