

Achmad Imam Kistijantoro  
Anggrahita Bayu Sasmita  
Yudistira Dwi Wardhana Asnar  
Rahmat Mulyawan  
**Infall Syafalni**

# IF2130/II2130 – Organisasi dan Arsitektur Komputer

sumber: Greg Kesden, CMU 15-213, 2012

**Representasi Informasi – Floating Point Puzzles and Properties**

# Floating Point Puzzles

---

► For each of the following C expressions, either:

- Argue that it is true for all argument values
- Explain why not true

```
int x = ...;  
float f = ...;  
double d = ...;
```

Assume neither  
**d** nor **f** is NaN

- `x == (int)(float) x`
- `x == (int)(double) x`
- `f == (float)(double) f`
- `d == (float) d`
- `f == -(-f);`
- `2/3 == 2/3.0`
- `d < 0.0`  $\Rightarrow$  `((d*2) < 0.0)`
- `d > f`  $\Rightarrow$  `-f > -d`
- `d * d >= 0.0`
- `(d+f)-d == f`



# Interesting Numbers

{**single**, **double**}

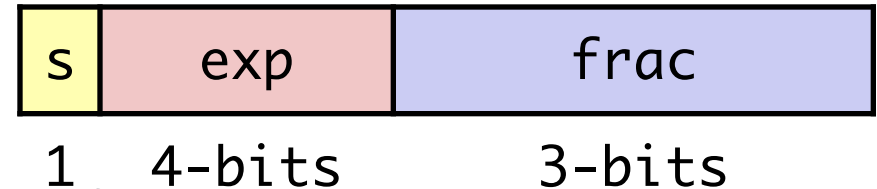
<i>Description</i>	<i>exp</i>	<i>frac</i>	<i>Numeric Value</i>
▶ Zero	00...00	00...00	0.0
▶ Smallest Pos. Denorm.	00...00	00...01	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$
▶ Single $\approx 1.4 \times 10^{-45}$			
▶ Double $\approx 4.9 \times 10^{-324}$			
▶ Largest Denormalized	00...00	11...11	$(1.0 - \epsilon) \times 2^{-\{126,1022\}}$
▶ Single $\approx 1.18 \times 10^{-38}$			
▶ Double $\approx 2.2 \times 10^{-308}$			
▶ Smallest Pos. Normalized	00...01	00...00	$1.0 \times 2^{-\{126,1022\}}$
▶ Just larger than largest denormalized			
▶ One	01...11	00...00	1.0
▶ Largest Normalized	11...10	11...11	$(2.0 - \epsilon) \times 2^{\{127,1023\}}$
▶ Single $\approx 3.4 \times 10^{38}$			
▶ Double $\approx 1.8 \times 10^{308}$			



# Creating Floating Point Number

## ► Steps

- Normalize to have leading 1
- Round to fit within fraction
- Postnormalize to deal with effects of rounding



## ► Case Study

- Convert 8-bit unsigned numbers to tiny floating point format

Example Numbers

128	10000000
15	00001111
17	00010001
19	00010011
33	00100001
35	00100011
138	10001010
63	00111111

# Rounding

1.BBG**RXXX**

Guard bit: LSB of result

Round bit: 1<sup>st</sup> bit removed

Sticky bit: OR of remaining bits

## ▶ Round up conditions

- ▶ Round = 1, Sticky = 1 → > 0.5
- ▶ Guard = 1, Round = 1, Sticky = 0 → Round to even

Value	Fraction	GRS	Incr?	Rounded
128	1.000 <b>0000</b>	000	N	1.000
15	1.111 <b>0000</b>	100	N	1.111
17	1.000 <b>1000</b>	010	N	1.000
19	1.001 <b>1000</b>	110	Y	1.010
138	1.000 <b>1010</b>	011	Y	1.001
▶ 63	1.111 <b>1100</b>	111	Y	10.000

# Mathematical Properties of FP Add

---

## ▶ Compare to those of Abelian Group

- ▶ Closed under addition?
  - ▶ But may generate infinity or NaN
- ▶ Commutative?
- ▶ Associative?
  - ▶ Overflow and inexactness of rounding
- ▶ 0 is additive identity?
- ▶ Every element has additive inverse
  - ▶ Except for infinities & NaNs

## ▶ Monotonicity

- ▶  $a \geq b \Rightarrow a+c \geq b+c$ ?
  - ▶ Except for infinities & NaNs



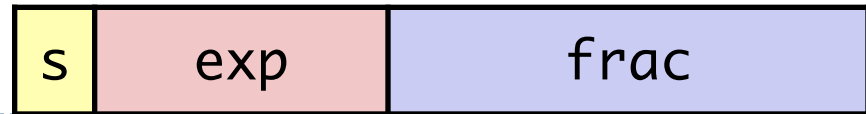
# Mathematical Properties of FP Mult

---

- ▶ **Compare to Commutative Ring**
  - ▶ Closed under multiplication?
    - ▶ But may generate infinity or NaN
  - ▶ Multiplication Commutative?
  - ▶ Multiplication is Associative?
    - ▶ Possibility of overflow, inexactness of rounding
  - ▶ 1 is multiplicative identity?
  - ▶ Multiplication distributes over addition?
    - ▶ Possibility of overflow, inexactness of rounding
- ▶ **Monotonicity**
  - ▶  $a \geq b \ \& \ c \geq 0 \Rightarrow a * c \geq b * c$ ?
    - ▶ Except for infinities & NaNs



# Normalize



1      4-bits      3-bits

## ► Requirement

- Set binary point so that numbers of form 1.xxxxx
- Adjust all to have leading one
  - Decrement exponent as shift left

Value	Binary	Fraction	Exponent
128	10000000	1.0000000	7
15	00001101	1.1010000	3
17	00010001	1.0001000	4
19	00010011	1.0011000	4
138	10001010	1.0001010	7
63	00111111	1.1111100	5





# Postnormalize

---

## ► Issue

- Rounding may have caused overflow
- Handle by shifting right once & incrementing exponent

Value	Rounded	Exp	Adjusted	Result
128	1.000	7		128
15	1.101	3		15
17	1.000	4		16
19	1.010	4		20
138	1.001	7		134
63	10.000	5	1.000/6	64



# End of Segment

---

