1. 情報の表現

整数、負の数、浮動小数点
BCDコード、文字
をどうやって計算機内部で表すか？

概要

- 数値の表現
  - 2進数／16進数と基数変換
  - 2の補数による負の数の表現
  - 浮動小数点数
- 様々な情報の表現
  - 2進化10進数
  - 文字コード
  - 他の情報の表現の例

1.1 数値やデータの表現

- Q: 計算機内部では何でデータを表すか？
  ->A: 電気信号
- Q: 電気信号を使ってどのように表すか？
  ->A: 電気信号のON/OFF
  - 中間の値を使うのは効率が悪い
  - 現在は電圧の印加あり／印加なしが主流

- この状態をどのようにデータに対応させましょう？
  ->ON/OFFを数字(1/0)に対応させましょう

ビット数と表現の数

1ビット: 一つの”0”と”1”の値で表現できる情報量の最小単位

ビット数を増やして扱える情報量を増やせる

		表せる状態の数
1ビット	1	0, 1 →2通り
2ビット	1 1	00, 01, 10, 11 →4(2の2乗)通り
3ビット	1 1 1	000, 001, 010, 011, 100, 101, 110, 111 →8(2の3乗)通り
4ビット	1 1 1 1	
5ビット	1 1 1 1 1	
6ビット	1 1 1 1 1 1	
7ビット	1 1 1 1 1 1 1	
8ビット	1 1 1 1 1 1 1 1	00000000, 0000001, ….. →256(2の8乗)通り

1バイト = 8 ビット

(2) 2進数／16進数と基数変換

- n進数: 桁が上がるごとに数値の持つ意味がn倍になる記数法
  - 世の中の大部分は10進数で動いています
    - 10の桁の数字は1の桁の10倍の、100の桁の数字は10の桁の10倍の意味を持つ
  - 世の中の10進数以外の例
    - 時計の分／秒は60進数
- 先ほどのビットによる表現は2進数と1対1で対応
  - 例: 000, 001, 010, 011, 100, ...

    (上の桁は2倍の意味を持つ)

    1ビット = [2進数の一桁]

基数変換

- 2進数では桁が多くなりすぎるので16進数も併用

- では、10進数とのやりとりはどうする？

  ->基数変換: 数の基数を異なる基数に変換すること

  – ”桁が上がるごとの数値の意味する量”を基数と呼ぶ

2進数の基数は	2
10進数の基数は	10
16進数の基数は	16

2進数/10進数/16進数の関係の例

- 16進数の10-15はA-Fで表記
- 16進数と2進数の桁上がりに着目!　->基数変換が楽

- 何進数か分かりやすくする工夫
  - 0x1F
  - &H03A
  - $(0110)_2$
  - 8b'00101110

10進数	2進数	16進数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

基数変換を行うには？

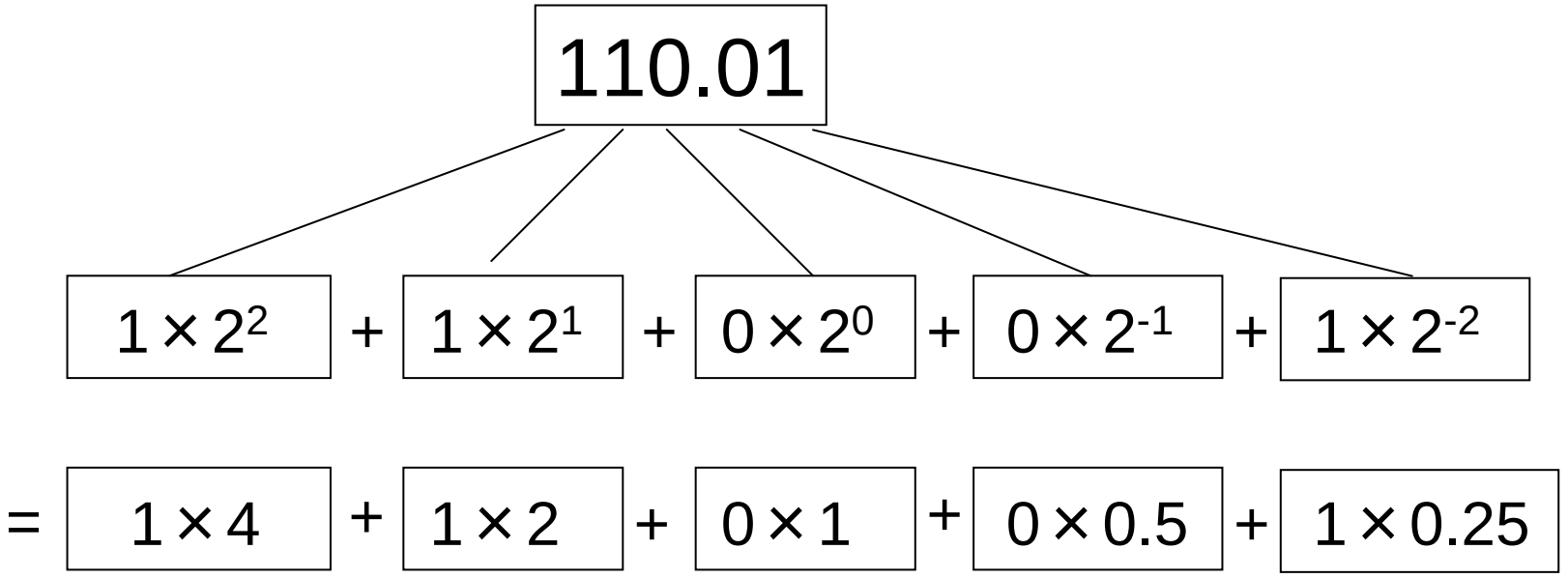
- 桁の重みに着目して、変換する
- 10進数の場合

例　348.5

$3 \times 10^{2} + 4 \times 10^{1} + 8 \times 10^{0} + 5 \times 10^{-1}$

2進数から10進数への基数変換

- 桁の重みに着目して、変換する
- 2進数の場合



16進数から10進数への基数変換

- 桁の重みに着目して、変換する
- 16進数の場合

2D.8

$= 2 \times 16^1 + D \times 16^0 + 8 \times 16^{-1}$
$= 2 \cdot 16^1 + 13 \cdot 1 + 8 \cdot (1/16)$
$= 32 + 13 + 0.5$
$= 45.5$

10進数から2進数への基数変換

1. 整数部と小数部に分ける
2. 整数部は2で割り、その剰余を2進数整数部の下位の桁の数値とする。これを、被除数が0になるまで続ける。
3. 小数部は2倍し、その整数部を2進数小数部の上位の桁の数値とする。これを、小数部が0になるまで続ける。
4. 得られた2進数整数部と小数部を連結する。

10進数から2進数への基数変換の例

21.625

21 ＋ 0.625

2で割り余りに注目

2	21	1	2^0
2	10	0	2^1
2	5	1	2^2
2	2	0	2^3
2	1	1	2^4
	0		

2倍して整数の数字に注目

0.625 x 2 = 1.25	1	2^{-1}
0.25 x 2 = 0.5	0	2^{-2}
0.5 x 2 = 1.0	1	2^{-3}

→ 10101.101

16進数から2進数への基数変換

2進数の時と同様。2が16になるだけ。

21.625

21 + 0.625

16で割り余りに注目

16倍して整数の数字に注目

16 | 21　5　16^0

16 | 1　1　16^1

0

$0.625 \times 16 = 10.0$　A　16^{-1}

15.A

16進数から2進数への基数変換

- 4ビットごとに区切って16進数に変換するだけ
  - 非常に楽！
- 逆もまた然り
- 16進数は2進数を短く表す手段として有用

$$\underline{1100}\ \underline{0011}.\underline{0100}$$

$$= 1100 \cdot 16^{1} + 0011 \cdot 16^{0} + 0110 \cdot 16^{-1}$$
$$= C \cdot 16^{1} + 3 \cdot 16^{0} + 4 \cdot 16^{-1}$$
$$= C3.4$$

(3) 2の補数による負の数の表現

- 負の数はどのように表現する？
  ->2の補数(2進数の2の補数)というルールを使う
  - 一応、n進数のnの補数の一部ですが、割愛します
- 2の補数の作り方

0100 0001	(65)
1011 1110	0/1を反転させる
1011 1111	1を足す
1011 1111	(-65)　2の補数を負の数と扱う

先頭ビットが1のとき負の数

2の補数による表現の値域

- nビットの2進数の場合、$-2^{(n-1)}$～$2^{(n-1)}-1$
- 8ビットの例

2進数	10進数	16進数
0111 1111	127	7F
0111 1110	126	7E
...	....	...
0000 0001	1	1
0000 0000	0	0
1111 1111	-1	FF
1111 1110	-2	FE
...	...	...
1000 0001	-127	81
1000 0000	-128	80

2の補数の考え方

- 最上位の桁の重みの正負が反転していると考える

正負なし表記

$0011\ 1111 = 0\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 63$
$1111\ 1111 = 1\cdot2^7 + 1\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 255$
$1000\ 0000 = 1\cdot2^7 + 0\cdot2^6 + 0\cdot2^5 + 0\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 = 128$

2の補数表記

$0011\ 1111 = -0\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 63$
$1111\ 1111 = -1\cdot2^7 + 1\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = -1$
$1000\ 0000 = -1\cdot2^7 + 0\cdot2^6 + 0\cdot2^5 + 0\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 = -128$

2の補数表記に関する諸知識(1/2)

- 正の数の2の補数を取ると負の数に、負の数の2の補数を取ると正の数になる
  - 0100 0001 (65) -> 1011 1110 -> 1011 1111 (-65)
  - 1011 1111 (-65) -> 0100 0000 -> 0100 0001 (65)
- 上記の特徴により、加減算を同時に実現しやすい
  - 加算: そのまま足せば良い
  - 減算: 引く数の2の補数を取り、加算する。
    - 0/1を反転して1を加算するという作業は、回路で実現しやすい(詳細はハードウェア設計論Iで)

10進数の筆算に同じ

```
  0000 1010   (10)
+ 0000 1100   (12)
  0001 0110   (22)
```

```
  0001 0110   (22)
+ 1111 1100   (-4)
1 0001 0010   (18)
```

8ビットの範囲を
超えた分は無視する

2の補数表記に関する諸知識(2/2)

- 負の数を使わない場合、正の数だけの表記を利用することもある
  - 利用できる正の数の範囲が広がる (0～2^n-1)
- 最上位ビットが符号に一致するため、符合ビットとも呼ぶ
  - 0: 正、1: 負
- オーバーフローには注意しましょう
  - 正の数同士を足したら結果が負の数になった？？？

    ->値がnビットで表現できる範囲を超えたため
    - 例：結果(153)が8ビットで表現できる-128～127の範囲を超えた

    0110 0110 (102) + 0011 0011 (51) = 1001 1001 (-103)

演習

[1] 55と120を8ビットの2進数で表そう

```
2 | 55
2 | 27 ... 1
2 | 13 ... 1
2 |  6 ... 1
2 |  3 ... 0
2 |  1 ... 1
     0 ... 1
```

```
2 | 120
2 |  60 ... 0
2 |  30 ... 0
2 |  15 ... 0
2 |   7 ... 1
2 |   3 ... 1
2 |   1 ... 1
     0 ... 1
```

$(55)_{10} = (00110111)_2$

$(120)_{10} = (01111000)_2$

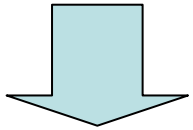
[2] -120を2の補数で表し、55 – 120を計算しよう。

120の2の補数: 01111000 ---(ビット反転)---> 10000111 ---(+1)---> 10001000

$$
\begin{array}{r}
00110111\ (55)_{10} \\
+\ 10001000\ (-120)_{10} \\
\hline
10111111\ (-65)_{10}
\end{array}
$$

(4)浮動小数点表記

- 計算機は大きな数や小さな数のどちらも取り扱うことになることが多い
  - 整数部と小数部の最適な割合は？
    - 整数部が少ないと大きな数を表せない
    - 小数部が少ないと小さな数を表せない
  - ビット数を増やすとデータ量が増えて嬉しくない



- 浮動小数点表記を使うという解

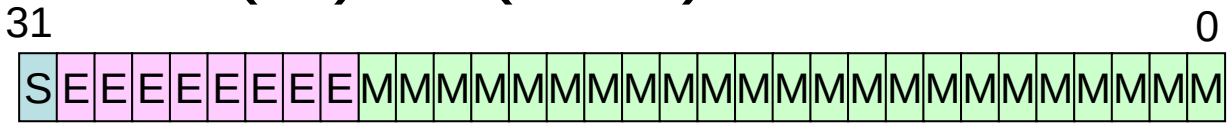
$$(\text{仮数}) \times (\text{基数})^{(\text{指数})}$$

$$1234000000000 = (1.234) \times (10)^{(12)}$$

IEEE(IEEE 754)規格の浮動小数点表記

- 32ビットのデータ量で-2^{127}～2^{128}の数値を表記
  - ただし、厳密に表現できるわけではありません(誤差あり)
- 以下のフィールドからなる
  - 1ビットの符号ビット(S)　0:正、1:負
  - 8ビットの指数部(E)
  - 23ビットの仮数部(M)
    - 1桁目が1になるように左詰で表現 0.00101 -> 1.01
    - 1桁目が必ず1になるので、その部分はMには含めない
      ->仮数部の実質的な長さが24ビットになる

$$(-1)^S \times (1+M) \times 2^{E-127}$$



IEEE規格への変換の例

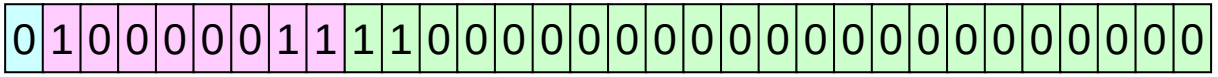
1. 数値を符号無し2進数で表す。同時に、正負の値より符号ビットSを決定する。
2. 最初の1となるビットが左詰になるように仮数を設定する。最初の1を除いた23ビットがMとなる。
   - 24ビット目移行は無視される。これは、丸め誤差となる。
3. 指数部の値をx－127で表せる10進数xを求め、それを2進数に変換してEとする。

IEEE規格への変換の例

- 10進数28の変換

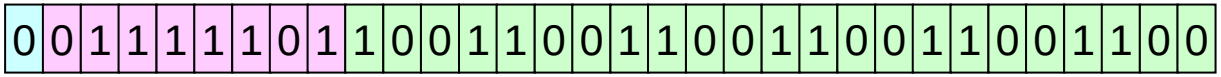
$(28)_{10}$
$= (-1)^0 \times (00011100)_2$
$= (-1)^0 \times (1.1100) \times 2^4$
$= (-1)^0 \times (1 + 0.1100) \times 2^{131-127}$
$= (-1)^0 \times (1 + 0.1100) \times 2^{10000011-01111111}$

よって、
S=0
M=110 0000 0000 0000 0000 0000
E=1000 0011



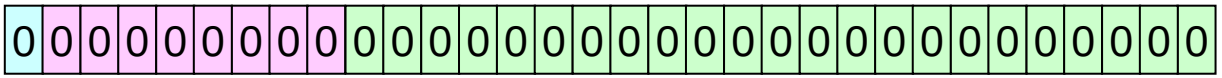
IEEE規格への変換による誤差の例

- 大小の値を表記できるには誤差という代償がある
  - 0.4の変換結果
    - 0.4は2進数では循環小数になってしまう ->打切り誤差



=0.399999976

  - 0の変換結果



$= 1.0 \times 2^{-127} = 5.87747175 \times 10^{-39}$

->厳密な結果が必要な場合は、浮動小数点表記は使わない
  - 倍精度や4倍精度の浮動小数点表記という物もある

演算の過程の誤差

- 浮動小数点表記の演算
  - 指数の大きいほうに仮数を揃えて演算

$$(-1)^0 \times (1.1100) \times 2^4 + (-1)^0 \times (1.1000) \times 2^6$$
$$= (-1)^0 \times (0.0111) \times 2^6 + (-1)^0 \times (1.1000) \times 2^6$$
$$= (-1)^0 \times (1.1111) \times 2^6$$

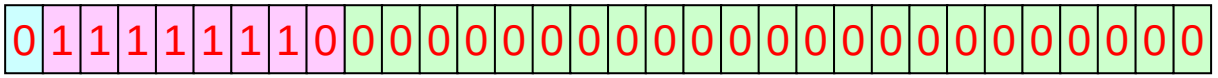
- 誤差
  - 丸め誤差: 仮数に入らなかった部分を捨てるため
  - 情報落ち: 指数を揃えたら仮数に使えるビットからはみ出した
  - 桁落ち: 引き算の結果によっては仮数のビット数が最大値より短くなる

演習

- 0.5をIEEE規格に従ってビット列であらわしてみよう

$(0.5)_{10}$
$= (-1)^0 \times (0000.1000)_2$
$= (-1)^0 \times (1.0000) \times 2^{-1}$
$= (-1)^0 \times (1 + 0.0000) \times 2^{126-127}$
$= (-1)^0 \times (1 + 0.0000) \times 2^{01111110-01111111}$

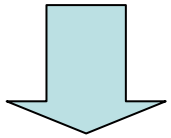
よって、
S=0
M=000 0000 0000 0000 0000 0000
E=0111 1110



1.2様々な表現
(1)2進化10進数

- 2進数表記の欠点
  - 通常の表記では、小数点以下が近似になるケースが多い
  - 浮動小数点表記だと誤差が多い

→厳密な数値を使いたい業界(金融業界など)も多々ある



- 多少効率は悪くても、厳密な数値表現が欲しい

→2進化10進数

  - 基本的に、10進数の1桁を4ビットのデータに1対1変換

２進化10進数の例

- ゾーン10進数
  - ゾーンビットで文字コードと共存

		ゾーンビット	数値	ゾーンビット	数値	符号ビット	数値
JISコード形式	123 ->	0011	0001	0011	0010	1100	0011
	-123 ->	0011	0001	0011	0010	1101	0011
EBCDICコード形式	123 ->	1111	0001	1111	0010	1100	0011
	-123 ->	1111	0001	1111	0010	1101	0011

- パック10進数
  - データサイズを縮小

	数値	数値	数値	数値	数値	数値	符号ビット
123456 ->	0001	0010	0011	0100	0101	0110	1100
-123456 ->	0001	0010	0011	0100	0101	0110	1101

(2)文字の表現

- 文字を数字として表現
  - 65 = "A", 66 = "B", 67 = "C", ...
  - 48 = “0”, 49 = “1“, ...
- 上記はASCIIコードの割り当てです
  - ASCII: American Standard Code for Information Interchange Code
  - 他にも様々な割り当て方が存在

演習

- “NAIST”をASCIIコードの数値で表現してみよう
  - “A”に対応する数値から、各文字の数値を類推して下さい

前スライドより65 = “A”なので、”N” = 78, “I” = 73,
“S” = 83, “T” = 84が類推できる。よって、
“NAIST” = “78 65 73 83 84”
とASCIIコードの数値で表現できる。

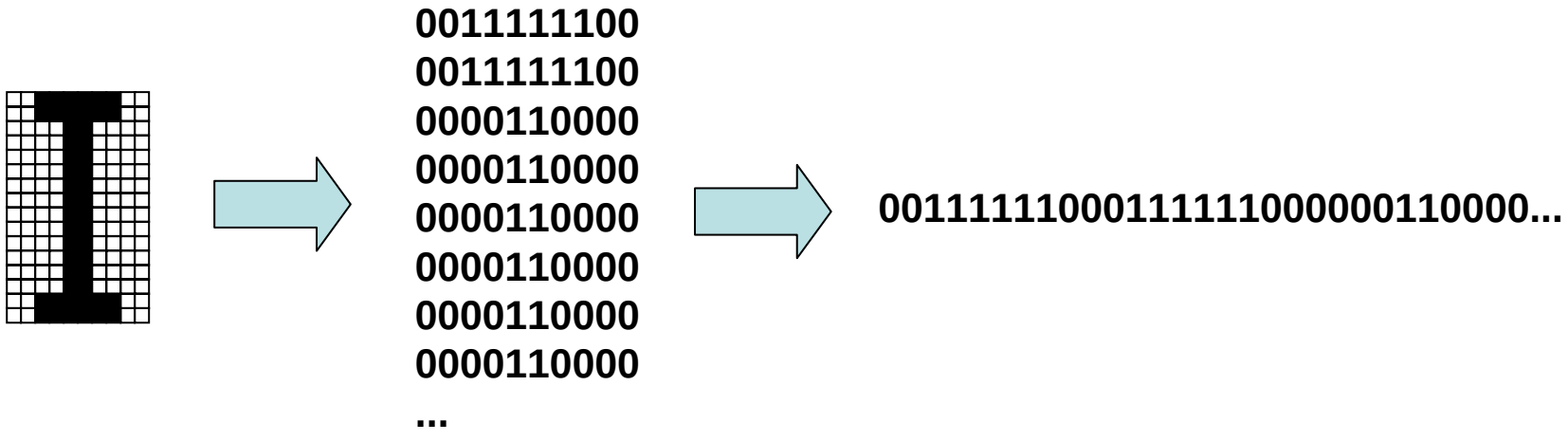
アルファベット以外を表示するためには？

- 漢字等は文字が沢山！

  ->16ビット以上の数値で表現
- アルファベット(8ビット)と共存させるための工夫で様々な規格がある
  - EUC
  - JIS
  - シフトJIS
- 近年はUnicodeに統一されつつある

(3) 他の情報の割り当ての例

- 色の割り当て
  - 3原色(Red, Green, Blue)を数値化して割り当て
  - 例1: 赤色は(R, G, B)=(255, 0, 0)　　注:値域0-255
  - 例2:スカイブルー: (R, G, B) = (135, 206, 235)
- 画像の割り当て
  - 画素に区切って、画素情報を左上から羅列する



(4) 補助単位

- 大きな値や小さな値をいちいち乗数を利用して表現するのは面倒

  ->国際単位系で定義されている補助単位を使う
- 10^3単位だが、コンピュータ業界では2^{10}単位にも使う
  - 2^{10}単位はki, Mi, Gi, Tiなど、”i”を付加して区別することもある

記号	読み方	乗数表現	(2の乗数での近似表現)
E	エクサ	10^{18}	($2^{60} \fallingdotseq 1.153 \times 10^{18}$)
P	ペタ	10^{15}	($2^{50} \fallingdotseq 1.126 \times 10^{15}$)
T	テラ	10^{12}	($2^{40} \fallingdotseq 1.010 \times 10^{12}$)
G	ギガ	10^{9}	($2^{30} \fallingdotseq 1.074 \times 10^{9}$)
M	メガ	10^{6}	($2^{20} \fallingdotseq 1.048 \times 10^{6}$)
k	キロ	10^{3}	($2^{10} \fallingdotseq 1.024 \times 10^{3}$)
m	ミリ	10^{-3}	($2^{-10} \fallingdotseq 0.977 \times 10^{-3}$)
μ	マイクロ	10^{-6}	($2^{-20} \fallingdotseq 0.954 \times 10^{-6}$)
n	ナノ	10^{-9}	($2^{-30} \fallingdotseq 0.931 \times 10^{-9}$)
p	ピコ	10^{-12}	($2^{-40} \fallingdotseq 0.905 \times 10^{-12}$)

1章のまとめ

- 計算機上では情報は0/1のビットで表す
- 0/1の羅列で情報を表すため、数値表現は2進数となる
- 負の数を表すために2の補数表現が用いられる
- 大きな数を表すため、浮動小数点表記が用いられる
- 文字等の情報も数値で表現できるように符号化される