

2進数とn進数

0 1 2 3 4 5 6 7 8 9

← 0~9という10個の数字を使って数をあらわすのが **10進数**

0と1という2個の数字を使って数をあらわすのが **2進数**

0 1

...という2つの他に

情報処理の世界では次の2つもよく使います

8個の数字であらわす **8進数** 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 10 → 11 ...

7の次で桁あがり

10個の数字と6個のアルファベットであらわす **16進数** 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → A → B → C → D → E → F → 10 → 11 ...

Fの次で桁あがり



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が、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高くて仕方ありません。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、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。

0 → 1 → 10 → 11 → 100 → 101 → 110 → 111 ...

10進数 → 0 1 2 3 4 5 6 7

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。

1桁 = 1ビット 0 ~ 1 2 (= 2^1) 通り

2桁 = 2ビット 00 ~ 11 4 (= 2^2) 通り

3桁 = 3ビット 000 ~ 111 8 (= 2^3) 通り

2の何乗通りになるのかと、ビット数の数字が一致しているとこにちゃーもく!!

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。

	1 桁	2 桁
8 進 数	0 ~ 7 0 7 8 (= 8^1) 通り 2 ³ 通り	00 ~ 77 0 63 64 (= 8^2) 通り 2 ⁶ 通り
16 進 数	0 ~ F 0 15 16 (= 16^1) 通り 2 ⁴ 通り	00 ~ FF 0 255 256 (= 16^2) 通り 2 ⁸ 通り

8進数の1桁が持つ情報量は、2進数の3桁に等しく...

16進数の1桁が持つ情報量は、2進数の4桁に等しいのがわかります

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。

電球1個が1ビット 電球8個が1バイト

これであらわせる数は 2^8 通りなので...

2桁の16進数で表現できる 00 ~ FF 0 255

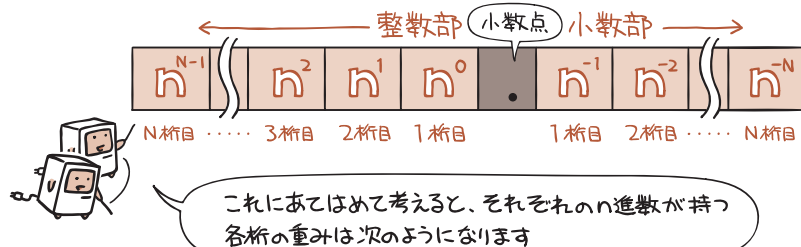
基数と桁の重み

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進数を持つ各桁の**重み**といいます。

$$1 \rightarrow 10 \rightarrow 100 \rightarrow 1000 \rightarrow 10000 \rightarrow 100000 \dots$$

10進数 → 1 2 4 8 16 32

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。



10進数	2進数
... 1000 100 10 1 . 1/10 1/100 1/1000 ...	... 8 4 2 1 . 1/2 1/4 1/8 ...
8進数	16進数
... 512 64 8 1 . 1/8 1/64 1/512 ...	... 4096 256 16 1 . 1/16 1/256 1/4096 ...

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「さんびや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くわけです。

10進数には慣れているので

332.5

数字を見ると

3 × 100 ← 3桁目
+ 3 × 10 ← 2桁目
+ 2 × 1 ← 1桁目
+ 5 × 0.1 ← 小数部 1桁目

無意識にこのような計算が行える

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。

10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。

たとえば 1101.011 という2進数の場合だと...

2進数	1	1	0	1	.	0	1	1
各桁の重み	8	4	2	1		1/2	1/4	1/8
	↓	↓	↓	↓		↓	↓	↓
① 各桁の数に重みをかけた数を計算 →	8	4	0	1		0	0.25	0.125
② それらを合算すると10進数の数値となる →	13.375							

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。

たとえば 13.375 という10進数の場合だと...

10進数	13.375	5.375	1.375	1.375	0.375	0.375	0.125	0
各桁の重み	8	4	2	1		1/2 (0.5)	1/4 (0.25)	1/8 (0.125)
① 10進数を桁の重みでわり算して、商を下に書く	1	1	0	1		0	1	1
② 余りを次の桁でわり算して...と、余りが0になるまで繰り返すと2進数のできあがり!	1101.011							

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そうですので、前述の10進数「13.375」を例に計算方法を見てみましょう。

整数部

基数で整数部分を除算して...

2 | 13 ... 1
2 | 6 ... 0
2 | 3 ... 1
2 | 1 ... 1
0 ... 1

その商と
余りを下に書く
...というのを商が0になるまで繰り返したら、余りを下から順に並べ直す

1101

小数部

基数と小数部分を乗算して...

2 × 0.375 = 0.75
2 × 0.75 = 1.5
2 × 0.5 = 1.0

その小数点、よ下も
また乗算して
計算結果の小数部分が0になるまで乗算を繰り返したら、解の正数部分を上から順に並べる

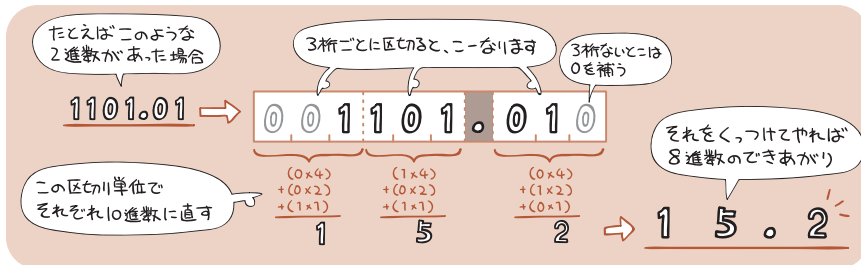
1101.011

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。

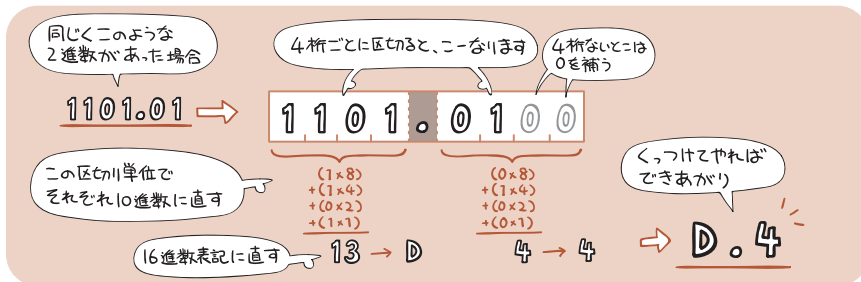
2進数から8進数への基数変換

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。



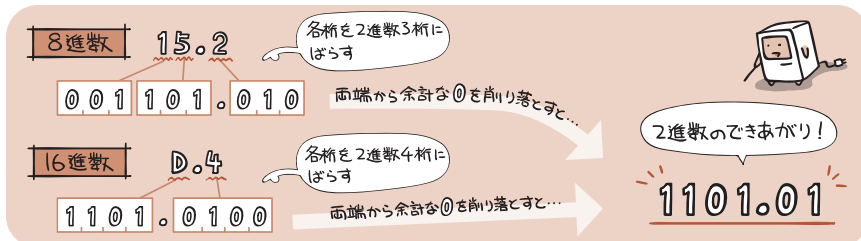
2進数から16進数への基数変換

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

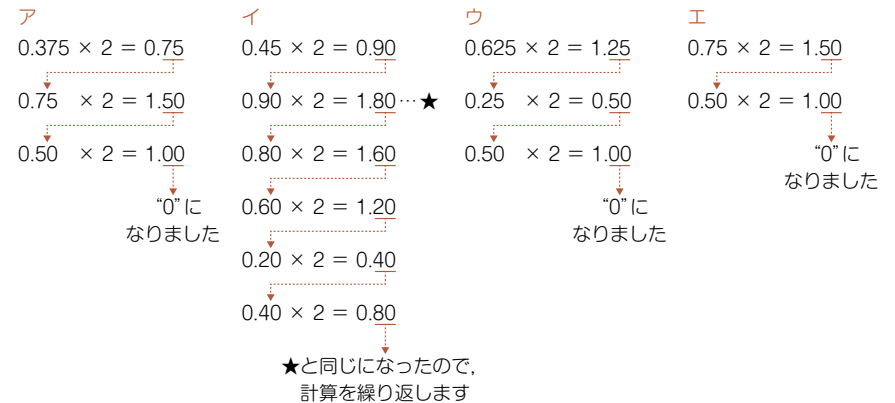
ウ 0.625

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。



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10mV変化する。データに0を与えたときの出力は0mVである。データに16進数で82を与えたときの出力は何mVか。

ア 820

イ 1,024

ウ 1,300

エ 1,312

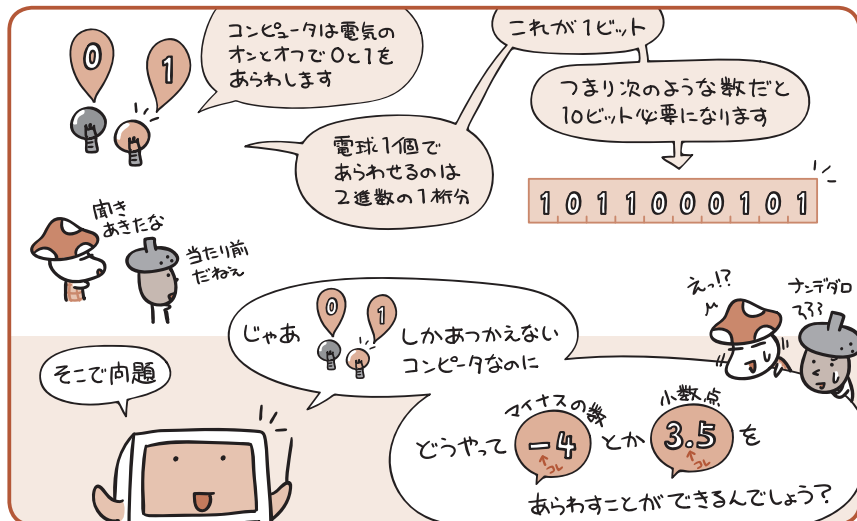
解説

正解: ウ

問題文の条件に従って、下記のように計算します。

- 16進数の82は、 $8 \times 16 + 2 = 130$ より、10進数130になります。
- $1(01) \rightarrow 2(10) \rightarrow 3(11) \dots$ 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。
- 問題文には、「最下位の1ビットの変化で10mV変化する」とあるので、 $130 \times 10 = 1,300$ mVが出力されます。

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、もう一方の小数点とは、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ある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が、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。

たとえばこんな4桁の2進数が

0011

これが1の補数

1111 - 0011 = 1100

これが2の補数

10000 - 0011 = 1101

あ、たとする

この2の補数を用いて、ある計算をしてみましょう。

たとえば8ビットの2進数に対して

00000011 + 11111101 = 100000000

3 + (-3) = (0)

その2の補数を加算してやると...

あふれかえた9ビット目は無視されるので0になる!

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。

(3) ... 00000011

11111100

(-3) ... 11111101

すべてのビットを反転させて

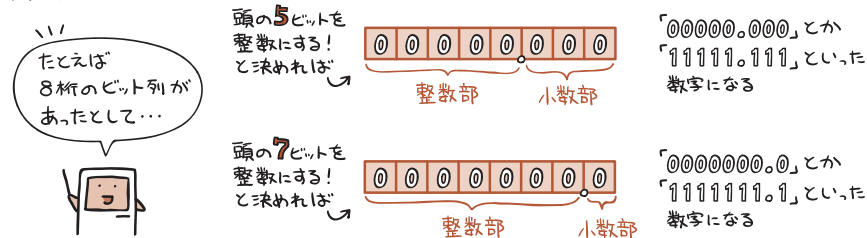
1を加算する

2の補数

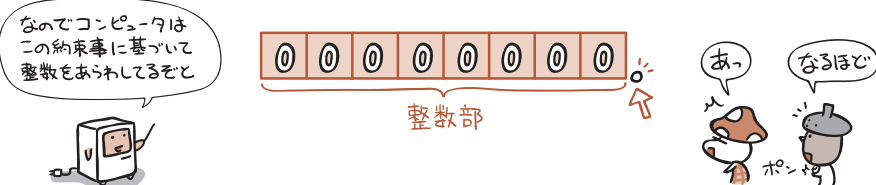
ちなみに2の補数の最上位ビットは、符号として扱うことができます。正の数と負の数は、互いに2の補数表現となる関係にあります。

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

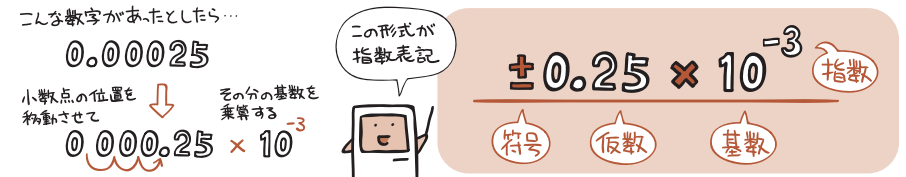
nビットで表現できる範囲は…

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

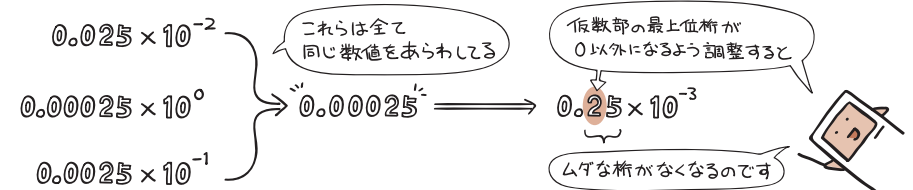
	2進数	10進数
最小	10000000	-128
	}	}
	00000000	0
	}	}
最大	01111111	+127

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

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。

