Lecture 3

normal case.

	1	1	0	1	-3
	0	1	0	1	5
1	0	0	1	0	2

complementary overflow:

	1	1	0	1	-3
	1	0	1	0	-6
~~X~~	0	1	1	1	7

negative overflow

0	1	1	1	7
0	1	0	1	5
1	1	0	0	-4

positive overflow

Wraps Around

① if sum $\geq 2^{w-1}$ ⟹ Become negative 2) At most once

② if sum $< -2^{w-1}$ ⟹ Becomes positive 3) At most once

Standard Multi Function : ignore high order w bits

Implements : $UMult_w(u,v) = u \cdot v \bmod 2^w$.

$5 \times 5 = 25$ $\boxed{1\,0\,0\,1 \mid 1\,0\,0\,1} = 9 = 25 \bmod 16$

But when in the signed number $9 \to -7$

Complementary →

	signed	unsigned
1101	-3	13
1110	-2	14

⇒ $\boxed{\quad\quad \mid 0110}$ while, lower bits →

$13 \times 14 = 182$

$-3 \times (-2) = 6$

while we can have $x = \sum_{i=0}^{w-1} x_i 2^i$

and when $x \ll 2 \Rightarrow x' = \sum_{i=0}^{w-1} x_i \cdot 2^{i+2} = 4x$

in the Reverse, $u \gg k \Leftrightarrow \lfloor u/2^k \rfloor$

for example: 0110 6 ⇒ >>1 0011 3 ⇒ 0001 1 ← unsigned (round it to 0.)

But when using the complementary number

if the number is negative.

eg: 1010 -6 ⇒ >>1 1101 -3 ⇒ >>1 1110 -2

copy the sign value to the empty space. the devide need 30 T.

$X \to -X$ all the bits are fliped, and add one in the tail after this, all the bits are reversed. and add one to it. we will get X again.

when using variable in the for-loop. you should gurantee it will not cause memory error.

the max memory address is 47 bits.

and $2^{10} \cong 10^3 \Rightarrow 2^{40} \cong 10^{12}$ $2^7 \cong 128 \Rightarrow 2^{47} \cong 1.28 \times 10^{14}$

Addr=0	Addr=4	addr=8	addr=12	...	32 bit

addr=0	addr=8	...	64 bit

Byte Ordering

Conventions:

1) Big Endian: Sun. PPC Mac. Least significant byte has highest address

2) Little Endian: x86. ARM, ios, and windows.

For eg: X (variable) = 0x01234567 &x = 0x100

		0x100	0x101	0x102	0x103		
Big		01	23	45	67		
Little		67	45	23	01		

Puzzle

(x) $x<0 \not\Rightarrow ((x*2)<0)$

(✓) $ux >= 0 \overset{\checkmark}{\Rightarrow}$ ux is 0 number be 0

(✓) $x \& 7 == 7 \overset{\checkmark}{\Rightarrow} (x<<30)<0$

(x) $ux > -1$ (is never true)

(x) $x>y \overset{\times}{\Rightarrow} -x<-y$

(x) $x*x>=0$

(x) $x>0 \;\&\&\; y>0 \Rightarrow x+y>0$

✓ $x>=0 \Rightarrow -x<=0$

(x) $x<=0 \Rightarrow -x>=0$ (if the width is 8 and it will represent -128 ~ 127 and -128 is the specific one)

✓ $(x|-x)>>31 == -1$

Lecture 4 Floating Point

Fractional Binary Number

Value	Representations
5 3/4	101.11_2
2 7/8	10.111_2
1 7/16	1.0111_2

1) Divide by 2 by shifting right (unsigned)
2) Numbers of form $0.11111\ldots_2$ are just below 1.0
   we may use notation $1.0 - \varepsilon$

Floating Point Representation

Form: $(-1)^s M \cdot 2^E$

Encoding:

s	exp	frac

sign bit

1) single precision: 32 bits

1	8-bits	23-bits
s	exp	frac

2) double precision: 64 bits

1	11-bits	52-bits
s	exp	frac

"Normalized" value

exp ≠ 000...000 and exp ≠ 111...1

$E = Exp - Bias$ while Exp: unsigned value of exp field

$Bias = 2^{k-1} - 1$ where k is number of exponent bits.

1) single precision: 127 (Exp: 1 ... 254. E: −126 ... 127)
2) Double precision: 1023 (Exp: 1 ... 2046. E: −1022 ... 1023)

$M = 1.xxx\ldots x_2$ while $xxx\ldots x$: bits of frac field

Minimum when frac = 000...0. (M = 1.0) Max when frac = 111...1 (M = 2.0 − ε)

---

eg. $F = 15213.0 \Rightarrow 15213_{10} = 1.1101101101101_2 \times 2^{13}$

M = 1101101101101 E = 13 Bias = 127 Exp = 140 = 10001100

Result: 0 (s) 10001100 (exp) 11011011011010000000000 (frac)

Summary:

$0 \le Exp \le 255$, $-127 \le E \le 128$

Special Values

Condition: exp = 111...1

1) frac = 000...0 → represent infinity: Both positive and negative
2) frac ≠ 000...0 → Not-a-Number (NaN)

Condition: exp = 000...0 → Denormalized value

1) frac = 000...0, → represent zero. 2) frac ≠ 000...0 Numbers closest to 0.0

NaN | −∞ | − Normalized | − Denorm | −0 | +0 | + Denorm | + Normalized | +∞ | NaN

Rounding

Nearest even (default)

$1.40 \rightarrow 1$ $\quad 1.60 \rightarrow 2$ $\quad 1.50 \rightarrow 2$ $\quad 2.50 \rightarrow 2$

$-1.50 \rightarrow -2$

when exactly halfway between two possible values
  Round so that least significant digit is even

Rounding Binary Numbers

odd is 1, even is 0.

(2 bits right of binary point)

e.g. Value	Binary	Rounded	Action	Rounded Value
$2\ 3/32$	10.00011	10.00	$< \frac{1}{2}$ – down	2
$2\ 3/16$	10.00110	10.01	$> \frac{1}{2}$ – up	$2\frac{1}{4}$
$2\ 7/8$	10.11100	11.00	$\frac{1}{2}$ – up	3
$2\ 5/8$	10.10100	10.10	$\frac{1}{2}$ – down	$2\frac{1}{2}$

FP Multiplication

$(-1)^{S1}\, M1\, 2^{E1} \times (-1)^{S2}\, M_2\, 2^{E_2}$

Exact Result: $(-1)^{S}\, M\, 2^{E}$

① $S = S1 \wedge S2$ ② $M = M1 \times M2$ ③ $E = E_1 + E_2$

if $M \geq 2$ shift M right, increment E

puzzles.

① x == (int)(float) x (X)

② x == (int)(double) x (√)

③ f == (float)(double) f (√)

④ d == (double)(float) d (X)

⑤ f == −(−f) (√)

⑥ 2/3 == 2/3.0 (X)

⑦ d < 0.0 ⇒ ((d*2) < 0.0) (√)

⑧ d > f ⇒ −f > −d (√)

⑨ d*d >= 0.0 (X)

⑩ (d+f) − d == f (X)