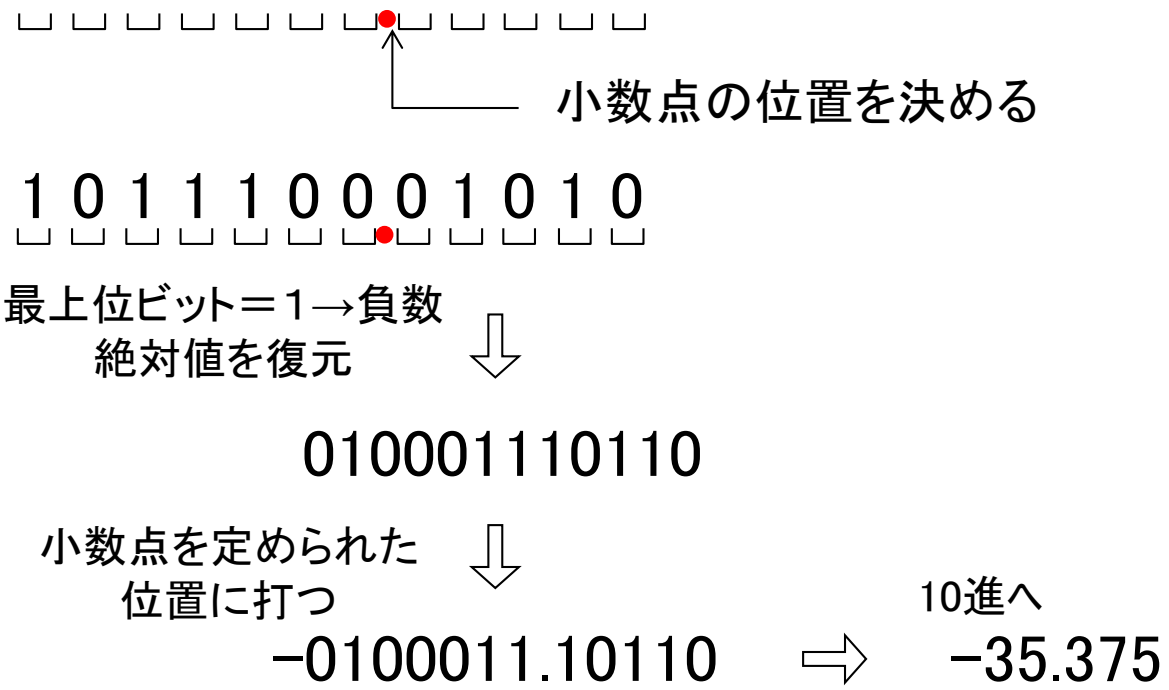
第3章「実数の表現」

- 固定小数点
- 浮動小数点
- けた落ち，情報落ち

固定小数点(fixed point)数

・ 小数点の位置を固定して実数を表現する方法

例)12ビット幅，下から6ビットの右に小数点，2の補数による負数表現



固定小数点数の表現例

12ビット幅，2の補数による負数表現，下から6ビットの左に小数点

計算機内部の表現	2進表記	10進表記
000000000001	000000.000001	0.015625
111111111111	−000000.000001	−0.015625
110001001000	−001110.111000	−14.875

12ビット幅，2の補数による負数表現，下から3ビットの左に小数点

計算機内部の表現	2進表記	10進表記
000000000001	000000000.001	0.125
111111111111	−000000000.001	−0.125
110001001000	−001110111.000	−119

浮動小数点(floating point)数

- 数を次の形で表現: $x \times r^y$
- x を仮数(mantissa), r を基数(base), y を指数(exponent)と呼ぶ
- 基数は予め決めておき, x と y を格納

- 仮数, 指数部分の表現方法, 基数の選択などに自由度がある
  - IBM方式
  - IEEE方式浮動小数点表現

IBM方式の浮動小数点表現形式

表現される数: $(-1)^S \times 16^{E-64} \times F$

符号:	表現する数（仮数）の符号を表す （0：正，1：負）
指数部:	16を基数とし+64のバイアス表現
仮数部:	1以下の2進小数，正規化される

正規化

・ 仮数部を無駄なく使えるよう，桁を上下して調整する

例）10 進小数 0.05 を 2 進小数に変換すると

$$0.0000110011001100110011001100\cdots$$

仮数部24ビット(指数部0)

このままだと先頭にならぶ 4 ビットの0が無駄になる

⇩

そこで，4 ビット左にシフトして，その分指数部を修正

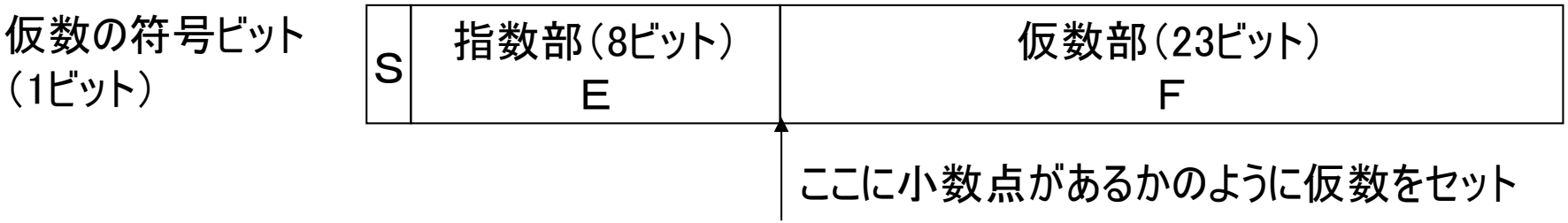
$$0.1100110011001100110011001100 \times 16^{-1}$$

仮数部24ビット　　指数部

IEEE*の浮動小数点数の表現形式(IEEE-754)

*) Institute of Electrical and Electronics Engineers

計算機内部の表現:



表現される数: $(-1)^S \times 2^{E-127} \times (1+F)$

符号:	表現する数(仮数)の符号を表す (0:正, 1:負)
指数部:	2を基数とし +127のバイアス表現
仮数部:	1以下の2進小数, 正規化され, さらに economized form をとる

IEEEの浮動小数点数の表現形式

- 本来の指数に 127 を加えたものを指数部に格納
  - 指数部に 127 が入っていれば，指数は 0 ，128 なら 1，126 なら −1 ということ
- economized form: 仮数部は，1.xxxx となるように正規化され，1 は省略して .xxxx の部分だけを記憶する（「けち表現」）

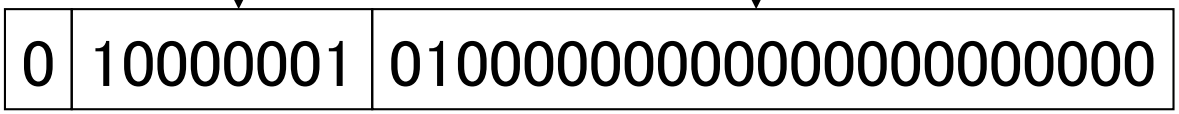
例）10進数の 5

正規化

$$5 = (101)_2 = (101)_2 \times 2^0 = (1.01)_2 \times 2^2$$

バイアス　　economized form

指数部： 2 + 127 = 129　仮数部： 1.01 − 1 = 0.01



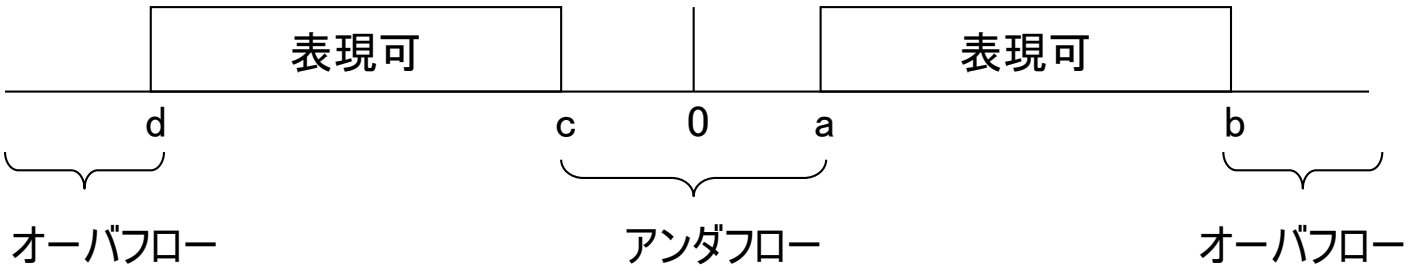
IEEE方式での「特別な値」の表現

・ 次のようなコードが「特別な値」に割り振られている

特別な値	符号	指数部	仮数部
＋∞	0	000…0	000…0以外
－∞	1	000…0	000…0以外
Nan（非数）	0	111…1	000…0以外
	1	111…1	000…0以外
＋0	0	000…0	000…0
－0	1	000…0	000…0

表現し得る数の範囲

- IEEE方式には32ビット(単精度)の他に64ビット(倍精度;指数部11ビット・仮数部52ビット), 拡張倍精度がある
- 32ビット幅(単精度)で表現できる最大・最小数
  - a) 正の最小数: $(1+0) \times 2^{-126}$
  - b) 正の最大数: $(1+0.1111\cdots) \times 2^{127} \sim 2^{128}$
  - c) 負の最大数: $-(1+0) \times 2^{-126}$
  - d) 負の最小数: $-(1+0.1111\cdots) \times 2^{127} \sim -2^{128}$



浮動小数点数の表現例

IEEE浮動小数単精度の表現例

10進数値	ビット列
10	01000001001000000000000000000000
−1	（問1）
0.125	00111111000000000000000000000000
（問2）	10111111010000000000000000000000

10 −> $(1010)_2$ −> $(1.01)_2 \times 2^3$ −> $2^{(130-127)} \times (1.01)_2$

S:0, E: 10000010, F: 010…0 −>（上）

S:0, E: 01111100 (−> $(7C)_{16}$ −> 124), F: 0…0 (−> $(1.0)_2$)

−> $2^{(124-127)} \times (1.0)_2$ −> $2^{-3} \times (1.0)_2$ −> $(0.001)_2$ −> 0.125

浮動小数点数の諸問題

- 仮数部が有限桁しかないこと
  - 丸め誤差
  - 10進小数0.1 = $(0.000110011001100\cdots)_2$
- 指数部も有限桁→オーバフローとアンダフローの発生
- けた落ち
  - ほぼ等しい値の浮動小数点数同士の減算
- 情報落ち
  - 小さな数と大きな数の加減算

⇩

※計算のさせ方一つで結果の精度に差がでる

けた落ち

- ほぼ等しい数値の減算において上位の有効数字が失われること

例）有効数字4桁の10進浮動小数点計算

1235 − 1234 = ?

1235 − 1234 = 1.000

保証されない桁
→ 有効数字1桁に！

計算の仕方で結果が変わる例

$\sqrt{401} - \sqrt{400}$ の計算　　（10進浮動小数点表現を仮定）

直接計算すると…

$$\begin{array}{rl} & \sqrt{401} = 0.20024984 \times 10^2 \\ - & \sqrt{400} = 0.20000000 \times 10^2 \\ \hline & 0.00024984 \times 10^2 = 0.24984000 \times 10^{-1} \end{array}$$

trickyだが次のようにすると…

$$\sqrt{401} - \sqrt{400} = \frac{(\sqrt{401} - \sqrt{400})(\sqrt{401} + \sqrt{400})}{\sqrt{401} + \sqrt{400}}$$

$$= \frac{1}{\sqrt{401} + \sqrt{400}} = 0.24984398 \times 10^{-1}$$

情報落ち

- 大きな数値と小さな数値の加減算により小さい方の仮数部の下位部分が失われること

例）有効数字4桁の10進浮動小数点計算

1234 + 1.234 = 1235.234(真の値）→ 1235（4桁に丸められる）

計算の仕方で結果が変わる例

計算：898.0+1.349+2.348+3.245+4.244 = 909.186（真の値）

加減算の順序を変えると･･･（10進4桁の表現を仮定）

		丸め後
898.0		
1.349	(899.349 →	899.3)
2.348	(901.648 →	901.6)
3.245	(904.845 →	904.8)
+ 4.244	(909.044 →	909.0)
909.0		

		丸め後
1.349		
2.348	(3.697 →	3.697)
3.245	(6.942 →	6.942)
4.244	(11.186 →	11.19)
+ 898.0	(909.19 →	909.2)
909.2		

※ 一般に小さいものから順に加減算してゆくと精度が保たれる