コンピュータとディジタル表現（概要）

関連：p.34-35

① コンピュータにおけるデータ(0,1)の表現方法
（電圧，磁石，凹凸，電荷等）

② データの量を表す単位（ビット，バイト，接頭辞）

③ analogをdigitalへ(量子化)・・・ そしてOと1だけに(符号化)・・・
（ディジタル化のメリット・デメリット）

今後，更に下記の内容も学習していくことになる。

- 電子計算機の動作を意識する・・・2進数の処理確認
（加法・シフト・補数・誤差 等）
- コンピュータが計算するしくみ（論理演算と論理回路）
- ハードウェア＋ソフトウェア＝コンピュータ
- コンピュータの内部構造・外部装置・インターフェース

接頭語（接頭辞）の補足説明

関連：p.34-35

バイトの単位一覧

SI接頭辞			2進接頭辞		
単位（記号）	慣用値	SI基準	単位（記号）	値	SIとの差(概数)
キロバイト (kB)	2^{10}	10^3	キビバイト (KiB)	2^{10}	2.400000%
メガバイト (MB)	2^{20}	10^6	メビバイト (MiB)	2^{20}	4.857600%
ギガバイト (GB)	2^{30}	10^9	ギビバイト (GiB)	2^{30}	7.374182%
テラバイト (TB)	2^{40}	10^{12}	テビバイト (TiB)	2^{40}	9.951163%
ペタバイト (PB)	2^{50}	10^{15}	ペビバイト (PiB)	2^{50}	12.589991%

図：ウィキペディアより

SI接頭辞(十進法ベース)
International System

2進接頭辞(二進法ベース)
新しい**表現**(**i**が付く)はそれほど普及しておらず，まだ混乱解消には至らず。
（つまり２進とSI：同じ表記 ⇔ 見ただけで区別できず）

単位	英語名（省略形：右側）	2進接頭辞でのデータ量（情報量）表記
キビバイト	Kilobinary Byte= Kibi Byte（KiB）	1KiB = 1,024B= 2^{10} = 1,024 Byte
メビバイト	Megabinary Byte=Mebi Byte（MiB）	1MiB = 1,024KiB= $(2^{10})^2 = 2^{20}$ = 1,048,576 Byte
ギビバイト	Gigabinary Byte= Gibi Byte（GiB）	1GiB = 1,024MiB $= (2^{10})^3 = 2^{30}$ = 1,073,741,824 Byte
テビバイト	Terabinary Byte= Tebi Byte（TiB）	1TiB = 1,024GiB $= (2^{10})^4 = 2^{40}$ = 1,099,511,627,776 Byte

「蟹，見たことある？，宇宙人，コンピュータ」をまとめると・・・

- Computer　1, **10**, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, …
- 蟹　1, 2, 3, **10**, 11, 12, 13, 20, 21, 22, 23, …
- 妖怪人間　1, 2, 3, 4, 5, **10**, 11, 12, 13, 14, 15, …
- ハ…大魔王　1, 2, 3, 4, 5, 6, 7, **10**, 11, 12, 13, …
- 人間　1, 2, 3, 4, 5, 6, 7, 8, 9, **10**, 11, …

～の表記方法 → ～の立場なら ↓	computer	蟹	妖怪人間	ハクション大魔王	人間	6本指の宇宙人	8本指の宇宙人
computerなら	10進法						
蟹なら		10進法					
妖怪人間なら			10進法				
ハクション大魔王なら				10進法			
人間なら	2進法	4進法	6進法	8進法	10進法	12進法	16進法
6本指の宇宙人なら						10進法	
8本指の宇宙人なら							10進法

5．n進数の演算　主に2進数を扱う

関連：p.38-39

加法　・・・・筆算で。

補数　・・・・ある数に数xを加えると桁上りが生じて，今までの数(桁)が全てOになってしまうとき，xを補数という。

補数の例と計算
（10進数の場合）　・・・　（2進数の場合）

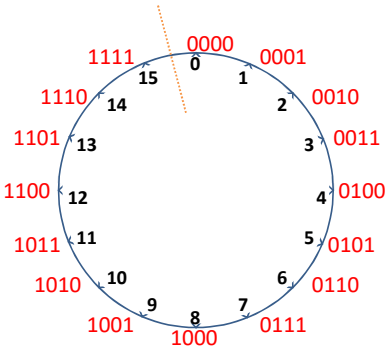
減法　・・・・補数と加法を利用して計算してみよう。⇒プリントへ
※ 減法・乗法・除法も，加法を利用して行うことができる。

乗法　・・・・加法と何かを用いて
例　$(0000\ 0101)_2 \times (0100\ 1000)_2 =$

除法　・・・・加法と何かを用いて
例　20÷６ ＝ ３・・・２　（わかりやすくするために10進法で）

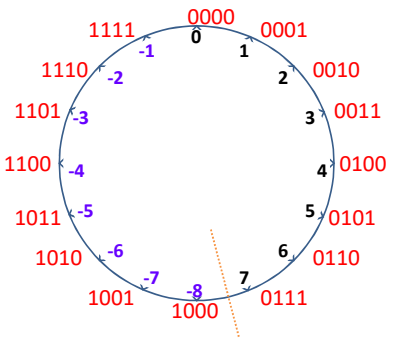
符号なし整数と符号付き整数を図示(4bit)

関連：p.38-39,97



符号なし整数と符号付き整数を図示(4bit)

関連：p.38-39,97



10000

浮動小数点表示：実数（小数）や大きい数（概数）の表現方法

00000	00001	00010	00011	00100	・・・・・
・・・・・	11011	11100	11101	11110	11111

これらを $\pm 1.\square \times 2^{\square}$ の形で扱う（浮動小数点表示）

符号　仮数 ×基数(底ともいう)　指数

つまり□□□□□(0,1)を符号・指数・仮数(小数部分)に振り分ける

- ±について　・・・・ 0 ⇔ ＋　　1 ⇔ －
- 「1.」と「×2」・・・・ 表現する必要なし
- 0や±∞はこの方式で表現できない　・・・・特別ルール必要

順番に注意！

※ 厳密には更に詳細な規則が多く存在するが，まず，簡略化した実習プリントでざっくりと感覚をつかもう。

【実習】浮動小数点表示等の体験（5bit）

関連：p.38-39,97

- まず単純な**固定小数点表示**を体験してから，**浮動小数点表示**を体験しよう。表現した数は**○○小数点数**という。
- 実際は32個の数しか表示できない5bitは実用化されていないが，実習用に作成した教材で理解を深めよう。
- 0に近い小数を表現するには，指数に負の数を使う必要がある。
- 指数に使う符号付きの整数は，「符号付き整数表示」とは異なる「**バイアス(bias)表現**」と呼ばれる方法を用いる。表現された**バイアス値**は，符号付き整数の規則に対して**1だけ**ずれている。例えば8bitならば −128〜+127 ではなく**−127〜+128**となる。
- 今回は5bitなので，符号1bit，仮数1bit，指数3bitとして実習(体験)してみよう。もちろん基数（底）は2である。

参考：浮動小数点表示(IEEE754)の具体的説明1

関連：p.38-39,97

32bit浮動小数点表示で表現可能な範囲はなぜ約10^{-45}〜約10^{38}か？

□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□

符号部 & 仮数部 × 基数^{指数部} の形式

符号部(1bit)　指数部(8bit)　仮数部(23bit)

- 符号部(1bit)　＋⇒0　－⇒1
- 指数部(8bit)　0000 0000〜1111 1111
- 仮数部(23bit)　0が23個〜1が23個(2^{23}個)

種類	Exp（指数部）	Fraction（仮数部）
ゼロ	0	0
非正規化数	0	0以外
正規化数	1〜254	任意
無限大	255	0
NaN	255	0以外の任意

指数表記 ±1.□□□・・・×2□□・・・

の形式において，次の課題や要望等が生じる。

- ゼロが表現できない。
- ±無限大も表現しなければならない。
- さらに微小な(0に近い)数を表現したい。

⇒特別ルールを用いた工夫で，課題を克服する。

これは何？あとで・・・。

参考：浮動小数点表示(IEEE754)の具体的説明2

関連：p.38-39,97

32bit浮動小数点表示(IEEE754)で表現可能な範囲を求める。

符号部 & 仮数部 × 基数^{指数部}　符号部(1bit)　指数部(8bit)　仮数部(23bit)　の順

□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□

- 符号部(1bit)　＋⇒0　－⇒1
- 指数部(8bit)　2^8 ＝ 256通り（0000 0000〜1111 1111）

  0000 0000〜1111 1111は符号なしだと0〜255と読めるが，小数(マイナス○乗)を表現しなければならない。そのために−127〜128（バイアス値という）を使用する（符号付き整数−128〜127ではないことが鬱陶しいかも）。

  更に，−127乗と+128乗だけ特別ルールが適用（0〜255として扱うと両端の0，255が特別扱い）なので，実際は 2^{-126}〜2^{+127}（1〜254）。

  計算：$2^{+127} \fallingdotseq 1.701411834604692317 31\cdots \times 10^{38}$

  「約10^{38}まで」が解決！
- 仮数部(23bit)　±1.□□・・・□□□（□が23個）

  仮数部だけで，正負共に2^{23}＝8,388,608通り。

  すなわち $2 \times 2^{23} = 16,777,216$通り表現可。

種類	Exp（指数部）	Fraction（仮数部）
ゼロ	0	0
非正規化数	0	0以外
正規化数	1〜254	任意
無限大	255	0
NaN	255	0以外の任意

参考：浮動小数点表示(IEEE754)の具体的説明3

関連：p.38-39,97

32bit浮動小数点表示(IEEE754)で表現可能な範囲を求める。

符号部 & 仮数部 × 基数^{指数部}　符号部(1bit)　指数部(8bit)　仮数部(23bit)　の順

□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□

- 符号部(1bit)　＋⇒0　－⇒1
- 指数部(8bit)　256通り（0000 0000〜1111 1111）
- 仮数部(23bit)　0か1が23個(2^{23}個)　±1.□□・・・□□□（□が23個）

考察

2進数の指数(ここでは23)に約0.30103をかけると，10進数に変換した時の指数が確認できる(詳しい概念は数学で「対数」を学習すると理解できる)。

例1：$23 \times 0.30102999566 = 6.92368990018$　$10^{6.92368990018} = 8,388,607.99823132\cdots$

例2：2^{10} の指数10に0.30102999566をかけて$10^{3.0102999566}$とすれば10^3=1000より少し大きい1,023.99907893・・・となる。無限に計算できるなら1024(=2^{10})だ。わかった？

$2^{+127} \fallingdotseq 10^{+127 \times 0.30102995\cdots} = 10^{38.230803\cdots}$

$= 1.701411834604692\cdots \times 10^{+38}$

という計算も可。

上の数値(赤)と左の数値(青)，見覚えあるだろ？

参考：浮動小数点表示の具体的説明4

関連：p.38-39,97

「指数部 0000 0000(−127乗)と1111 1111(+128乗)の特別ルール」とは？

- 指数部(8bit)：0000 0000(−127乗)のとき
  - 仮数部(23bit)が全て0（0が23個）⇒ ±0と見なす(指数表記で0は表現不可能だが)。
  - 仮数部(23bit)に1が含まれる　⇒ 仮数部を ±1.□□・・・□□□ではなく，特別に ±□.□・・・□□□とする(23bitに整数部分も含める)。これにより，ゼロに最も近い数は，

    $\pm 0.00\cdots001$(整数部分も含めて0が1+21個と1が1個で23bit) $\times 2^{0000\ 0000}$

    $= \pm 2^{-22} \times 2^{-127} = \pm 2^{-149} = \pm 10^{-149 \times 0.301029995\cdots} = \pm 10^{-44.853469255\cdots}$

    つまり　「±ともに約10^{-45}〜約10^{38}まで」が解決！

    ちなみに $\pm 10^{-44.853469255\cdots} = \pm 1.401298783543596303\cdots \times 10^{-45}$ である。
- 指数部(8bit)：1111 1111(+128乗)のとき
  - 仮数部(23bit)が全て0（0が23個）⇒ ±無限大∞　※ 固定小数点表示には∞なし
  - 仮数部(23bit)に1が含まれる　⇒ NaN(Not a Number 非数)

    例えば「無限大−無限大，0/0，負数の平方根」等の実数として表現できない結果や，オーバーフロー(桁あふれ)，計算上必要な値が得られない等，確かな数値を表示できない場合等における表現方法として使用される。

□□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□