

86% Average	85%	71%	95%	95%	70%	75%	85%	60%	78%	81%	93%
86% Median	85%	94%	100%	100%	75%	100%	100%	73%	88%	100%	100%
11% STD	9%	40%	15%	8%	33%	44%	25%	30%	25%	37%	13%
102% Max	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
65% Min	75%	0%	50%	75%	0%	0%	0%	0%	25%	100%	67%



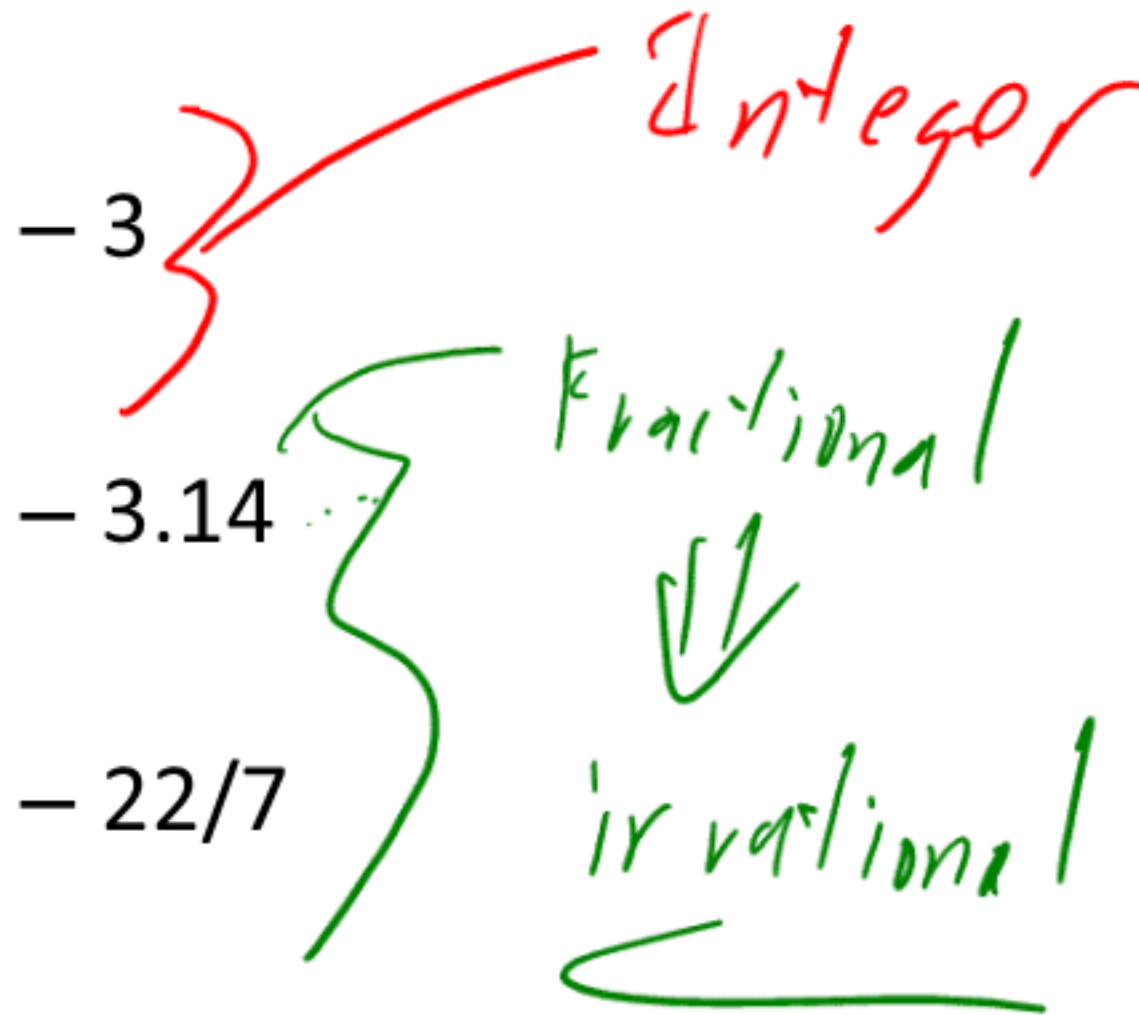
Floating Point Numbers

Lecture Objectives:

- 1) Define floating point number.
- 2) Define the terms fraction and exponent when dealing with floating point numbers.
- 3) Define overflow and underflow in relation to floating point numbers.
- 4) Convert a floating point number from binary to decimal format.
- 5) Convert a floating point number from decimal to floating point format.
- 6) Draw the algorithm for adding two floating point numbers.
- 7) Calculate the result of adding two floating point numbers together.

Numbers

- What is the difference between the following numbers



Definitions

- Scientific Notation
 - A notation which renders numbers with a single digit to the left of the decimal place
 - Normalized
 - A number in floating point notation that has no leading zeros
- Normalized*
- Not normalized*
- -2.34×10^{56}
 - $+0.002 \times 10^{-4}$
 - $+987.02 \times 10^9$
- Floating Point
 - Computer arithmetic that represents numbers in which the binary point is not fixed

Definitions

- Fraction 4
 - The value, between 0 and 1, placed in the fraction field
- Exponent
 - The value that is placed in the exponent field

→ negative and big #'s

Left part ranging
between 0 and 1

Floating Point

- In binary

$$- \pm 1.xxxxxxx_2 \times 2^{yyyy}$$

Power showing the scale.

- Types float and double in C

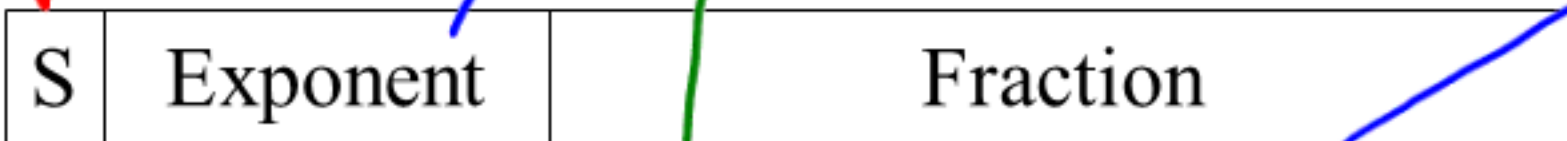
Floating Point Standard

- Defined by IEEE Std 754-1985
- Developed in response to divergence of representations
 - Portability issues for scientific code
- Now almost universally adopted
- Two representations
 - Single precision (32-bit)
 - Double precision (64-bit)

IEEE Floating Point Standard

32 bit #1
26 bit sign

single: 8 bits
double: 11 bits



23 bits
single: 23 bits
double: 52 bits

86 bits

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: sign bit (0 \Rightarrow non-negative, 1 \Rightarrow negative)
- Normalize significand: $1.0 \leq |\text{significand}| < 2.0$
 - Always has a leading pre-binary-point 1 bit, so no need to represent it explicitly (hidden bit)
 - Significand is Fraction with the "1." restored
- Exponent: excess representation: actual exponent + Bias
 - Ensures exponent is unsigned
 - Single: Bias = 127; Double: Bias = 1203

power



.5 \Rightarrow Sign bit \Rightarrow 0
positive \neq

Fraction \Rightarrow All zeros

Power/Exponent \Rightarrow -1

$$X = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exp} - \text{bias})}$$

$$1 \times (1 + 0) \times 2^{(126 - 127)} = 0.5$$

IEEE Floating Point Encodings

Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	\pm denormalized number
1–254	Anything	1–2046	Anything	\pm floating-point number
255	0	2047	0	\pm infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

Converting a number from Binary to Decimal Floating Point

S	Exponent	Fraction
0	7 6 5 4 3 2 1 0	1 0

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

$$(-1)^0 \times (1 + 17.5) \times 2^{(130 - 127)}$$

$$1 \times 1.5 \times 2^3$$

$$1 \times 1.5 \times 8$$

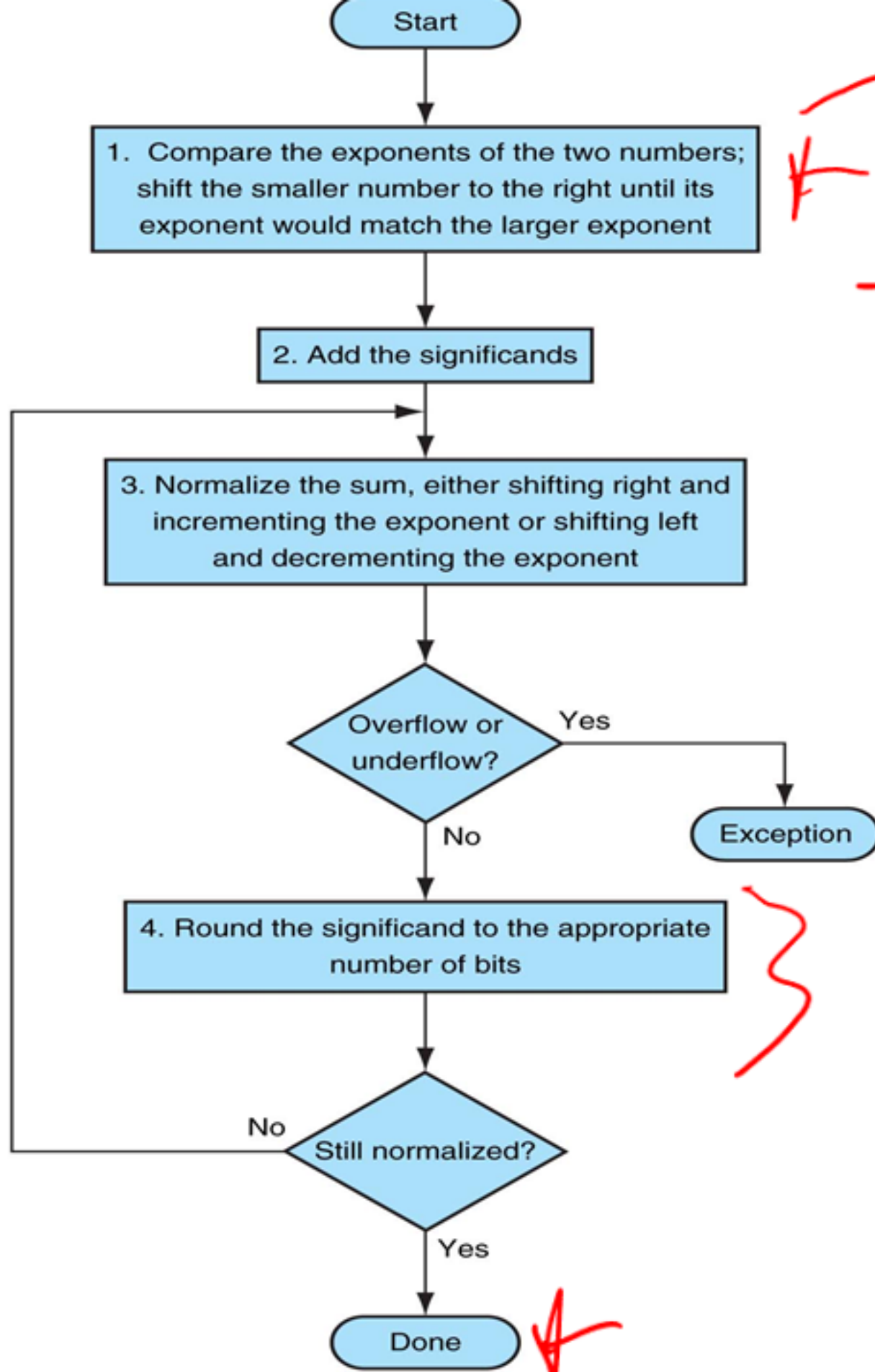
12

Ranges between ≤ 0 and ≤ 1



Scientific Notation Addition

Algorithm



Making the exponents the same.

How accurate is the number.

Example

- Consider a 4-digit decimal example

$$9.999 \times 10^1 + 1.610 \times 10^{-1}$$

$$9.999 \times 10^1 + .01610 \times 10^1$$

$$9.999 \times 10^1$$

$$+ .016 \times 10^1$$

$$\hline 10.015 \times 10^1$$



$$1.0015 \times 10^2$$

increased # of

$$1.002 \times 10^2$$

5 sig Figs.
BND



Binary Example

- $0.5 + -0.4375 \in \text{Not binary}$
 $- 1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ *is binary*

~~$$\begin{array}{r} 1.000_2 \times 2^{-1} \\ 0.1110_2 \times 2^{-1} \\ \hline 1.1110_2 \times 2^{-1} \end{array}$$~~

Binary Example

• $0.5 + -0.4375$

4 sig figs

$-1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$

$0.0000_2 \times 2^{-1}$

0.1110×2^{-1}

0.0010×2^{-1}

1.0×2^{-4}

2 sig figs

$1.000_2 \times 2^{-4} \Rightarrow$

1. Align binary points.

2. Added significant.

3. Normalize.

-0.0625



0.4375

$2^{-1} \Rightarrow .5$

.0111

$2^{-2} \Rightarrow .25$

$2^{-3} \Rightarrow .125$

$2^{-4} \Rightarrow .0625$

$2^{-5} \Rightarrow$

.4375
- .25

.1875
- .1250

.0625

.0625

0

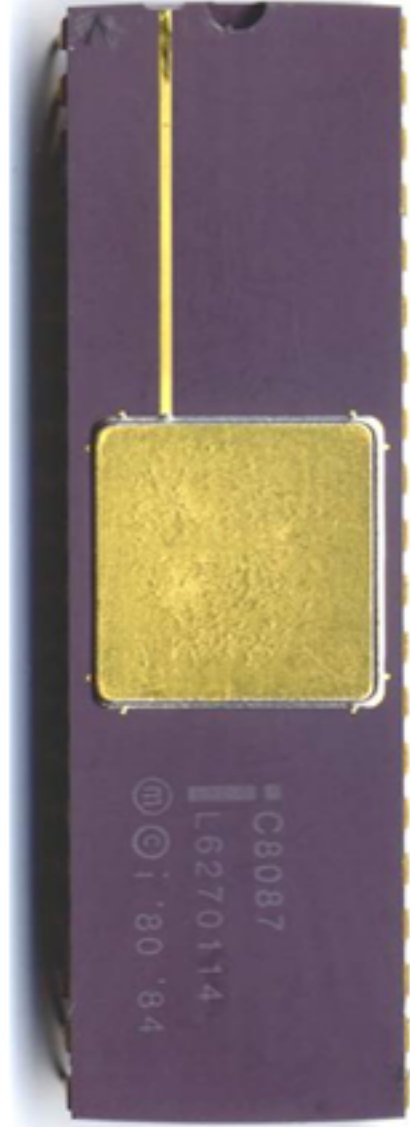
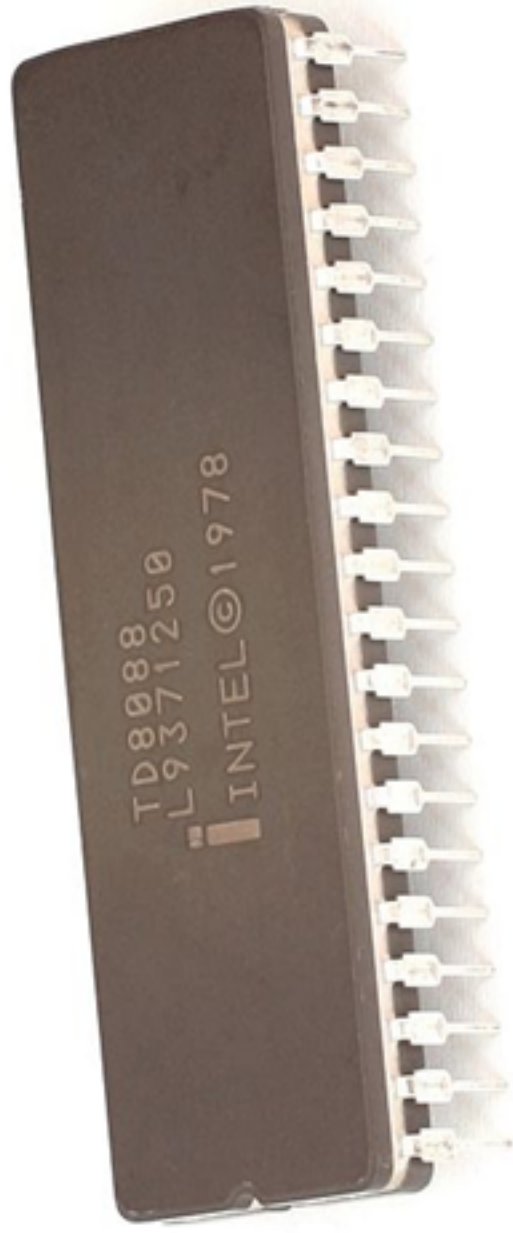
0111
↓ ↓ ↓ ↓ ↓
Normalize:

1.11 $\times 2^{-2}$

Floating-Point Addition

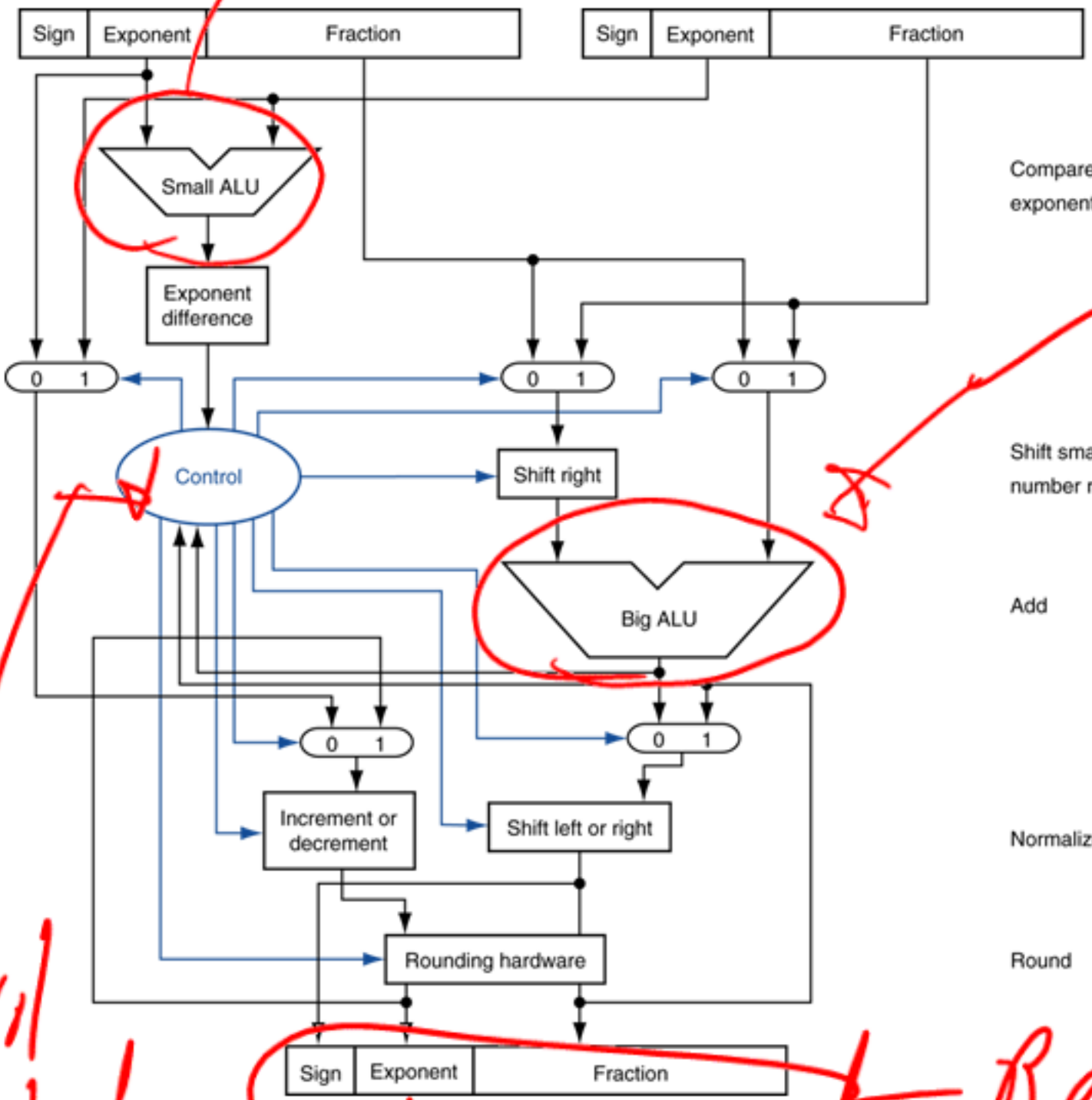
- Now consider a 4-digit binary example
 - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + -0.4375)
- 1. Align binary points
 - Shift number with smaller exponent
 - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- 2. Add significands
 - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- 3. Normalize result & check for over/underflow
 - $1.000_2 \times 2^{-4}$, with no over/underflow
- 4. Round and renormalize if necessary
 - $1.000_2 \times 2^{-4}$ (no change) = 0.0625

Floating Point Hardware



FP ALU

FP Adder Hardware



Adding the fraction

Step 1

Shift smaller number right

Step 2

Add

Step 3

Normalize

Step 4

Round

Control

Result





Floating Point Numbers Part 2

Lecture Objectives:

- 1) Draw the algorithm for multiplying two floating point numbers together.
- 2) Calculate the result of the multiplication of two floating point numbers.
- 3) Explain the ramifications of the pentium division bug.

Minute Quiz

- Take out a sheet of paper and complete the following:
 - Convert the following number to decimal from IEEE floating point

S	Exponent	Fraction
0	1 0 0 0 0 0 1 1	1 1 0

- Add the following two binary numbers from floating point
 - 1.001×2^{-1}
 - 1.100×2^{-2}

Converting a number from Binary to Decimal Floating Point

S	Exponent	Fraction
0	1 0 0 0 0 0 1 0	1 0

$$(-1)^S \times (1 + \textit{Fraction}) \times 2^{(\textit{Exponent} - \textit{Bias})}$$



Floating Point Multiplication

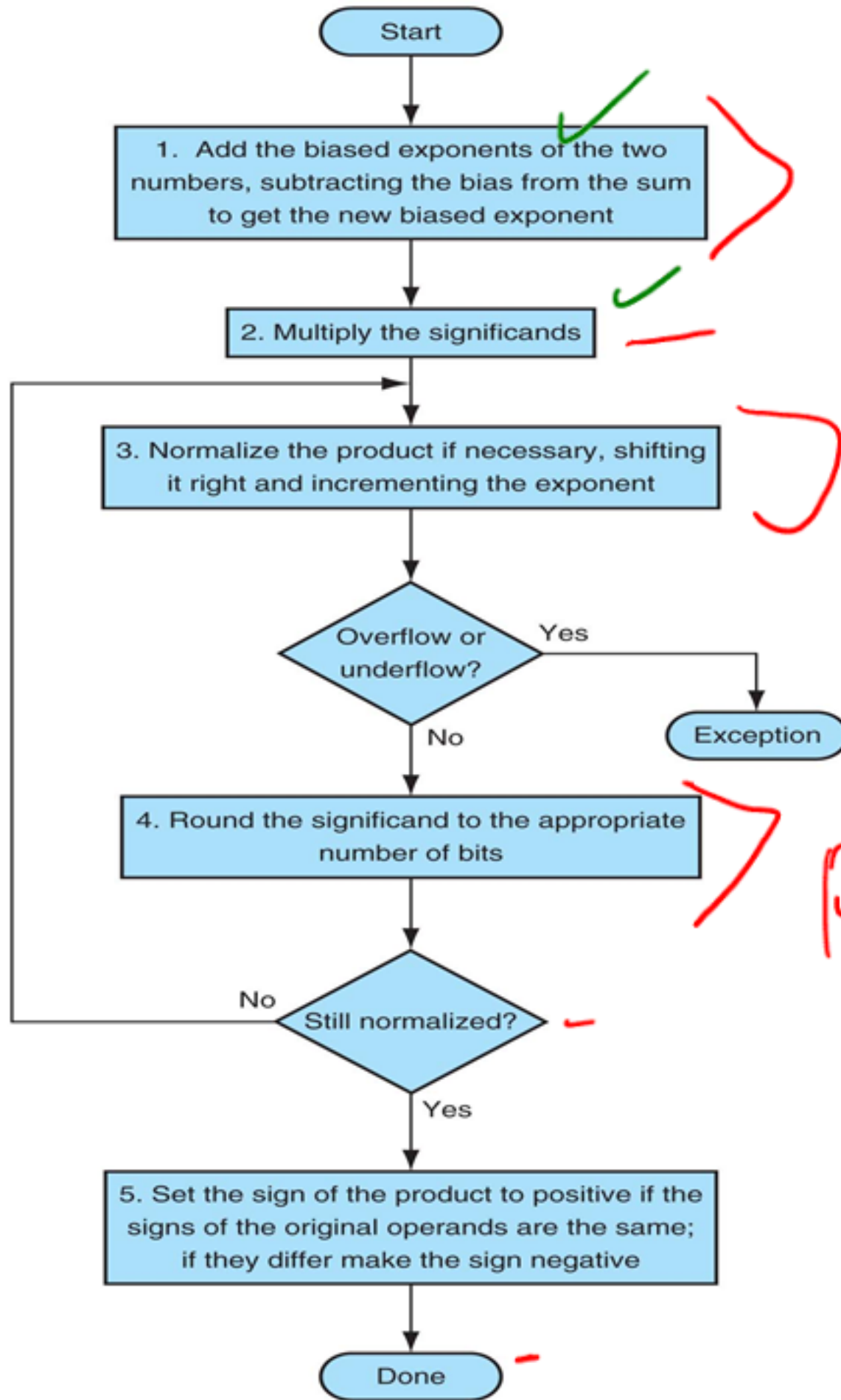


Figure out how close to the answer we are



Floating Point Multiplication Example

$$1.100_{two} \times 2^{-1}$$

$$\times 1.010_{two} \times 2^2$$

$$\begin{array}{r} 1.100 \\ \times 1.010 \\ \hline 0000 \\ 11000 \\ \hline 000000 \\ 1100000 \\ \hline 1.1110000 \end{array}$$

What are these bits?

$$11.111_6 \times 2^0$$

3.75

$$1100 \times 2^{-3} \Rightarrow 0.75$$

\downarrow \uparrow
 1 1
 15 25

~~$$1100 \times 2^{-3} \Rightarrow 0.75$$~~

$$.75$$

$$3.75$$

$$1010 \times 2^2$$

$$101,0 \times 2^0$$

$$5$$


$$4 \quad 1$$


FP Hardware

- FP multiplier is of similar complexity to FP adder
 - But uses a multiplier for significands instead of an adder
- FP arithmetic hardware usually does
 - Addition, subtraction, multiplication, division, reciprocal, square-root
 - FP \leftrightarrow integer conversion
- Operations usually takes several cycles
 - Can be pipelined

~~slow~~
"slow"

FP Instructions in MIPS

- FP hardware is coprocessor 1 
 - Adjunct processor that extends the ISA
- Separate FP registers
 - 32 single-precision: \$f0, \$f1, ... \$f31
 - Paired for double-precision: \$f0/\$f1, \$f2/\$f3, ...
 - Release 2 of MIPS ISA supports 32 × 64-bit FP reg's
- FP instructions operate only on FP registers
 - Programs generally don't do integer ops on FP data, or vice versa
 - More registers with minimal code-size impact
- FP load and store instructions
 - lwc1, ldc1, swc1, sdc1
 - e.g., ldc1 \$f8, 32(\$sp)

Coprocessor 1 

FP Instructions in MIPS

- Single-precision arithmetic
 - add.s, sub.s, mul.s, div.s
 - e.g., `add.s $f0, $f1, $f6`
- Double-precision arithmetic
 - add.d, sub.d, mul.d, div.d
 - e.g., `mul.d $f4, $f4, $f6`
- Single- and double-precision comparison
 - `c.xx.s`, `c.xx.d` (`xx` is `eq`, `lt`, `le`, ...)
 - Sets or clears FP condition-code bit
 - e.g. `c.lt.s $f3, $f4`
- Branch on FP condition code true or false
 - `bc1t`, `bc1f`
 - e.g., `bc1t TargetLabel`

S
↓
Single

precision
Double

precision

Lets work an example

- We want to write a program which will convert from degrees into Radians

$$\text{radians} = \text{degrees} \times \frac{\pi}{180}$$

Handwritten notes: $3.14\dots$ with an arrow pointing to π . The word "degrees" is underlined in red. The entire formula is circled in red.

Formula

floating point program

\Rightarrow MITS coprocessor
 \Rightarrow double precision.

Program Flowchart



Who Cares About FP Accuracy?

- Important for scientific code
 - But for everyday consumer use?
 - “My bank balance is out by 0.0002¢!” ☹
- The Intel Pentium FDIV bug
 - The market expects accuracy
 - See Colwell, *The Pentium Chronicles*

Pentium Floating point bug

BUSINESS
MARKETS • HIGH TECH • ECONOMY

PERSONAL TECHNOLOGY
By Walter S. Mossberg

Warning: Intel Inside
Microprocessors... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Intel to work on software patch for Pentium bug
Quick fix? Program will work around the... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Grove says I'm sorry about Pentium bug
Andrew S. Grove... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Flawed Chip Bruises Intel
Investors react, stock plunges... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Photo finish: Although Photoshop outdoes Picture Perfect, it has an image bug
Bug Dodge Boomed... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Flower
Pentium's... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Chip Short
Computer Giants' War Over Flow in Pentium Joins the PC Industry... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Who Is Twisting the Truth?
Intel Stands by Product As IBM Halls Suggest... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

TI: Be Totally Confused
Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Some Scientists Are Angry Over Flow in Pentium Chip, and Intel's Response
Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Intel's Pentium Problem Persists
Business... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

IBM to stop shipping Pentium PCs
Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Intel's Groove Aids Apology for Pentium Over the Internet
Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Big CEO, Defending Policy Not to Replace All Chips, Does a Sharp U-turn
Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

Pentium woes continue
Faulty FPU flubs math
Multithreading gets lost on P100 systems

INFO WORLD
The Voice of Personal Computing in the Enterprise

MARKETS
Dow Jones Industrial Average... Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

TECHNOLOGY
Intel... Pentium... bug... warning... inside... Intel... Pentium... bug... warning... inside...

