

2進数と16進数

山本昌志*

2005年6月2日

1 本日の学習内容

いよいよC言語の最難関であるポインタの学習を始めることになる。ポインタについての深い知識を得ようとする、メモリーのことを理解しなくてはならない。メモリーのことを理解するためには、2進数と16進数が分からなくてはならない。そこで、本日は2進数と16進数について説明する。

諸君の大部分の者は、コンピューター内部では2進数が使われていることを知っていると思う。2進数が使われているので、その取り扱いになれる必要がある。16進数は2進数の親戚で、表示に便利なのでコンピューターの世界でよく使われる。そのため、16進数も学習する。コンピューターで2進数が使われる理由は、付録の5.1に示してある。

2 位取り記数法

2.1 数の表現

通常使われている10進数では、0~9までの10個の数字を使って、数を表現する。9の次は10で桁が上がる。10進数以外にいろいろな数の数え方がある。2進数、10進数、16進数の数の表現を図1に示す。合わせて、桁上がりという考えの無い漢字や楔形文字、ローマ数字も併記する。桁上がりという考えが無い表記の場合、桁が上がるたびに、新しい記号が必要であることがわかる。大きな数字を表す場合、非常に不便である。また、筆算を用いた計算もできない。

*独立行政法人 秋田工業高等専門学校 電気情報工学科

原始人	2進法	10進法	16進法	漢字	楔形文字	ローマ数字
	0	0	0			
•	1	1	1	一	▼	I
••	10	2	2	二	▼▼	II
•••	11	3	3	三	▼▼▼	III
••••	100	4	4	四	▼▼▼▼	IV
•••••	101	5	5	五	▼▼▼▼▼	V
••••••	110	6	6	六	▼▼▼▼▼▼	VI
•••••••	111	7	7	七	▼▼▼▼▼▼▼	VII
••••••••	1000	8	8	八	▼▼▼▼▼▼▼▼	VIII
•••••••••	1001	9	9	九	▼▼▼▼▼▼▼▼▼	IX
••••••••••	1010	10	A	十	◀▼▼▼▼▼▼▼▼	X
•••••••••••	1011	11	B	十一	◀▼▼▼▼▼▼▼▼▼	XI
••••••••••••	1100	12	C	十二	◀▼▼▼▼▼▼▼▼▼▼	XII
•••••••••••••	1101	13	D	十三	◀▼▼▼▼▼▼▼▼▼▼▼	XIII
••••••••••••••	1110	14	E	十四	◀▼▼▼▼▼▼▼▼▼▼▼▼	XIV
•••••••••••••••	1111	15	F	十五	◀▼▼▼▼▼▼▼▼▼▼▼▼▼	XV
••••••••••••••••	10000	16	10	十六	◀▼▼▼▼▼▼▼▼▼▼▼▼▼▼	XVI
•••••••••••••••••	10001	17	11	十七	◀▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼	XVII
••••••••••••••••••	10010	18	12	十八	◀▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼	XVIII
•••••••••••••••••••	10011	19	13	十九	◀▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼	XIX

図 1: 数の数え方

2.2 現代の数の表現 (位取り記数法)

現代の便利な数の表現は桁上がりの考えがあるからこそである。この桁上がりの考えは、ゼロが発見されたので可能となった。このように桁上がりの考えで、数を示すのが位取り記数法 (place value system) である。10進数は9の次で桁上がりが生じ10となる。0~9の数字を使って、数を表すのである。この10を基数と、0~9間での数を底と言う。10進数の他、いろいろの基数の数が考えられるが、コンピューター科学で使われるのは、主に2進数と16進数である。それぞれの基数と底を表1に示す。

表 1: 基数と底

数の表現	基数	底
2進法	2	0, 1
10進法	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
16進法	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

図1や表1から、自然数の数え方は理解できたと思う。そうすると相互の変換ができれば、ある程度それを応用することができるようになる。それぞれの変換を考える前に、数の表記方法について、勉強することにする。位取り記数法での数の表し方が理解できれば、それぞれの変換が分かるはずである。

例えば、今年は2005年である。10進法の2005の表記はどのような意味があるか？。これは、次のように解釈する。大げさではあるが、こう解釈すると、他の基数の数字の意味ははっきりする。

$$(2005)_{10} = (2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 5 \times 10^0) \quad (1)$$

括弧の下の10は10進法の意味で、非常に簡単な話である。これさえ、分かれば基数の変換なんか、怖くない。この式の右辺の \times と 10^p をとって、それを並べたのが位取り記数法である。

コーヒーブレイク

ゼロがない時代は、大変だった。ゼロが無いと、式(1)の左辺のような表記は出来ない。その0(ゼロ)が発見されたのは、6世紀頃のインドと言われている。西暦0年がないのは、このためである。キリストが生まれた頃は、ゼロがなかったのである。

3 基数の変換

3.1 2進数と10進数の変換

3.1.1 2進数 → 10進数

2進数から10進数への変換は簡単である。式(1)を理解していれば、分かる。2進数であろうが10進数であろうが、表記法は同じで、約束に従って変形すれば良い。次のようにする。

$$\begin{aligned} (10011)_2 &= (1 \times 10^{100} + 0 \times 10^{11} + 0 \times 10^{10} + 1 \times 10^1 + 1 \times 10^0)_2 \\ &= (1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10} \\ &= (16 + 0 + 0 + 2 + 1)_{10} \\ &= (19)_{10} \end{aligned} \quad (2)$$

この手中を順を追って説明すると、以下のようになる。

1. まずは、1行目右辺のように位取り記数法で表現する。
2. そうして、表1に従い、式を変換したい基数に直す。
3. 後は地道に計算するだけ。

通常は、1行目の右辺は省き、2行目から計算する。

- 2進数の各桁の10進数での値(重み)を覚えておくと便利である。

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096

3.1.2 10進数 → 2進数

今度は逆に10進数から2進数への変換である。原理的に、先ほどと同じように変換ができるが、計算してみるとそれは難しい¹。これは、我々は10進数の演算になれていることが原因となっている。自然には、10進数であろうと2進数であろうと優位性は無いからである。

10進数から2進数への計算手法は簡単であるが、その内容を理解することが大事である。計算手法を忘れても、内容が理解できていれば、その方法はいつでも自分で作ることができる。また、応用範囲も広がる。では、簡単な例で説明する。10進数の $(19)_{10}$ を2進数に変換する方法を示す。具体的には、19を

$$(19)_{10} = (\cdots + a_4 \times 2^4 + a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0)_{10} \quad (3)$$

と表現したい。これは式(2)の2行目の式で、ここで求められた係数を $(\cdots a_4 a_3 a_2 a_1 a_0)_2$ と並べれば位取り記数法になる。それぞれ、 a_n を求めなくてはならない。そこで、次のように19を2で割った商と余りを考える。これは、

$$(9 \times 2 + 1)_{10} = (\cdots + a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0)_{10} \times 2 + a_0 \quad (4)$$

と書ける。これをよくにらむと、 $a_0 = 1$ ということが分かる。すなわち、 a_0 は19を2で割ったあまりである。残りの部分は、

$$(9)_{10} = (\cdots + a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0)_{10} \quad (5)$$

となることも分かるだろう。商について同じことをすると、

$$(4 \times 2 + 1)_{10} = (\cdots + a_5 \times 2^3 + a_4 \times 2^2 + a_3 \times 2^1 + a_2 \times 2^0)_{10} \times 2 + a_1 \quad (6)$$

となる。したがって、 $a_1 = 1$ である。しつこいが、さらに商について同様に進めると、

$$(2 \times 2 + 0)_{10} = (\cdots + a_6 \times 2^3 + a_5 \times 2^2 + a_4 \times 2^1 + a_3 \times 2^0)_{10} \times 2 + a_2 \quad \Rightarrow \quad a_2 = 0 \quad (7)$$

$$(1 \times 2 + 0)_{10} = (\cdots + a_7 \times 2^3 + a_6 \times 2^2 + a_5 \times 2^1 + a_4 \times 2^0)_{10} \times 2 + a_3 \quad \Rightarrow \quad a_3 = 0 \quad (8)$$

$$(0 \times 2 + 1)_{10} = (\cdots + a_8 \times 2^3 + a_7 \times 2^2 + a_6 \times 2^1 + a_5 \times 2^0)_{10} \times 2 + a_4 \quad \Rightarrow \quad a_4 = 1 \quad (9)$$

となる。最後の式から、 $a_n = 0$ ($5 \leq n$)が分かる。以上をまとめると

$$\begin{aligned} (19)_{10} &= (a_4 \times 2^4 + a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0)_{10} \\ &= (1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10} \\ &= (10011)_2 \end{aligned} \quad (10)$$

となる。要するに、2で割ったあまりを書いていけば良いのである。計算方法は分かった。だがこの方法は実際的ではない。よく使われるのは、図2のように2で割った余りを並べる。これは、 $(19)_{10} = (10011)_2$ 、 $(2003)_{10} = (11111010011)_2$ を示している。

¹本当に難しいか、試して見よ

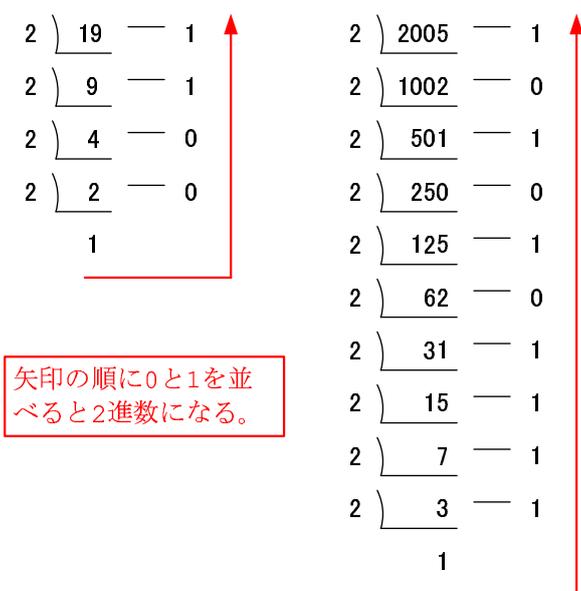


図 2: 10 進数から 2 進数への変換方法。

内容を十分理解し、変換の練習をしなくてはならない。

3.2 16 進数

2 進数の変換が理解できたら、16 進数の変換はまったくもって簡単である。

3.2.1 16 進数 → 10 進数

2 進数と同じで次のようにする。

$$\begin{aligned}
 (376)_{16} &= (3 \times 10^2 + 7 \times 10^1 + 6 \times 10^0)_{16} \\
 &= (3 \times 16^2 + 7 \times 16^1 + 6 \times 16^0)_{10} \\
 &= (3 \times 256 + 7 \times 16 + 6 \times 1)_{10} \\
 &= (886)_{10}
 \end{aligned} \tag{11}$$

3.2.2 10 進数 → 16 進数

これも 2 進数と同様に考える。16 で割ったあまりを並べれば良い。図 3 のようにして、 $(25391)_{10} = (632F)_{16}$ を計算する。

$$\begin{array}{r}
 16 \overline{) 25391} \text{ --- } 15 \text{ --- } F \\
 16 \overline{) 1586} \text{ --- } 2 \text{ --- } 2 \\
 16 \overline{) 99} \text{ --- } 3 \text{ --- } 3 \\
 \phantom{\text{ --- }} 6 \text{ --- } 6 \text{ --- } 6
 \end{array}$$

図 3: 10 進数から 16 進数への変換方法

手計算ではこの方法を用いるが、実際のエンジニアは電卓の変換機能を使う。

コーヒーブレイク

昔から言われるジョークをひとつ。プログラマーは、クリスマス (12 月 25 日) とハロウィーン (10 月 31 日) が区別できない。なぜか?。ヒント

- 10 進数 (decimal number) のことを DEC と書く。DEC 23 と書けば、10 進数の 23 をあらわしている。
- 8 進数 (octal number) はのことを OCT とかく。OCT 23 と書けば、8 進数の 23 を表している。

3.3 2 進数と 16 進数の変換

2 進数と 16 進数の相互の変換は簡単である。 $2^4 = 16$ なので、2 進数の 4 桁は 16 進数の一桁に対応している。図 4 のように、1 桁の 16 進数を 4 桁の 2 進数に変換すれば良い。反対に 4 桁の 2 進数は、1 桁の 16 進数に変換できる。

$$\begin{array}{r}
 \text{16進数} \quad \quad \quad \text{B} \quad \quad \quad \text{7} \\
 \quad \quad \quad \leftarrow \quad \quad \quad \rightarrow \quad \quad \quad \leftarrow \quad \quad \quad \rightarrow \\
 \text{2進数} \quad \quad \quad 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \\
 \quad \quad \quad \uparrow \ \uparrow \\
 \quad \quad \quad 8 \ 4 \ 2 \ 1 \ 8 \ 4 \ 2 \ 1 \\
 \\
 (\text{B})_{16} = (11)_{10} = (8+2+1)_{10} \quad (\text{7})_{16} = (7)_{10} = (4+2+1)_{10}
 \end{array}$$

図 4: 2 進数と 16 進数の相互変換方法

4 課題 (レポート)

4.1 記数法の変換

以下の練習問題を解き、レポートとして提出すること。

問題の指示に従い、自然数の基数の変換を行え。

[問題 1] 以下の 10 進数を 2 進数と 16 進数へ変換せよ。

- $(1)_{10}$
- $(2)_{10}$
- $(4)_{10}$
- $(8)_{10}$
- $(16)_{10}$
- $(32)_{10}$
- $(65536)_{10}$
- $(2004)_{10}$
- $(999)_{10}$
- $(698)_{10}$

[問題 2] 以下の 2 進数を 10 進数と 16 進数へ変換せよ。

- $(1)_2$
- $(10)_2$
- $(100)_2$
- $(1000)_2$
- $(1010)_2$
- $(11111)_2$
- $(10101111)_2$
- $(10010101)_2$
- $(10101010)_2$
- $(11111111)_2$

[問題 3] 以下の 16 進数を 2 進数と 10 進数へ変換せよ。

- $(8)_{16}$

- $(F)_{16}$
- $(1F)_{16}$
- $(AF)_{16}$
- $(F98)_{16}$
- $(89AB)_{16}$
- $(CDEF)_{16}$
- $(4E3B)_{16}$
- $(FFFF)_{16}$
- $(A000)_{16}$

4.2 レポート提出要領

提出方法は、次の通りとする。

期限	6月9日(木)PM5:00まで
用紙	A4
提出場所	山本研究室の入口のポスト
表紙	表紙を1枚つけて、以下の項目を分かりやすく記述すること。 授業科目名「情報工学概論」 課題名「課題2 基数の変換」 2E 学籍番号 氏名 提出日
内容	問題の解答。計算課程をきちんと書くこと。

5 付録

5.1 コンピューターで2進数が使われる理由

人間の指は10本あることは、よく知られている。そのため、人類は10進法を使っていると言われている。小学校の低学年では指を使って計算する子供がいることから分かる。コンピューターの内部のハードウェアでは、電圧が0Vか5V(もっと低い場合もある)でデータやプログラムを表現している。指が2本しかないのと同じ。だから、コンピューターは2進法を使う。2進法を使うメリットに、何があるか? という疑問が湧くであろう。その答えとして、以下のようなことが考えられる。

- ノイズに強い

0~5Vで動作する素子からできたコンピューターを考える。2ビットと10ビットの場合、割り当てられる電圧のレベルは、図5の通りである。図からも分かるように、許されるノイズは、2ビットの方が格段に大きくなる。1ビットのエラーも許されないデジタルコンピューターにおいては、この差は非常に大きい。

- ハードウェアを実現するのが容易

コンピューター内部には、単純な動作をする同じような部品が数多くある。2進数であれば、入力は0と1、出力も0と1なので、構成する1個の部品が非常に単純になる。要するに2進数を採用すると、部品が簡単になるのである。

- 演算が簡単

例えば、掛け算九九を考えると分かる。10進数だと、0~9までの掛け算、合計100通りある。2進数だと、4通りで済む。

- ブール代数が使える、論理演算が容易

ブール代数については、他の授業で勉強することになっている。それまで待てない人は、私の講義ノート²でも見る。

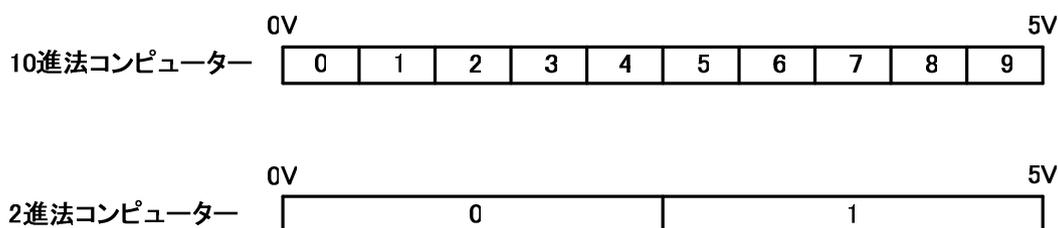


図 5: 2進法と10進法のコンピューターのノイズレベル

²http://www.akita-nct.jp/yamamoto/lecture/2003/2E/boolean_algebra/index.html

5.2 コンピューター内部での整数以外の表現

これは少しレベルの高い内容で、もう少し時間をかけないと説明できない。2年前、私が2年生で他の教科を教えていたときには、ここまで講義³した。従って、諸君が理解できないわけではない。将来、情報関係の仕事に就きたい者、あるいは自分の能力に自信のある者は独力で以下の内容を理解せよ。非常におもしろい内容であるはずである。

5.3 負の整数の表現

負の整数は、補数 (complement) を使って、コンピューター内部では表現される。それを図6に示すが、手順は、次の通りである。

1. 絶対値を2進数のビットパターンで表現し、その反転を行う。
2. 反転されたビットパターンに1を加算する。

このようにしてできたビットパターンをメモリーに記憶させ、それを負の数として取り扱う。

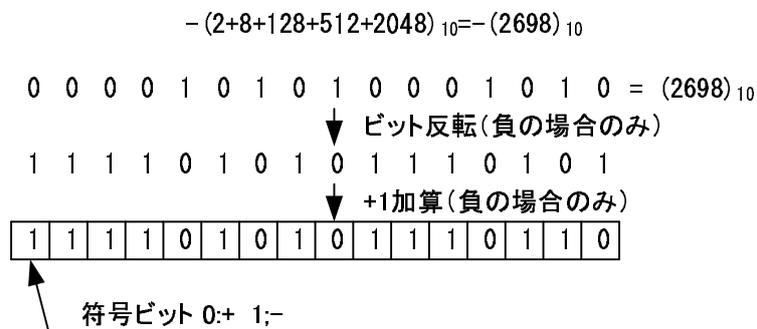


図 6: 負の整数をメモリーに格納する方法

この方法のメリットは、減算が加算器でできることである。補数表現のイメージは、図7の通りです。車の距離計に似ている。

³<http://www.akita-nct.jp/yamamoto/lecture/2003/2E/index.html>

表現している数 (10進数)	計算機の内部表現(2進数)																(16進数)				
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	3
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
- 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	F	F	F	F
- 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	F	F	F	E	
- 3	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	F	F	F	D		
- 4	1	1	1	1	1	1	1	1	1	1	1	1	0	0	F	F	F	C			

図 7: 距離計イメージ (2の補数表示)

それでは、なぜ、補数表現だと、減算が加算器で可能なのだろうか？。減算の演算は、負の数の加算と同じである。したがって、図7のように負の数を表示すると、負の整数の加算は正の整数の加算と同じと分かるであろう。したがって、加算器で減算が可能となる。実際、正の数の減算を行うときは、ビットの反転と+1加算を実施して、加算器で計算する。イメージは、図7の通りであるが、もう少し、理論的に説明をおこなうとしよう。ある正の整数を x とする。その負の数、 $-x$ は補数表現では、

$$[-x] = (FFFF - x + 1)_{16} \quad (12)$$

となる。左辺の $[-x]$ が $-x$ の意味である。[] の意味は、括弧内の負の整数を計算機内部の表現を表している。これは、私が作った表記なので、一般には用いられていない。右辺の $FFFF - x$ がビット反転になっている。ここでは、16ビットで整数を表現しようとしているので、 $FFFF$ から x を引いてビット反転させている。疑問に思う者は実際に計算して見よ。それに1を加えて、補数の表現としている。つぎに、ある整数 y を考えて、 $y-x$ を計算してみよう。

$$[y - x] = (y + FFFF - x + 1)_{16} \quad (13)$$

$FFFF - x + 1$ は、あらかじめ計算されて、コンピューター内部のメモリーに格納されているので、 $[y - x]$ は加算器で可能である。これは、あたりまえです。重要なことは、この結果が、負の場合、2の補数表現になっており、正の場合、そのままの値になっていることである。

演算の結果、 $y-x$ が負になる場合を考えよう。すると式 (13) は、

$$[y - x] = (FFFF - (x - y) + 1)_{16} \quad (14)$$

と変形できる。この場合、絶対値が $(x - y)$ なので、絶対値のビット反転と+1 加算となっていることが理解できる。つぎに、 $y - x$ が正になる場合を考えましょう。すると式 (13) は、

$$\begin{aligned} [y - x] &= (y - x + FFFF + 1)_{16} \\ &= (y - x + 10000)_{16} \end{aligned} \tag{15}$$

となる。10000 は計算機内部では、桁上りを示す。16 ビットの表示では無視される。したがって、内部の表現は、正しく表せる。

コーヒーブレイク

この方法で負の数を表すことは、1970 年頃には常識となったようです。驚いたことに、負の数をこの補数で表すアイデアは、パスカルが最初です。パスカルは、パスカリーヌという歯車式計算機を 1642 年頃に製作しています。その減算を加算器で行うために、補数というものを考えたようです。

5.4 浮動小数点表示

浮動小数点表示とは、指数化 (例えば、 -0.123×10^{-2}) して数値を表現する。これは非常に便利な方法で、自然科学では多くつかわれる。コンピューターでも同様で、データが整数と指定されない限りこの浮動小数点が用いられる。実際、この仮数部の (-0.123) と指数の (-2) をメモリーに格納する。この方法の長所と短所は、以下の通りである。

長所 決められたビット数内で、非常に小さな数値から大きな数値まで表現可能になる。

短所 桁落ち誤差が発生する可能性がある。

浮動小数点表示を学習するために、必要な言葉の意味は、図 8 の通りである。1 年生の数学の授業で学習したはず。

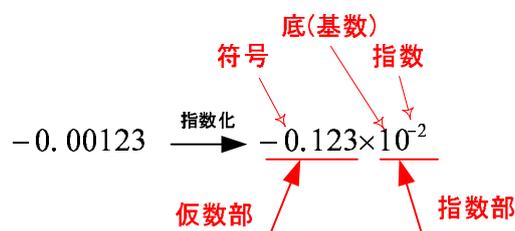


図 8: 指数表現の名称

IEEE の規格の C 言語の倍精度実数型の double の表現について説明する。まず、浮動小数点表示のための正規化を図 5 に示す。当然、仮数部、指数部とも 2 進数表現です。仮数部は、符号と 1.XXXX のように表す。

$$(-0.0007696151\ 733398438)_{10} = (-1.100100111 \times 10^{-1011})_2$$

図 9: IEEE 規格表現のための規格化

つぎに、これを IEEE 規格の浮動小数点に表すことを考える。まずその規格の仕様は、以下のようになっている。

- 64ビット (第0ビット～第63ビット) で、浮動小数を表わす。各ビットの構成は、図10の通りである。
- 最上位の第63ビットが仮数部の符号ビットである。正の場合ゼロで、負の場合1になる。
- 指数は11ビットでオフセットバイナリ方式で表す。11ビットで0～2047の値になる。ただし、指数部11ビットの値0と2047は例外処理のために予約されている。11ビットで表現される値からオフセット値1023を引くことにより指数の値が-1022～1023の範囲になるように定められている。
- 仮数部は52ビットである。小数点以下を、絶対値で表現する。規格化のための整数部は1と分かっているため、このためのビットは割り当てられていない。

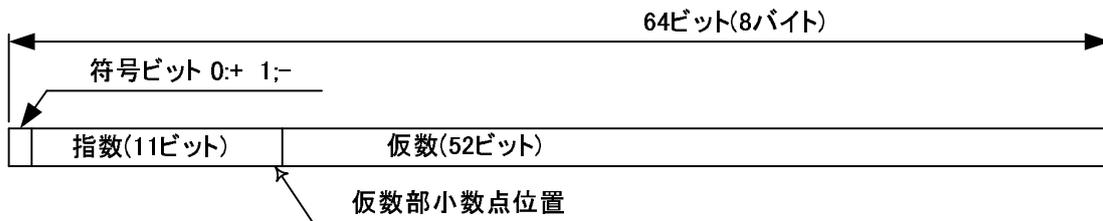


図 10: IEEE 規格 (C 言語の倍精度実数) 表現のビットの内訳

以上の仕様をもとに、図9で規格化された数を浮動小数点表示を示す。ほとんどの部分は規格化で分かるが、指数のみ計算が必要である。指数は、オフセットバイナリで計算するために、まず10進数で表す。

$$(-1011)_2 = (-8 - 2 - 1)_{10} = (-11)_{10} \quad (16)$$

不動小数点表示の指数は、この式の値に1023を加算して求める。すると、

$$(-11 + 1023)_{10} = (1012)_{10} = (1111110100)_2 \quad (17)$$

となる。

これで、すべて準備が整った。不動小数点表示は、図11のようになる。実際のコンピューターには、この64ビットのデータが格納される。メモリーは8ビット(1バイト)毎アドレスが割り当てられているので、8番地分のデータ領域が必要である。

