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      3      positive overflow
-------------
  1 1 0 0     -4
```

Wraps Around

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

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

Standard Multi Function : ignore high order w bits

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

$5 \times 5 = 25$ |0 0 0 1|1 0 0 1| = 9 = 25 mod 16

But when in the signed number 9 → -7

Compile mentary →

	signed	unsigned
1 1 0 1	-3	13
1 1 1 0	-2	14

⇒ while |0 1 1 0| lower bits

$13 \times 14 = 182$

$-3 + (-2) = 6$

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

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

in the reverse, $u >> k \Leftrightarrow \lfloor u/2^k \rfloor$ round it to 0.

for example: 0110 6 ⇒ >>1 0011 3 ⇒ 0001 1 ← unsigned

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.

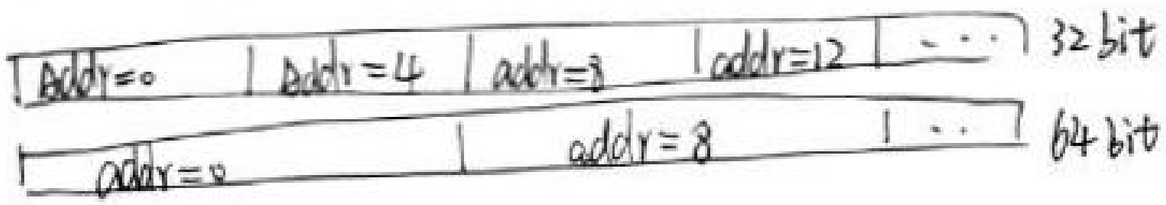
$X \to -X$ all the bits are fliped, and add one in the tail

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 guarantee it will not cause memory error.

the max memory address is 48 bits.

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



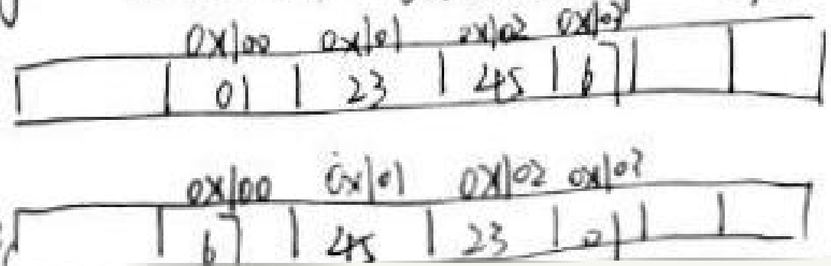
Byte Ordering

Convertions:

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

2) Little Endian: x86. BBM. 80s, and windows.

For eg: X(variable) = 0x012345 6] & X = 0x100



Puzzle

(X) $X < 0 \not\Rightarrow (1X * 2) < 0$

(V) $ux >= 0 \Rightarrow$ ux is 0 number be 0

(V) $X \& 7 == 7 \Rightarrow (X << 30) < 0$

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

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

(X) $X * X >= 0$

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

(V) $X >= 0 \Rightarrow -X <= 0$

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

(V) $(x | -x) >> 31 == -1$

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)

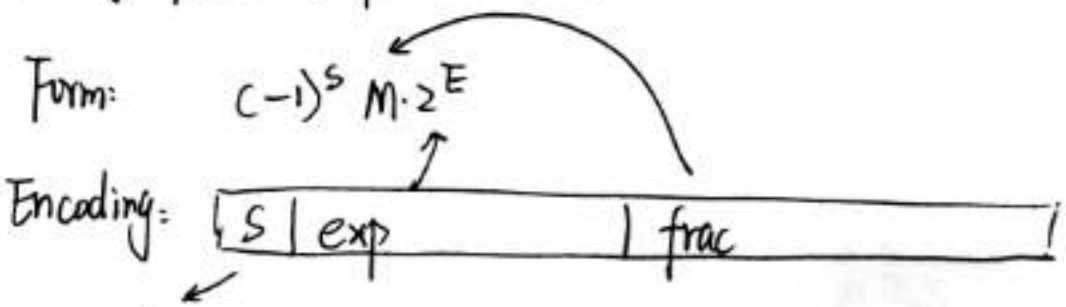
2) Numbers of form $0.111111\ldots_2$ are just below 1.0

we may use notation $1.0-\varepsilon$

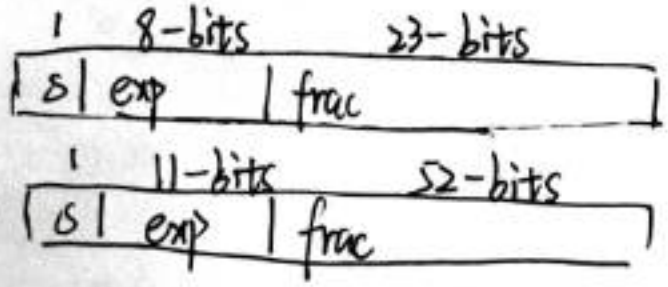
Floating point Representation

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

Encoding:



1) single precision: 32 bits



2) double precision: 64 bits

"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...x: bits of frac field

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

eg. $F = 15213.0 \Rightarrow 15213_{10} = 1.11011011011011_2 \times 2^{13}$

M = 1.10110110110 E = 13 Bias = 127 Exp = 140 = 10001100

Result: 0 | 10001100 | 10110110110 0000 0000

S exp frac

Summary:

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

Special Values

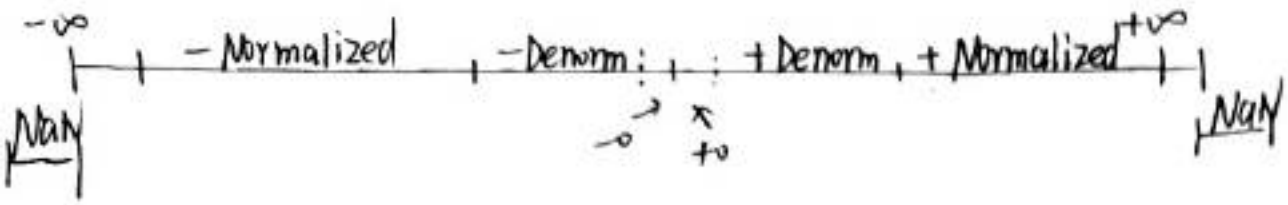
Condition: exp = 111...1

1) frac = 000...0 → represent infinity: Both positive and negative

2) frac ≠ 000...0 → Not-a-Number (NaN)

Condition: exp = 000...0 → Denormalized value to 0.0

1) frac = 000...0, → represent zero. 2) frac ≠ 000... Number closest



Rounding

Nearest even (default)

1.40 → 1 1.60 → 2 1.50 → 2 2.50 → 2

-1.50 → -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.

e.g.	Value	Binary	Rounded	Action (2bits right of binary point)	Rounded Value
	2 3/32	10.00011	10.00	< 1/2 - down	2
	2 3/16	10.00110	10.01	> 1/2 - up	2 1/4
	2 7/8	10.11100	11.00	1/2 - up	3
	2 5/8	10.10100	10.10	1/2 - down	2 1/2

FP Multiplication

$(-1)^{S1} M1 \; 2^{E1} \times (-1)^{S2} M_2 \; 2^{E2}$

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 (V)

③ f == (float) (double) f (V)

④ d == (double) (float) d (X)

⑤ f == -(-f) (V)

⑥ 2/3 == 2/3.0 (X)

⑦ d < 0.0 ⇒ ((d*2) < 0.0) (V)

⑧ d > f ⇒ -f > -d (V)

⑨ d*d >= 0.0 (X)

⑩ (d+f)-d == f (X)

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

$exp \neq 000\ldots000$ and $exp \neq 111\ldots1$

$E = Exp - Bias$. 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 ($M=2.0-\varepsilon$)

eg. $F = 15213.0 \Rightarrow 15213_{10} = 1.11011011011101_2 \times 2^{13}$

$M = 1.1011011011101$ $E = 13$ $Bias = 127$ $Exp = 140 = 10001100$

Result: 0 | 10001100 | 10110110110100000000

s | exp | frac

Summary:

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

Special Values

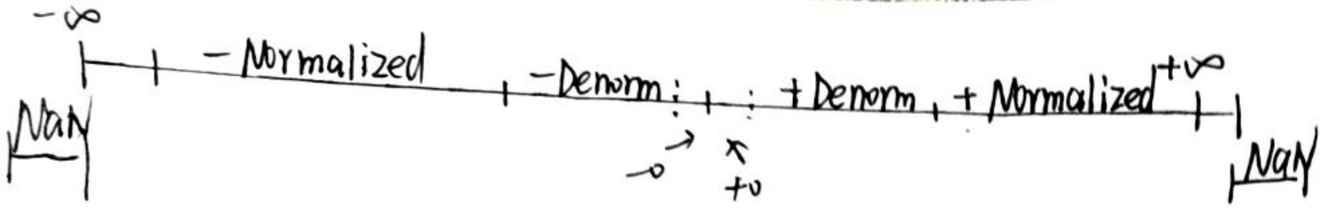
Condition: exp = 111...1

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... Numbers closest to 0.0



Rounding

Nearest even (default)

1.40 → 1 1.60 → 2 1.50 → 2 2.50 → 2

-1.50 → -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.

e.g. value	Binary	Rounded	(2bits right of binary point) Action	Rounded value
2 3/32	10.00011	10.00	< 1/2 - down	2
2 3/16	10.00110	10.01	> 1/2 - up	2 1/4
2 7/8	10.11100	11.00	1/2 - up	3
2 5/8	10.10100	10.10	1/2 - down	2 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 $M >= 2$ shift M right, increment E

puzzles.

① $x == (int)(float)\ x$ (X)

② $x == (int)(double)\ x$ (V)

③ $f == (float)(double)\ f$ (V)

④ $d == (double)(float)\ d$ (X)

⑤ $f == -(-f)$ (V)

⑥ $2/3 == 2/3.0$ (X)

⑦ $d < 0.0 \Rightarrow ((d*2) < 0.0)$ (V)

⑧ $d > f \Rightarrow -f > -d$ (V)

⑨ $d*d >= 0.0$ (X)

⑩ $(d+f)-d == f$ (X)