

CS 6210 Fall 2016

Bei Wang

Lecture 4

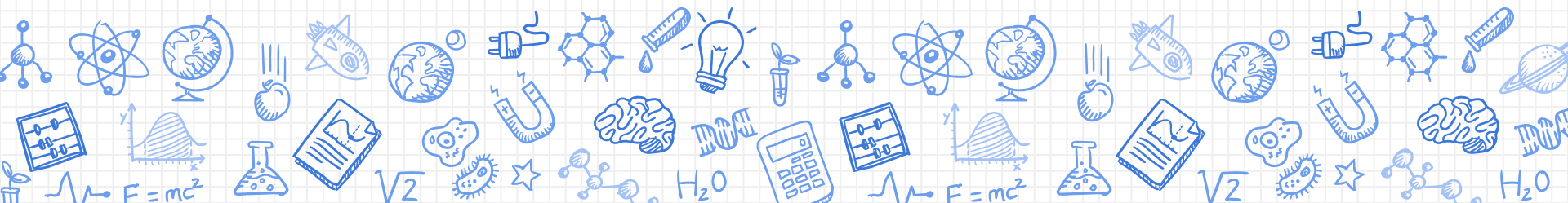
Floating Point Systems

Continued



The almost complete story on FP rounding

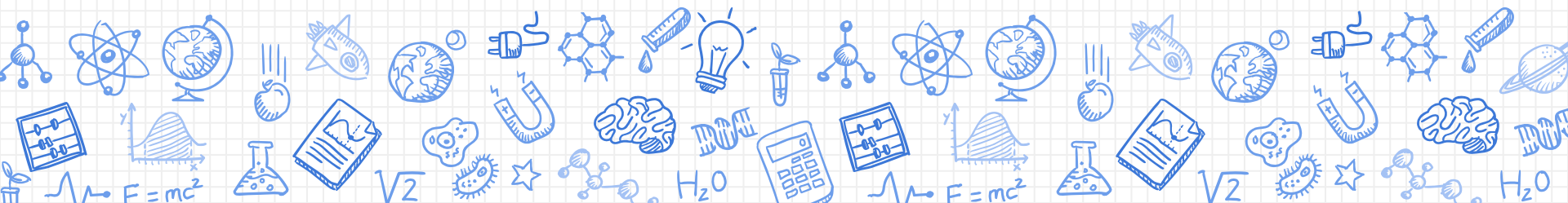
Decimal FP rounding (Integer rounding), Binary FP rounding
Possibly all you have to know about FP rounding
ON THE WHITEBOARD



Good (bad) coding practices

For FP arithmetic

<http://www.codeproject.com/Articles/29637/Five-Tips-for-Floating-Point-Programming>





Rule 1: Don't test for equality

```
double x;  
double y;  
...  
if (x == y) {...}
```

```
double x = 10;  
double y = sqrt(x);  
y *= y;  
if (x == y)  
    cout << "Square root is exact\n";  
else  
    cout << x-y << "\n";
```

-1.778636e-015.

```
double tolerance = ...  
if (fabs(x - y) < tolerance) {...}
```

Equality test likely to fail: FP numbers may not match in their last bits...

Solution: set a tolerance...



Rule 3: Use Logarithms to avoid overflow and underflow

How do you evaluate $200! / (190!10!)$?

Solution: consider $\log(xy) = \log x + \log y$, then $\log(n!) = \log(n) + \log(n-1) + \dots + \log(1)$

```
x = exp( logFactorial(200)
        - logFactorial(190)
        - logFactorial(10) );
```

```
double logFactorial(int n)
{
    double sum = 0.0;
    for (int i = 2; i <= n; ++i)
        sum += log((double)i);
    return sum;
}
```




Rule 5: SW specific rounding rules

MATLAB: $y = \text{round}(x)$: nearest integer; when tie, away from zero

MATLAB: $y = \text{convergent}(x)$: nearest integer; when tie, nearest even number

Mathematica: $y = \text{Round}[x]$: nearest integer; when tie, nearest even number

Python: $\text{round}(x, n)$: round to the nearest; when tie, round away from zero

```
>>> round(2.675, 2)
2.67
```

Why not 2.68?

```
2.67499999999999982236431605997495353221893310546875
```

Decimal to binary FP approximation with exact value closer to 2.67.

<https://docs.python.org/2/tutorial/floatingpoint.html>

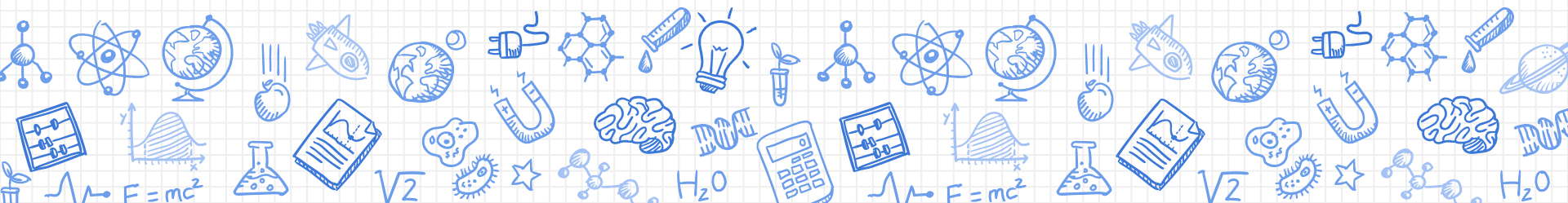
For your procrastination reading list

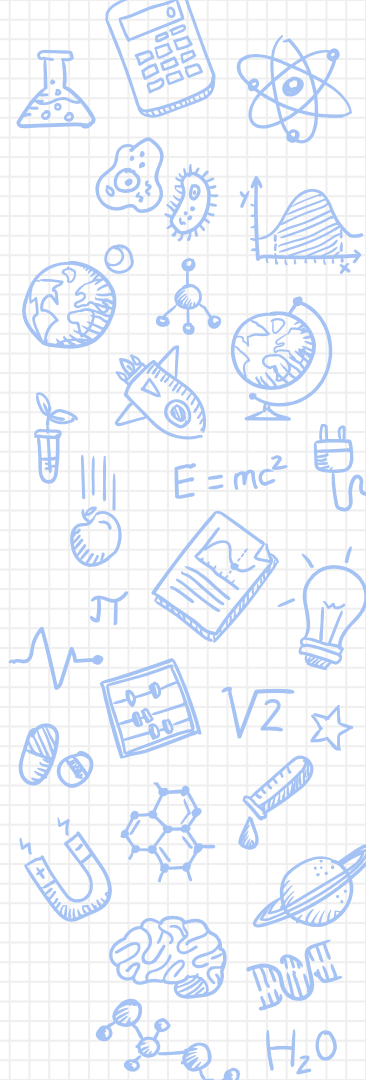
1. More details:
<http://www.codeproject.com/Articles/25294/Avoiding-Overflow-Underflow-and-Loss-of-Precision>
2. William D. (Bill) Young Lecture slides
<http://www.cs.utexas.edu/~byoung/cs429/slides4-fp.pdf>



The almost complete story on FP rounding

Decimal FP rounding (Integer rounding), Binary FP rounding
Possibly all you have to know about FP rounding
WHITEBOARD SUMMARY



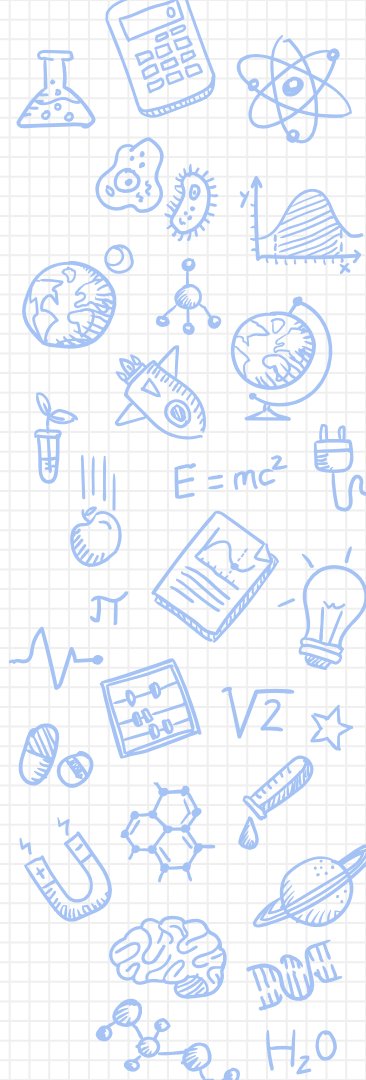


IEEE Standard for FP arithmetic (IEEE 754–1985, newer 2008)

Five rounding rules (1–3 directed rounding, 4–5 round to nearest)

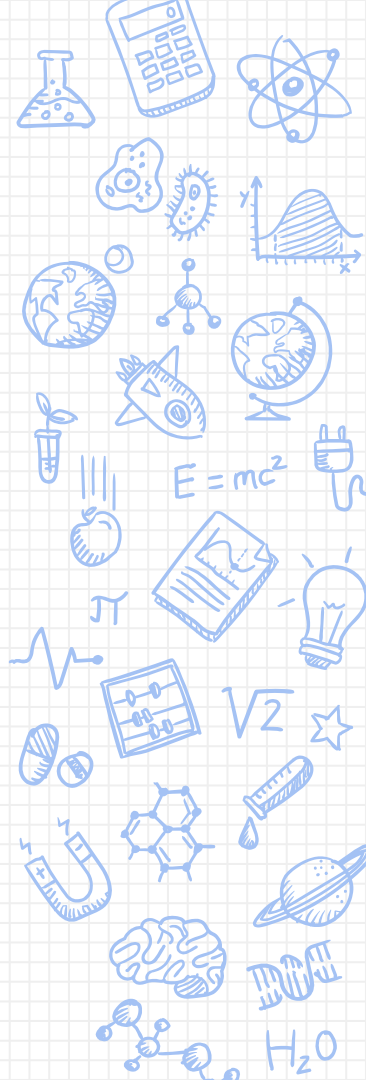
1. Round towards 0 “truncation”
2. Round towards +infinity “round up”
3. Round towards -infinity “round down”
4. Round to nearest, ties away from zero (typical for decimal FP)
5. **Round to nearest, ties to even** (default for binary FP, recommended for decimal FP): Round to the nearest value, if number falls **exactly** midway, it is rounded to the nearest value with an even (e.g. **zero for binary FP**) least significant bit.

Rule 5 prevents upwards or downwards bias in summing many #s that look like $x.5$, since round up and round down happen 50% of the time.



Example 1. Decimal FP round to integer

Rule	2.3	2.7	2.5	-2.3	-2.7	-2.5	3.5	-3.5
to 0	2	2	2	-2	-2	-2	3	-3
to +infinity	3	3	3	-2	-2	-2	4	-3
to -infinity	2	2	2	-3	-3	-3	3	-4
nearest, tie away from 0	2	3	3	-2	-3	-3	4	-4
nearest, tie to even	2	3	2	-2	-3	-2	4	-4



Example 2 decimal FP to nearest 100th (RULE 5)

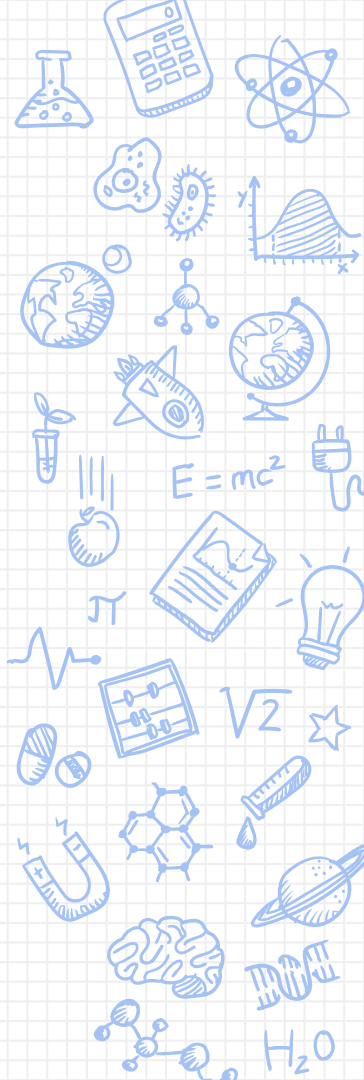
Decimal FP	Rounded Value	Rule
1.2349999	1.23	round down
1.2350001	1.24	round up
1.2350000	1.24	tie, round to even
1.2450000	1.24	tie, round to even

Theorems

Theorem 1: Using FP format with parameters b (base) and p (precision), counting differences using p digits, the relative error of the result can be as large as $b-1$.

Theorem 2: x and y are FP with base b and precision p , if subtraction is done with $p+1$ digits (i.e. 1 guard digit), then the relative error is upper bounded by 2 times machine precision.

<http://www.sci.utah.edu/~beiwang/teaching/cs6210/Goldberg.pdf>



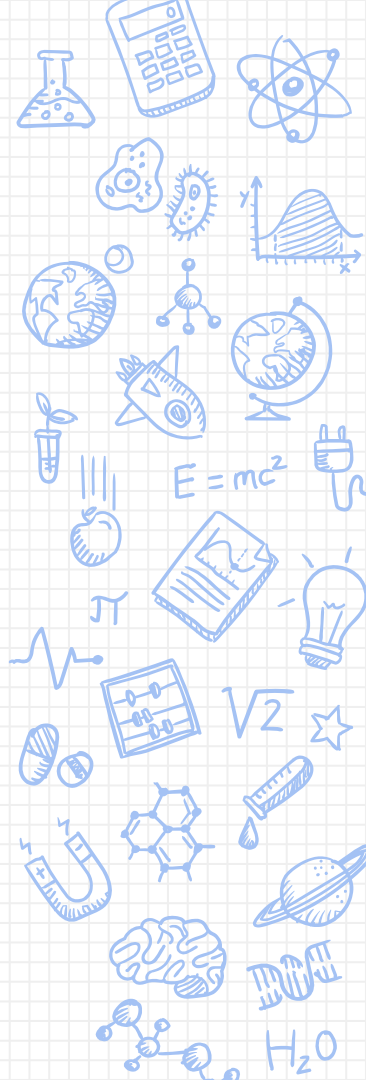
Deriving the Rounding Unit

Hint: using two FP numbers that are close to each other, with the same exponent e , they have a spacing of $b^{\{e-(p-1)\}}$.

Example:

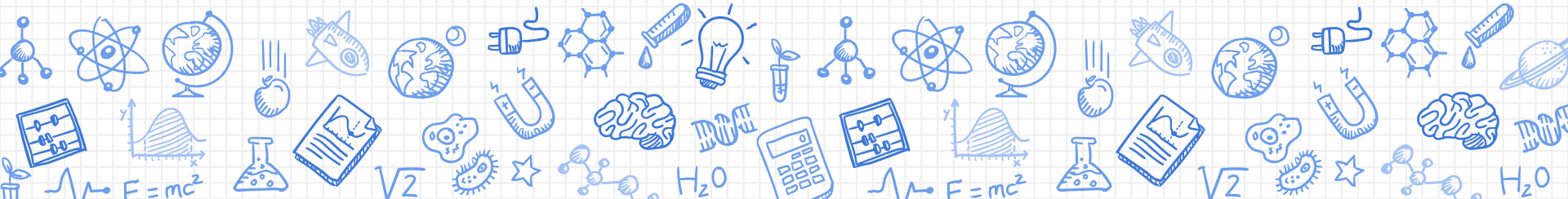
$b = 2, p = 1$, two adjacent FP numbers are 1×2^e and 0×2^e : spacing = 2^e

$b = 2, p = 2$, $1.0 \times 2^e, 1.1 \times 2^e$: spacing = $2^{\{e-1\}}$



General FP System

WHITEBOARD SUMMARY





General FP System IEEE Standard (b, p, L, U)

b : base; p : precision; $L \leq e \leq U$

Consider $b = 2$, common FP systems:

	name	exponent bit	fraction bit	L	U	exponent bias
binary16	half precision	5	11	-14	15	$2^4 - 1 = 15$
binary32	single	8	24	-126	127	$2^7 - 1 = 127$
binary64	double	11	53	-1022	1023	$2^{10} - 1 = 1023$
binary128	quadruple	15	113	-16382	16383	$2^{14} - 1 = 16383$

