Will you light my candle?

Besides getting a 1-lecture relief from listening to Matt, what was the point of Monday's lecture?

Creating Floating Point Number

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

7	6 … 3	2 … 0
s	exp	frac

- **Case Study**
  - Convert 8-bit unsigned numbers to tiny floating point format
  - Example Numbers

128	10000000	**0 1110 000**
138	10001010	~~**0 1111 000**~~ →

128	1000 0000	1.000 x 2^7	1 x 2^7	128
138	1000 1010	1.001 x 2^7	(1+1/8) x 2^7	144
130	1000 0010	1.000 x 2^7		128
137	1000 0111	1.000 x 2^7		128
139	1000 1011	1.001 x 2^7		144
152	1001 1000	1.010 x 2^7	(1+1/4) x 2^7	160

2.84a (will look at b later)

k-bit exponent, n-bit fraction

A) V=7.0

E = 2, M=1.11

frac = 1100…0

Exp = 1000…01

bias = 2^(e-1)-1 = 2^(k-1)-1 = 0111…1

To represent E=0, need to have Exp=bias: 0111…1

To represent E=2, need to have Exp=bias+10 = 1000…1

FP Multiplication

$$(-1)^{s1}\ M1\ \ 2^{E1}\ \ \times\ \ (-1)^{s2}\ M2\ \ 2^{E2}$$

- **Exact Result:** $(-1)^{s}\ M\ \ 2^{E}$
  - Sign *s*: *s1 ^ s2*
  - Significand *M*: *M1 * M2*
  - Exponent *E*: *E1 + E2*

- **Fixing**
  - If $M \geq 2$, shift *M* right, increment *E*
  - If *E* out of range, overflow
  - Round *M* to fit `frac` precision

- **Implementation**
  - Biggest chore is multiplying significands

Mathematical Properties of FP Mult

- **Compare to Commutative Ring**
  - Closed under multiplication?
  - Multiplication Commutative?
  - Multiplication is Associative?
  - 1 is multiplicative identity?
  - Multiplication distributes over addition?
- **Monotonicity**
  - $a \geq b \ \& \ c \geq 0 \Rightarrow a * c \geq b * c$?

Mathematical Properties of FP Mult

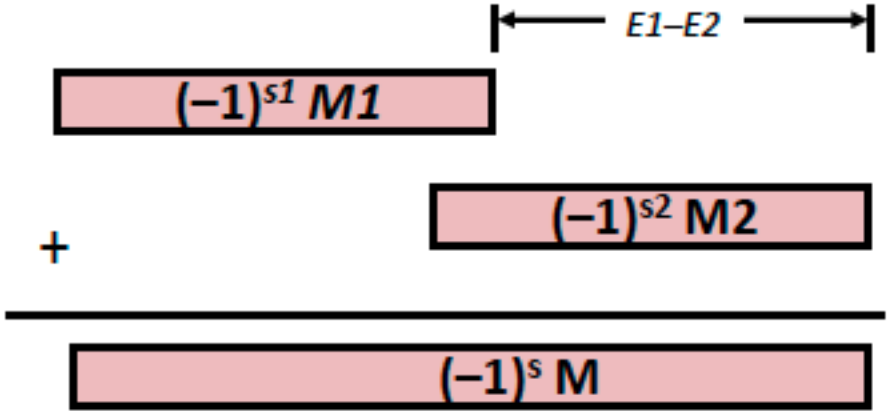
- **Compare to Commutative Ring**
  - Closed under multiplication? ***Yes***
    - But may generate infinity or NaN
  - Multiplication Commutative? ***Yes***
  - Multiplication is Associative? ***No***
    - Possibility of overflow, inexactness of rounding
  - 1 is multiplicative identity? ***Yes***
  - Multiplication distributes over addition? ***No***
    - Possibility of overflow, inexactness of rounding

- **Monotonicity**
  - $a \geq b \ \& \ c \geq 0 \Rightarrow a * c \geq b * c$? ***Almost***
    - Except for infinities & NaNs

Floating Point Addition

$(-1)^{s1}\ M1\ \ 2^{E1}\ \ +\ \ (-1)^{s2}\ M2\ \ 2^{E2}$

Assume $E1 > E2$



- **Exact Result:** $(-1)^{s}\ M\ \ 2^{E}$
  - Sign s, significand M:
    - Result of signed align & add
  - Exponent E: $E1$

- **Fixing**
  - If $M \geq 2$, shift M right, increment E
  - if $M < 1$, shift M left k positions, decrement E by k
  - Overflow if E out of range
  - Round M to fit `frac` precision

Mathematical Properties of FP Add

- **Compare to those of Abelian Group**
  - Closed under addition? ***Yes***
    - But may generate infinity or NaN
  - Commutative? ***Yes***
  - Associative? ***No***
    - Overflow and inexactness of rounding
  - 0 is additive identity? ***Yes***
  - Every element has additive inverse ***Almost***
    - Except for infinities & NaNs
- **Monotonicity**
  - $a \geq b \Rightarrow a+c \geq b+c$? ***Almost***
    - Except for infinities & NaNs

Floating Point in C

- **C Guarantees Two Levels**

  `float` single precision

  `double` double precision

- **Conversions/Casting**
  - Casting between `int`, `float`, and `double` changes bit representation
  - `Double/float` → `int`
    - Truncates fractional part
    - Like rounding toward zero
    - Not defined when out of range or NaN: Generally sets to TMin
  - `int` → `double`
    - Exact conversion, as long as int has ≤ 53 bit word size
  - `int` → `float`
    - Will round according to rounding mode

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

```
1.  x  ==  (int)(double) x
2.  x  ==  (int)(float) x
3.  f  ==  (float)(double) f
4.  d  ==  (double)(float) d
5.  f  ==  -(-f);
6.  1.0/2  ==  1/2.0
7.  d×d  >=  0.0
8.  (f+d)-f  ==  d
```

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

```
1.  x  ==  (int)(double) x      T
2.  x  ==  (int)(float) x       F
3.  f  ==  (float)(double) f    T
4.  d  ==  (double)(float) d    F
5.  f  ==  -(-f);               T
6.  1.0/2  ==  1/2.0            T
7.  d×d  >=  0.0                T
8.  (f+d)-f  ==  d              F
```

- 2.88 a, b-e

Moore's "Law"

- # transistors per chip would double every year for next 10 years
  - Turns out, it's more like every 18 month
- Storage also follows an exponential curve
- What critical improvements are sub-exponential (linear)?

Turning C into Object Code

– Code in files **`p1.c p2.c`**
– Compile with command: **`gcc -O p1.c p2.c -o p`**
  - Use optimizations (**`-O`**)
  - Put resulting binary in file **p**