■6 群（コンピュータ- 基礎理論とハードウェア）- 4 編（コンピュータアーキテクチャ（I）標準的）

2 章 データ形式

【本章の構成】

本章では，データ形式として 2 の補数（2-1 節），BCD（2-2 節），ASCII コード（2-3 節），及び IEEE 754（2 進浮動小数点数データ形式）（2-4 節）について，それぞれ述べる．

■6 群 - 4 編 - 2 章

2-1 2 の補数

（執筆者：井上弘士）[2011 年 9 月 受領]

符号付き整数を 2 進数で表現するために用いられる．単純な符号付き 2 進整数の表現法としては，プラスかマイナスの符号を示すビット（符号ビットと呼ぶ）を数値の前に付し，それに続くビット列が絶対値を表す「符号付き絶対値表現」がある．通常，符号ビットが‘0’のときはプラス，‘1’のときはマイナスとなり，符号ビットは最上位ビット（最も左に位置するビット）に割り当てられる．2 進 n 桁の符号付き絶対値で表現できる最大値は $2^{n-1}-1$，最小値は $-2^{n-1}+1$ である．この表現法は直感的に分かりやすい反面，値ゼロの表現が 2 種類（＋ 0 と－ 0）存在するため，一般には＋ 0 と－ 0 が何れも同じ値ゼロであるとみなすための工夫が必要となる．これに対し「2 の補数表現」では，値ゼロの表現が一意に定まる．また，加算が非常に簡単となるといった利点がある．演算対象となる 2 つの値をそのまま加え，最上位ビットからの桁上がりをそのまま捨てるだけでよい．このような理由により，現在ほとんどのコンピュータでは符号付き整数を表現するための形式として 2 の補数表現が採用されている．

表 2·1　10 進整数と 2 進整数の対応

10 進整数表現	2 進整数表現	
	符号付き絶対値	2 の補数
＋ 7	0111	0111
＋ 6	0110	0110
＋ 5	0101	0101
＋ 4	0100	0100
＋ 3	0011	0011
＋ 2	0010	0010
＋ 1	0001	0001
＋ 0	0000	0000
－ 0	1000	
－ 1	1001	1111
－ 2	1010	1110
－ 3	1011	1101
－ 4	1100	1100
－ 5	1101	1011
－ 6	1110	1010
－ 7	1111	1001
－ 8		1000

2 の補数では，2 進 n 桁の負の整数「－ N」を「－ N ＋ 2^n」で表現する．n 桁の 2 進数で表現できる $2n$ 個の数値のうち，最初の半分となる“000... 0”から“011... 1”までがゼロと正の数に，残り半分である“100... 0”から“111... 1”までが負の数に割り当てられる．**表 2·1** に，10 進整数－ 7～＋ 7 に対応する符号付き絶対値表現と 2 の補数表現（共に 4

桁の 2 進数）を示す．正の整数に関しては，2 の補数表現と符号付き絶対値表現は同じである．同じ桁数の符号付き絶対値表現と比較した場合，2 の補数表現では，　値ゼロの表現が「すべて‘ 0 ’」の一意に定まる，　1 つ多くの数値を扱うことができる，といった違いがある．なお，2 の補数表現においても最上位ビットは符号ビットとして機能しており，‘ 0 ’のときはプラス，‘ 1 ’のときはマイナスを表す．したがって，最上位ビットを見るだけで正か負かを判断をすることができる．

正の 10 進整数「＋ N」をビット幅 n（つまり 2 進 n 桁）で表現するには，N を 2 進数に変換し，余った上位の桁には‘ 0 ’を補填するだけでよい．一方，負の 10 進整数「－ N」を 2 の補数で表現するには，表現したい整数の絶対値 N を 2 進数に変換し，格桁を反転する（これにより得られた値を「一の補数」という）．そして，求めた値に‘ 1 ’を加算する．最後に，余った上位の桁には‘ 1 ’を補填すればよい（これを符号拡張という）．

■6 群 - 4 編 - 2 章

2-2 BCD

（執筆者：井上弘士）［2011 年 9 月 受領］

我々の日常生活において数値を表す場合は主に 10 進法が用いられる．これに対し，コンピュータでは 2 進法で数値を表現するのが一般的である．そこで，コンピュータにおいても何らかの形で 10 進法に基づく数値を表現したいといった要求が出てくる．これを可能にするのが BCD（Binary Coded Decimal）であり，「2 進化 10 進表現」や「2 進化 10 進符号」などと呼ばれる．2 進法で表現した数値の 4 桁を用いて 10 進数の 0～9 までを表現するものであり，その対応を**表**に示す．実際には 2 進数 4 桁で 16 種類の数値を表すことが可能であるが，BCD では 10 種類に対応するコード部分のみを使用する．例えば，10 進数「234」を BCD 表現すると「0010_0011_0100」となる．

表 2·2　BCD **対応表**

10 **進表現**	0	1	2	3	4
BCD **表現**	0000	0001	0010	0011	0100
10 **進表現**	5	6	7	8	9
BCD **表現**	0101	0110	0111	1000	1001

■6 群 - 4 編 - 2 章

2-3 ASCII コード

（執筆者：井上弘士）［2011 年 9 月 受領］

文字を表現するための代表的な標準コードである．ASCII（American Standard Code for Information Interchange，略称「アスキー」と呼ばれる）は，米国規格協会（ANSI）が制定したものであり，2 進 7 桁で表現できる整数値のそれぞれに，英大文字や小文字，数字，記号（#や$など），一文字削除の DEL といった制御文字などを割り当てている．**表**に ASCII コードの一部を示す．例えば，英大文字「C」は「ASCII コード値 67（2 進数 “ 1000011 ”）」に，小文字「c」は「99（2 進数 “ 1100011 ”）に割り当てられている．実際，大文字と小文字の ASCII コードの差はちょうど 32 である．2 進数で考えた場合には 6 桁目（左から 2 ビット目）だけが異なるため，特定のビットだけを反転させるだけで大文字と小文字の変換が可能となるよう設計されている．

表 2·3　ASCII コードの一部

ASCII	文字	ASCII	文字	・・・	ASCII	文字	ASCII	文字
48	0	64	@	・・・	96	’	112	p
49	1	65	A	・・・	97	a	113	q
50	2	66	B	・・・	98	b	114	r
51	3	67	C	・・・	99	c	115	s
52	4	68	D	・・・	100	d	116	t
53	5	69	E	・・・	101	e	117	u
54	6	70	F	・・・	102	f	118	v
55	7	71	G	・・・	103	g	119	w
56	8	72	H	・・・	104	h	120	x
57	9	73	I	・・・	105	i	121	y
58	:	74	J	・・・	106	j	122	z
59	;	75	K	・・・	107	k	123	{
60	<	76	L	・・・	108	l	124	|
61	=	77	M	・・・	109	m	125	}
62	>	78	N	・・・	110	n	126	˜
63	?	79	O	・・・	111	o	127	DEL

■6 群 - 4 編 - 2 章

2-4 IEEE 754（2 進浮動小数点数データ形式）

（執筆者：石井康雄）［2011 年 9 月 受領］

0 と 1 でデータを扱う計算機で実数を表現するために IEEE 754 で規定される 2 進浮動小数点数データ形式が広く利用されている．この形式では符号付きゼロ，非ゼロの実数，±∞，非数（Not-a-Number：NaN）を固定長のビット列に対応付けて浮動小数点数を表現する．

2 進浮動小数点数データ形式で実数は $(-1)^{\mathrm{sign}} \times 2^{\mathrm{exponent}} \times \mathrm{significand}$ と 3 つの部分に分割される．それぞれ符号（sign），指数部（exponent），仮数部（significand）と呼ばれる．この 3 つの要素は計算機上で**図 2･1** に示すデータ形式で表現される．

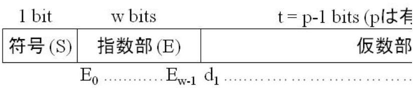

図 2･1　浮動小数点数データ形式

各フィールドのビット幅はデータ形式の精度によって異なり，その詳細を**表 2･4** に示す．

表 2･4　浮動小数点数の精度と各フィールドのビット幅

名称	**符号**	**指数部** (w)	**仮数部** (t)	**指数バイアス** (bias)
binary32(**単精度**)	1 bit	8 bit	23 bit	127
binary64(**倍精度**)	1 bit	11 bit	52 bit	1023

データ形式は**表 2･5** のルールで 1 つの実数に対応付けされる．非数は用途に応じて演算例外を検出する signaling NaN（sNaN）と例外を検出しない quiet NaN（qNaN）に区分される．

表 2･5　浮動小数点データ形式が表現する実数

条件	**区分**	**表現する実数**
$0 < E < 2^w - 1$	**正規化数**（Normal Number）	$(-1)^S \times 2^{E\text{-bias}} \times (1 + T \times 2^{-t})$
$E = 0$	$T \neq 0$ **の場合，非正規化数**	$(-1)^S \times 2^{1\text{-bias}} \times (T \times 2^{-t})$
	$T = 0$ **の場合，符号付きゼロ**	$(-1)^S \times 0$
$E = 2^w - 1$	$T = 0$ **の場合，符号付き無限大**	$(-1)^S \times \infty$
	$T \neq 0$ **の場合，非数**	$d_1 = 1$ **で** qNaN，$d_1 = 0$ **で** sNaN

実数を 2 進浮動小数点数データ形式に変換する場合には，その値と完全に一致する実数表現を持つデータ形式がない場合がある．その場合には丸め処理を行い，表現可能な数値に変換する．IEEE 754 では 5 つの丸め方式が定義され，Round to Nearest Even が標準で利用される．この丸め方式は実数を最も近い正規化数に対して丸め（入力が 2 つの正規化数の中間値だった場合には $d_p = 0$ の値に対して丸める），丸めで生じる誤差の大きさと偏りを最小化

する．

IEEE 754 では演算（加減乗除や比較など）も定義され，演算時に特殊な結果が得られた場合に出力する例外も定義されている．例外には無効演算，ゼロ除算など 5 つの例外が定義されている．0/0 などの不正な演算をした場合や sNaN を演算に利用した場合には無効演算例外が検出され，演算結果は qNaN となる．sNaN は未初期化の浮動小数点数に用いて，不正値の演算利用を検出するためなどに用いられる（sNaN は演算結果で生成されない）．

■参考文献

1) IEEE Standard for Floating-Point Arithmetic 754-2008, IEEE, 2008.