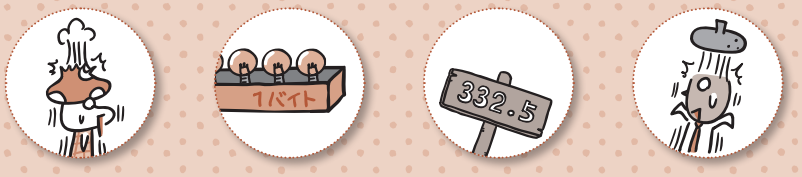
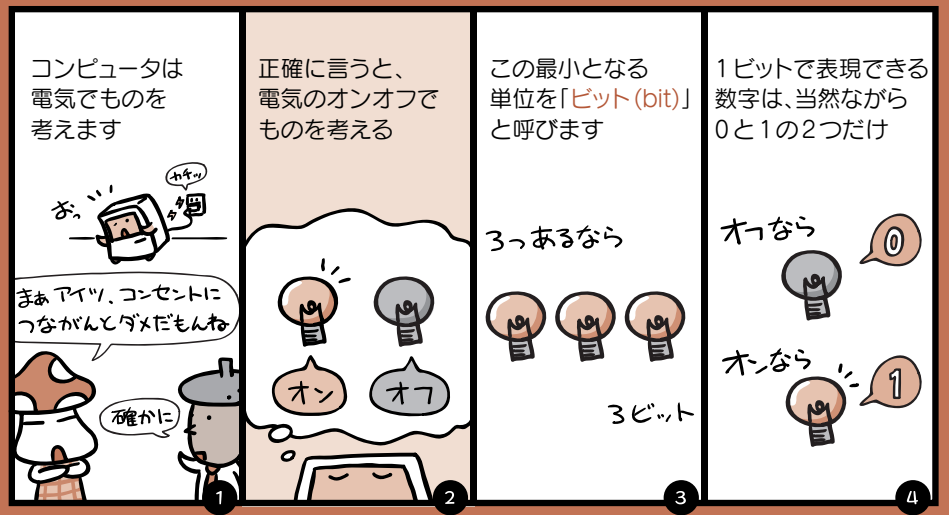
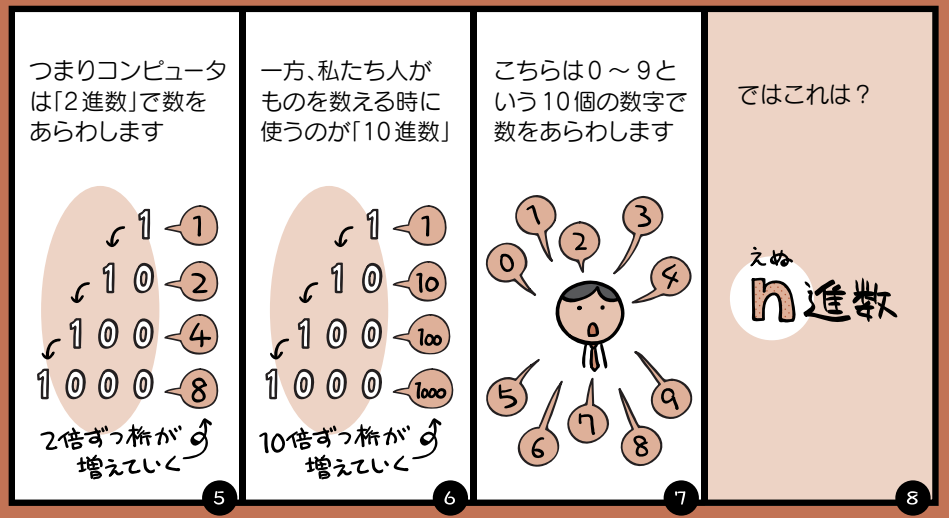
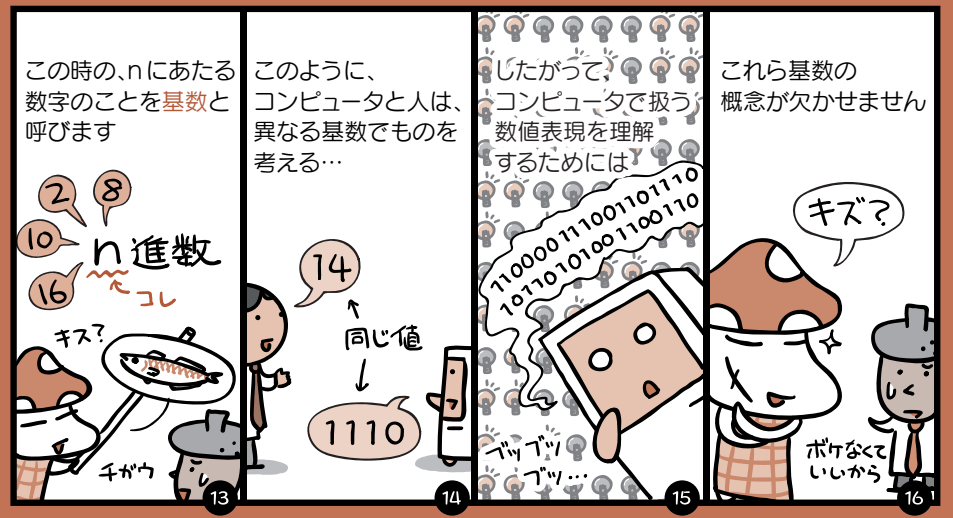
Chapter 1

基礎理論（離散数学）



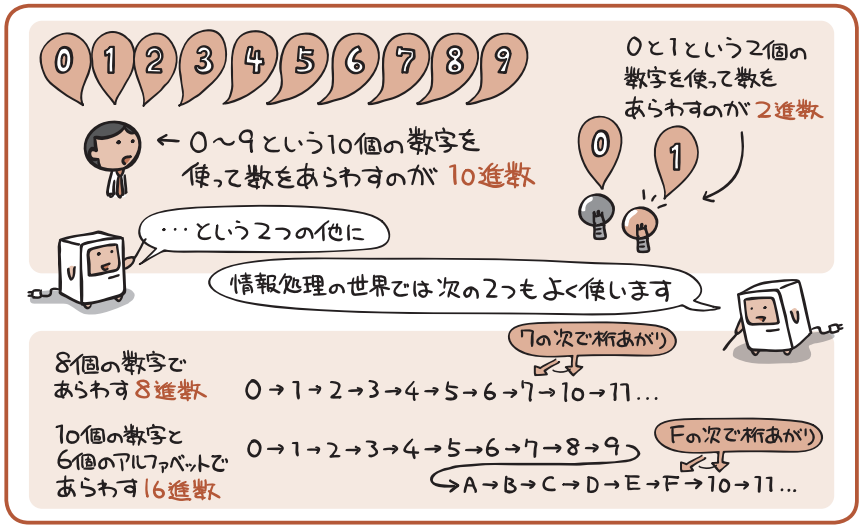






Chapter 1-1

2進数とn進数



情報処理でよく使われるn進数には、10進数、2進数、8進数、16進数などがあります。

「コンピュータといえば2進数！」はもう基本中の基本。あと他によく使われるn進数として、8進数や16進数などがあります。いや、8進数は正直あまり使いませんけど、でも情報処理の世界ではよく出てくるので無視できません。

そもそも、なぜ数の表現法をそんなに色々と併用しなきゃいけないのでしょうか？

それは、「8は2^3」「16は2^4」というところに答えが潜んでいます。

基本はやはり2進数。しかし、0と1しか使えない表記では、数をあらわすのにいちいち桁数が嵩んで仕方ありません。だから、一桁である程度まとまった区切りの数をあらわすことができて、かつコンピュータと相性の良いn進数表記が必要となります。

それがつまりは、8進数と16進数なわけ。

8は2の3乗。これなら、一桁に3ビット分の情報を持たせることができます。16なら2の4乗。一桁で4ビット分の情報をあらわすことができる。

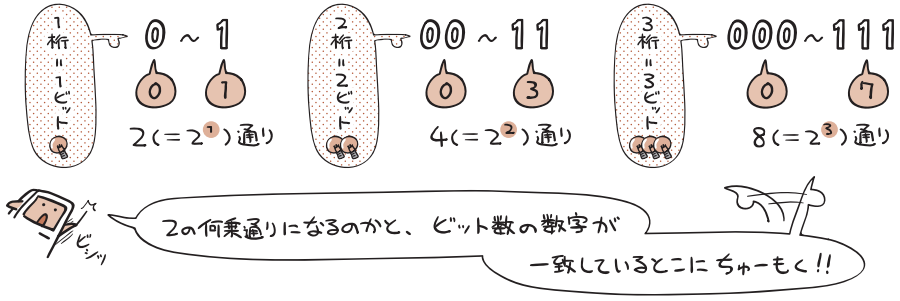
え？なんでそーなるのか？

では、これらn進数の特徴をおさらいしながら、上記の理屈を再確認していきましょう。

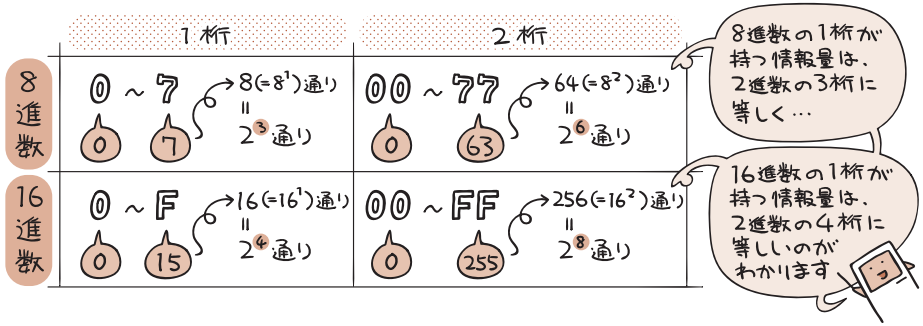
2進数と各基数との関係

2進数で数をあらわすと、次のようになります。

つまり2進数であらわすことのできる数の範囲は、桁数によって次のように変化するわけです。

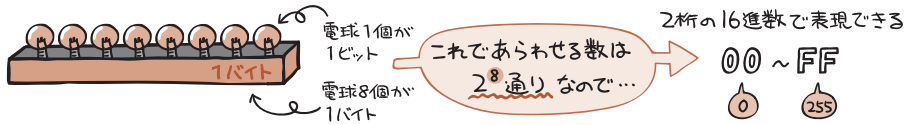


一方、8進数と16進数であらわすことのできる数は次の通り。



このことから、8進数では1桁で3ビット分、16進数では1桁で4ビット分の情報を持たせることができるというわけですね。

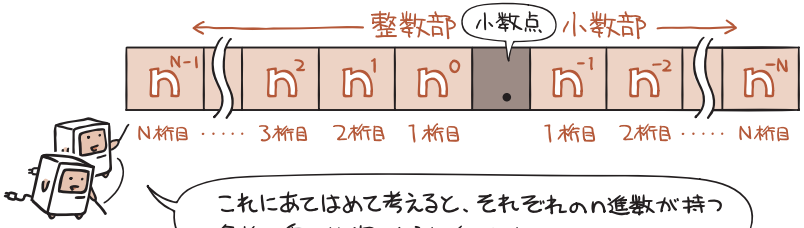
コンピュータの世界では、8ビットを1単位とするバイトという単位が一般的です。このバイトをあらわす数として、16進数がよく用いられます。

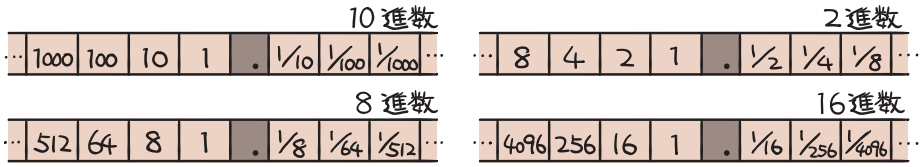


基数と桁の重み

2進数で示す数値はよく見ると、1桁あがるごとに倍々ゲームでその値が増えていくことがわかります。これを2進数が持つ各桁の**重み**といいます。

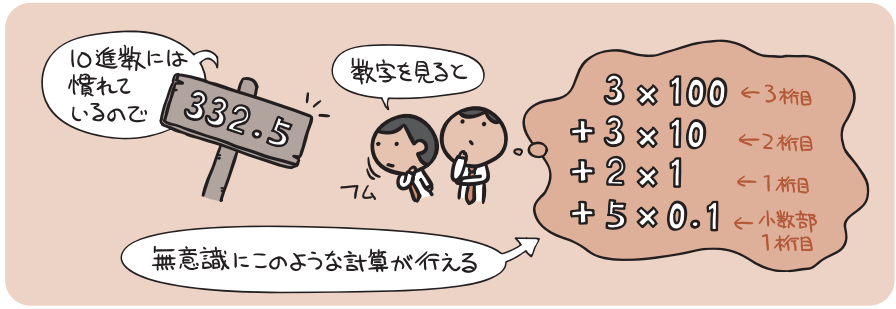
これは、「基数を累乗した数」というのがその正体で、他の基数においても同様です。したがって、n進数の持つ各桁の重みは、次の法則で決まります。





n進数では、各桁の数字に対して桁の重みをかけ算して合算することで、その数字が持つ値を導き出すことができます。

たとえば私たちは10進数には慣れているので、「332.5」という数字があると、普通に「さんびゃくさんじゅうにてんご」と認識することができます。あれも、自然と各桁の重みを使って次のような計算ができているわけです。



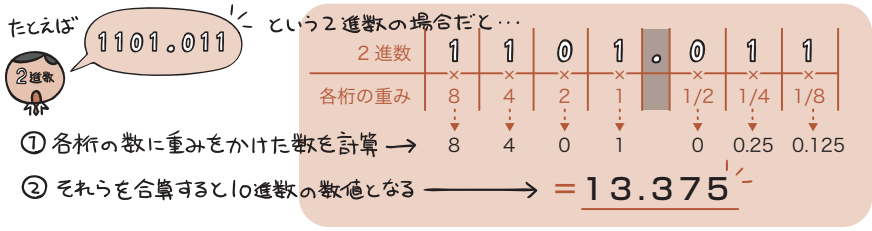
n進数と10進数間の基数変換

ある基数であらわした数値を、別の基数表現に置き換えるのが基数変換。

ここでは2進数を例に、10進数への変換と、10進数からの変換方法を見ていきます。

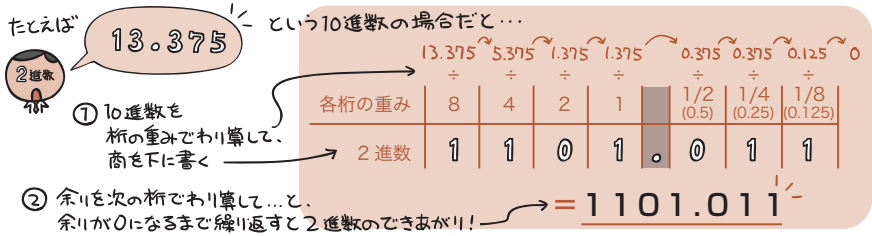
n10 n進数から10進数への基数変換

n進数から10進数への基数変換は、左ページでもふれたように、各桁の数に対して重みをかけ、それを合算することで求められます。



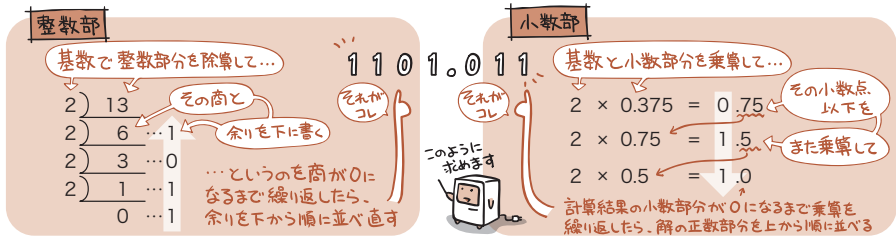
10n 10進数からn進数への基数変換（重みを使う方法）

10進数からn進数への基数変換は、上と逆の手順によって行うことができます。具体的には、10進数を桁の重みでわり算していき、その商を並べることで求められます。



10n 10進数からn進数への基数変換（わり算とかけ算を使う方法）

10進数からn進数への基数変換には、上記の他に「整数部は基数でわり算」「小数部は基数でかけ算」を行うことで求めるやり方があります。慣れてしまえば手早く計算を済ませることができますので、前述の10進数「13.375」を例に計算方法を見てみましょう。

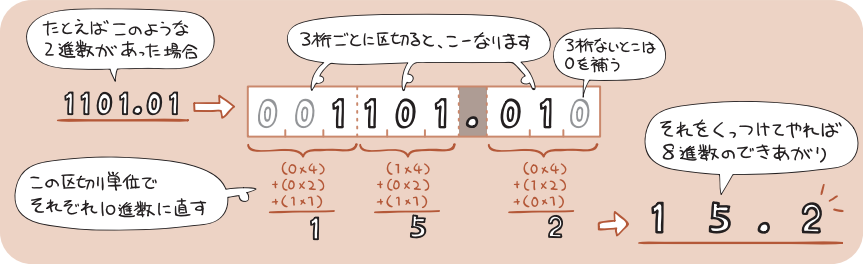


2進数と8進数・16進数間の基数変換

8進数や16進数に関しては、「コンピュータと相性の良いn進数表記」というだけあって、2進数との基数変換がもっと容易に行えます。

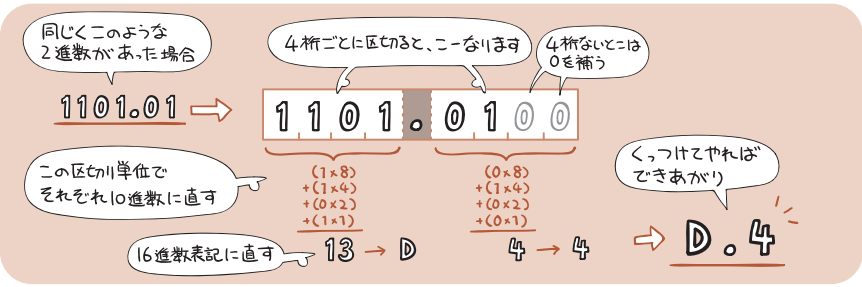
2 → 8 2進数から8進数への基数変換

8進数への変換は、2進数を3桁ごとに分けてから、8進数表記に直します。



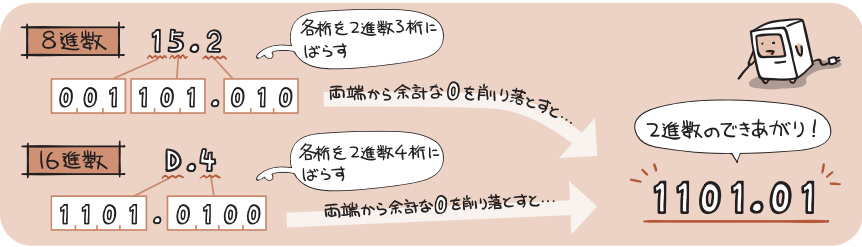
2 → 16 2進数から16進数への基数変換

16進数への変換は、2進数を4桁ごとに分けてから、16進数表記に直します。



8・16 → 2 8進数・16進数から2進数への基数変換

8進数・16進数から2進数に基数変換する場合は、上記とは逆の流れで、「8進数1桁を2進数3桁に」「16進数1桁を2進数4桁に」と分解します。



このように出題されています

過去問題練習と解説

問1 (AP-H26-S-01)

2進数で表現すると無限小数になる10進小数はどれか。

ア 0.375　イ 0.45　ウ 0.625　エ 0.75

解説　正解：イ

無限小数とは、小数部が無限である（＝有限ではない）ものです。10進小数を2進数に変換するには、下記の各選択肢のように10進数の小数部に2をかける計算を繰り返します。任意の時点において、小数部がゼロになると有限小数であり、ゼロにならず延々と計算を繰り返す場合は無限小数です。

ア
0.375 × 2 = 0.75
0.75 × 2 = 1.50
0.50 × 2 = 1.00
“0”になりました

イ
0.45 × 2 = 0.90
0.90 × 2 = 1.80 …★
0.80 × 2 = 1.60
0.60 × 2 = 1.20
0.20 × 2 = 0.40
0.40 × 2 = 0.80
★と同じになったので，計算を繰り返します

ウ
0.625 × 2 = 1.25
0.25 × 2 = 0.50
0.50 × 2 = 1.00
“0”になりました

エ
0.75 × 2 = 1.50
0.50 × 2 = 1.00
“0”になりました

問2 (AP-H26-A-19)

8ビットD/A変換器を使って，電圧を発生させる。使用するD/A変換器は，最下位の1ビットの変化で10ミリV 変化する。データに0を与えたときの出力は0ミリVである。データに16進数で 82を与えたときの出力は何ミリVか。

ア 820　イ 1,024　ウ 1,300　エ 1,312

解説　正解：ウ

問題文の条件に従って、下記のように計算します。

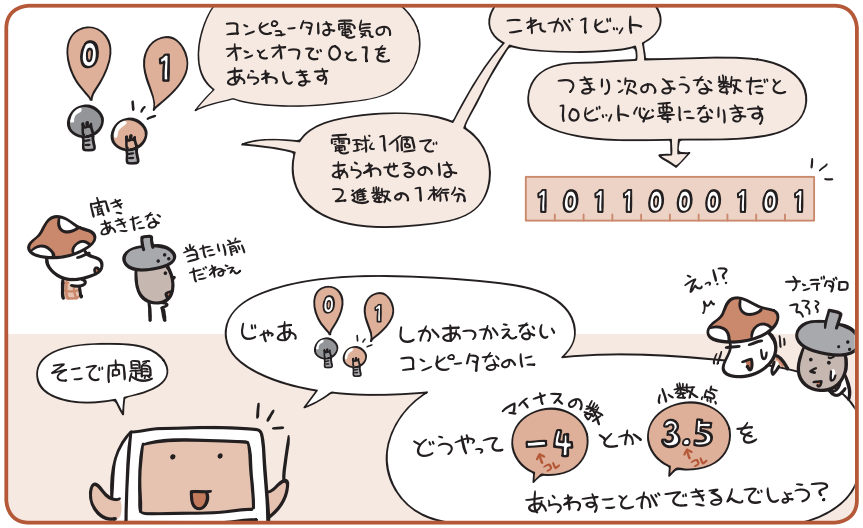
(1) 16進数の82は、8×16+2=130より、10進数130になります。

(2) 1(01)→2(10)→3(11)…と値が変化していく時、2進数の値は(当然ですが)その都度1ビットの変化が起きることになります。つまり10進数でいう130の変化を与えるとは、130ビットの変化をすることとイコールです。

(3) 問題文には、“最下位の1ビットの変化で10ミリV変化する” とあるので、130×10=1,300ミリVが出力されます。

Chapter 1-2

2進数の計算と数値表現



コンピュータは一般に、2の補数を使って負の数をあらわし、浮動小数点数で実数（小数点を含む数）をあらわします。

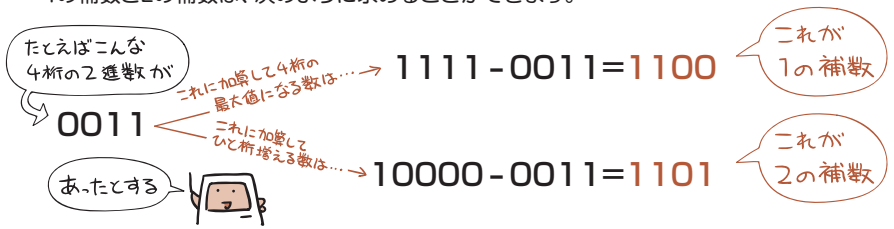
補数とは、字面の通り「補う数」という意味。補数の種類には「その桁数で最大値を得るために補う数」と「次の桁に繰り上がるために補う数」という2つの補数が存在します。

2進数では、「1の補数（その桁数での最大値）」「2の補数（桁が繰り上がる最大値）」という2つがそれに該当します。そして、コンピュータでは負の数をあらわすために、この「2の補数」を使用するのが一般的です。2の補数によって負の数をあらわすと、加算処理の回路ひとつで、加算も減算も行えるようになるので、回路をシンプルにすることができるのです。

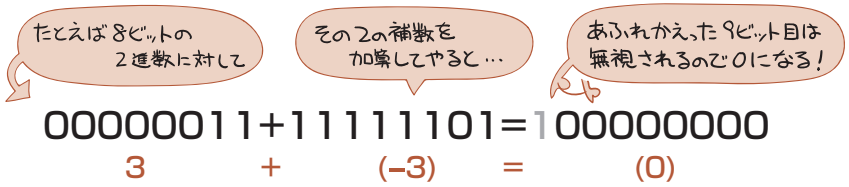
さて、もう一方の小数点はというと、コンピュータが数をあらわすやり方には、「固定小数点数」「浮動小数点数」という2つの方法があります。固定小数点数は、「あらかじめ小数点が何桁目にくるか決めてしまう」という数のあらわし方。小数点という言葉がついていますけど、コンピュータは整数をあらわすのに、こちらの方法を用いています。実は整数って、「小数点が一番後ろにある」という数に過ぎないんですね。浮動小数点数はその逆。決まったフォーマットに値を詰め込むことで、小数点の位置を固定せず数をあらわすところに特徴があります。

2の補数と負の数のあらわし方

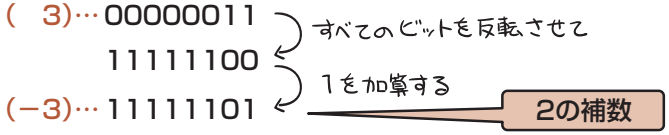
1の補数と2の補数は、次のように求めることができます。



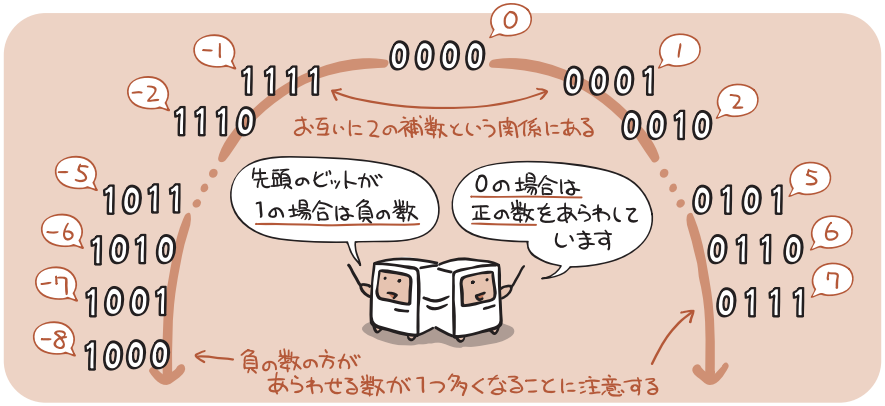
この2の補数を用いて、ある計算をしてみましょう。



このように、ある数値に対する2の補数表現は、そのまま**その数値の負の値として使える**というわけです。このことからコンピュータは一般に、負の数をあらわすのに2の補数を使います。2の補数は、次のようにすることで簡単に求めることができます。

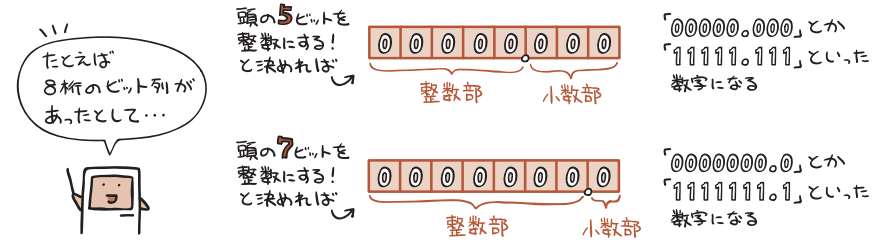


ちなみに2の補数の最上位ビットは、符号として扱うことができます。正の数と負の数は、互いに2の補数表現となる関係にあります。

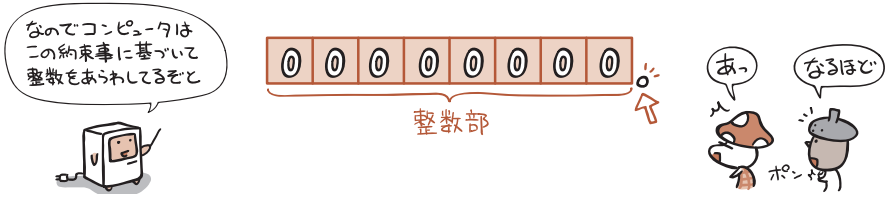


固定小数点数

固定小数点数は、「ビット列のどの位置に小数点があるか」を暗黙的了解として扱う表現方法です。



整数をあらわすには、「最下位ビットの右側を小数点とする」と決め打ちにします。これにより、小数部に割くビット数を0として、整数だけを扱うことになるわけです。



8ビットの固定小数点数であらわせる整数の範囲は、「符号なし」と「符号あり」とで、それぞれ次のようになります。

符号なし

nビットで表現できる範囲は…

$$0 \sim 2^n - 1$$

	2進数	10進数
最小	00000000	0
	00000001	1
	≀	≀
	11111110	254
最大	11111111	255

符号あり

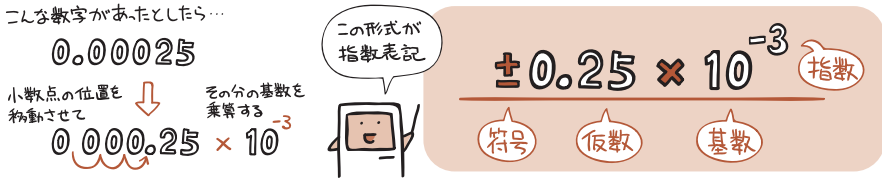
nビットで表現できる範囲は…

$$-2^{(n-1)} \sim 2^{(n-1)} - 1$$

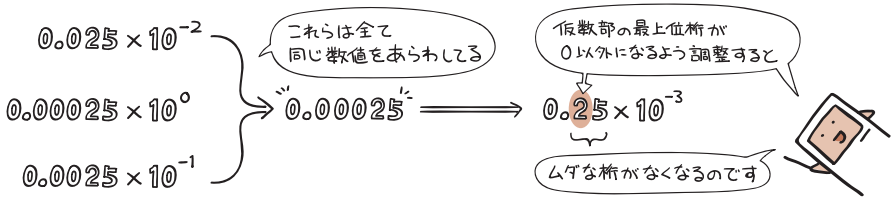
	2進数	10進数
最小	10000000	-128
	≀	≀
0	00000000	0
	≀	≀
最大	01111111	+127

浮動小数点数

浮動小数点数は、指数部と仮数部を用いた指数表記で数値を扱う表現方法です。



指数表現で用いる指数と仮数の組み合わせには、色んなパターンが考えられます。当然どれも数値としては間違いではありません。この時、「より有効桁を多く取れるように」と小数点位置を調整して、仮数部の最上位桁を0以外の数値にする作業を**正規化**といいます。



コンピュータで扱う数字は2進数なので、指数表現の中にある基数部は「2」と決まってきます。同じように、小数点より左側の整数部分も、自ずと値が決まってきます。

$$\pm m \times 2^{e}$$

符号 / 仮数 / 基数 / 指数 / ここは固定

このように固定値が入る部分はいちいち覚えておく必要がないので、コンピュータは残りの可変部分（符号、仮数、指数）をそれぞれビットに割り当てて、浮動小数点数として数をあらわします。