情報基礎シリーズ1

情報基礎理論

電子開発学園出版局

情報基礎シリーズ 1

情報基礎理論

電子開発学園出版局

はじめに

コンピュータが開発されて以来、コンピュータは多くの分野で利用されてきました。現在では例えば、旅券を発行する、ご飯を炊く、ゲームをする、ニュースを知る、電話をするなど私たちの生活の中に根付いた分野で利用され、なくてはならない物として、また身近な物として存在しています。また現在のネットワーク社会はコンピュータなくして成り立ちません。

コンピュータはこれらの多くの分野で実現する様々な事柄を実現するために、“情報”を取り扱い、制御し、処理しています。コンピュータで取り扱い、処理する“情報”とはいったいどのようなもので、それを取り扱い、処理するためにどのような技術が使用されているのでしょうか。本書を含む「情報基礎シリーズ」では、このような知識、技術を“インフォメーションテクノロジ（IT）”と総称しております。

この“インフォメーションテクノロジ”を学ぶことは、様々な情報とそれを取り扱うコンピュータを学ぶことであり、それにより現代社会の基盤となるコンピュータについて理解を深めることとなり、そしてその社会についても知ることになります。

本書は、多くの分野から成立している“インフォメーションテクノロジ”のうち、基礎理論として、次の内容について説明します。

- 基礎数学（離散数学、応用数学）
- 情報理論（情報に関する理論）
- 通信と制御に関する理論

これらを最初に学習することにより、コンピュータを含むインフォメーションテクノロジをいっそう理解できるようになります。

本書によって、シリーズ１は情報とはなにか、情報技術とはなにかを学び、理解を深め、情報処理技術者試験の取得等に役立てていただければ幸いです。

編著者

目　　次

情報基礎シリーズ 1

情報基礎理論

第１章　基礎数学

１．１　離散数学

離散数学とは、連続的ではないバラバラの対象を扱う数学のことであり、アルゴリズムやプログラミングなどコンピュータで扱う数値の分野として知られている。コンピュータを学ぶ前に、離散数学をまず学習することにより、コンピュータを含むインフォメーションテクノロジを理解できるようになる。

１．１．１　基数

（１）基数とは

我々が普段使用している数値は、０～９までの１０種類の組み合わせで表現している。そして、それぞれの桁は１０の累乗（下位の桁から順に０乗、１乗、２乗、３乗…）で重み付けされている。この時、重み付けの基本となる１０を**基数**と呼ぶ。

基数

例）３５６$_{(10)}$＝３×１００＋５×１０＋６×１

＝３×１０^{2}＋５×１０^{1}＋６×１０^{0}

例）０．２８４$_{(10)}$＝２×$\frac{1}{10}$ ＋ ８×$\frac{1}{100}$ ＋４×$\frac{1}{1000}$

＝２×１０$^{-1}$＋８×１０$^{-2}$＋４×１０$^{-3}$

（網掛け：基数）

特にその数値の基数を明記したい場合には、上図の３５６$_{(10)}$のように右下に括弧書きで基数を書き添える。

我々が普段使用している１０を基数にした数値の表現を１０進数と呼ぶ。

１０進数と同じように、別の数値を基数とした数値がある。例えば２を基数とした数値は２進数、１６を基数とした数値は１６進数となり、Ｎを基数とした数値は**Ｎ進数**と呼ばれる。

(a)　２進数と１６進数

２進数は基数が２であるので、０と１だけの数値の組み合わせで表現する。ただし、２進数では数が大きくなると桁数が非常に増えて見づらくなる。そのため、２進数と変換が比較的簡単な１６進数もよく利用される。１６進数は、０～９、A、B、C、D、E、Fの数字と英字の１６種類の組み合わせで表現される。

また、１６進数は１の場合は“$1_{(16)}$”と表記されるが、“０ｘ１”のように先頭に“０ｘ”をつけて表記することもよく行われる。この本では$1_{(16)}$の表記を使用する。

１０進数、２進数、１６進数を比較すると、次のようになる。

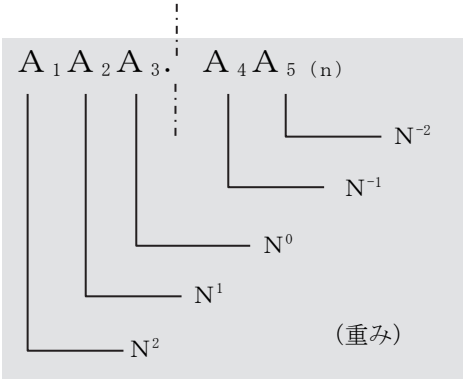
１０進数・２進数・１６進数の比

１０進数	２進数	１６進数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14

(b)　Ｎ進数

Ｎ進数はＮを基数とした数値である。０～（Ｎ－１）の整数の組み合わせで表現されて、Ｎごとに桁上がりし、各桁ではＮが重みとなる。

Ｎ進数



各桁の数
- ０～（Ｎ－１）の整数
- Ｎごとに桁上がり

$A_1A_2A_3.\ A_4A_5{}_{(n)}$

$=A_1\times N^2+A_2\times N^1+A_3\times N^0+A_4\times N^{-1}+A_5\times N^{-2}$

(c)　Ｎ進数の加減算

Ｎ進数の加減算は、１０進数の計算と同様である。ただし、加算の場合はＮで桁上がりをし、減算の場合は上の桁から借りた場合にＮを足すことが異なる。

１０進数と２進数の加減算（１）

例）１０進数の加算の場合

0	2	3	5
＋2	＋3	＋5	＋8
2	5	8	１３
			↩(桁上がり)

例）２進数の加算の場合

0	0	1	1
＋0	＋1	＋0	＋1
0	1	1	１０
			↩(桁上がり)

１０進数と２進数の加減算（２）

例）２進数で複数桁の加算の場合

```
  １１                      １１１
     （桁上がり）                （桁上がり）
  １０００１               １１１０１０
＋０１０１１              ＋００１００１
――――――――              ――――――――――
  １１１００              １０００１１
```

例）２進数での減算の場合

```
       （１桁上から借り）
   １１００１
－     １０１
――――――――――
   １０１００
```

例題

【問題】

次の計算は何進数で成立するか。

　１３１－４５＝５３

【解答】

式を変形すると、「１３１＝５３＋４５」となる。これの、１の桁だけに注目すると、「１＝３＋５」である。これは桁上がりが発生していることがわかるので、「１１＝３＋５」となる。

１１は１０進数ならば「１０＋１」だが、N進数の場合は「N＋１」である。よって、「N＋１＝３＋５」となり、「N＝７」であり７進数であることがわかる。

（２）基数変換

ある基数の数から、別の基数の数に変換することを**基数変換**と呼ぶ。例えば、１０進数を２進数に変換する、またはその逆の２進数から１０進数に変換することが基数変換である。

インフォメーションテクノロジの世界では、特に１０進数、２進数、１６進数をよく使用するため、これらの基数の間での基数変換がよく行われる。

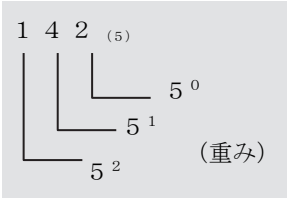
(a)　Ｎ進数と１０進数の間での基数変換

（ア）Ｎ進数から１０進数へ

Ｎ進数から１０進数へ基数変換するには、Ｎ進数の各桁に重みを掛け合わせ、各桁から求めた値の総和が１０進数になる。

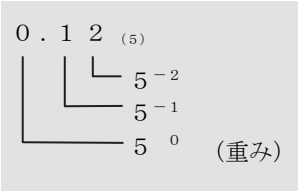
１０進数との基数変換

【整数の場合】例）１４２$_{(5)}$を１０進数に変換する



$$142_{(5)} = 1\times5^{2}+4\times5^{1}+2\times5^{0}$$
$$= 25+20+2$$
$$= 47_{(10)}$$

【小数の場合】例）０．１２$_{(5)}$を１０進数に変換する



$$0.12_{(5)} = 1\times5^{-1}+2\times5^{-2}$$
$$= \frac{1}{5}+\frac{2}{25}$$
$$= 0.28_{(10)}$$

（イ）１０進数からＮ進数へ

【整数の場合】

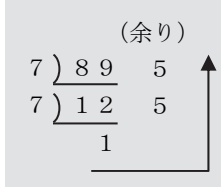
１）与えられた１０進数（Ｘとする）をＮで割り、商と余りを求める。

２）１）で求めた商を新しいＸとみなし、さらにＮで割り、商と余りを求める。これを商が１になるまで繰り返す。

３）商と余りを求めた逆の順番で並べる。

Ｎ進数との基数変換（１）

例）８９$_{(10)}$を７進数に変換する



商と余りを求めた逆の順番に並べると

$$89_{(10)} = 155_{(7)}$$

となる

【値が小数の場合】

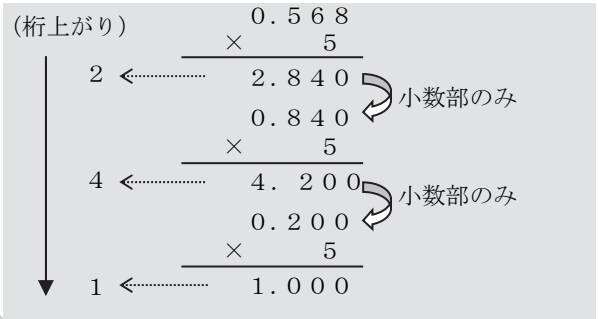
１）１０進数の小数部のみを抜き出した値（Ｘとする）をＮ倍して積を求める。

２）求めた積の小数部のみを抜き出して、新たなＸとし、Ｎ倍して積を求める。これを小数部が０になるまで繰り返す。

３）積の整数部分を並べる。

Ｎ進数との基数変換（２）

例）０．５６８$_{(10)}$を５進数に変換する



整数部への桁上がりを並べると

０．５６８$_{(10)}$＝０．２４１$_{(5)}$

となる

ただし、１０進数の小数は、Ｎ進数に変換すると割り切れる数値にならない場合がある。例えば、０．１$_{(10)}$を７進数に変換すると、０．０４６２０４６２…と“０４６２”が無限に続くようになる。このように同じ数が繰り返し出てくる小数を**循環小数**と呼び、$\dot{0}.\dot{0}\dot{4}\dot{6}\dot{2}_{(7)}$と表現する。

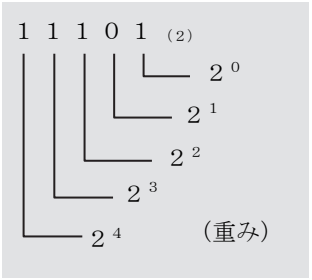
(b)　２進数と１０進数の間での基数変換

インフォメーションテクノロジで最も頻繁に使用するのは２進数と１０進数との間での基数変換である。

（ア）２進数から１０進数へ

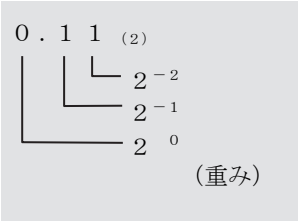
１０進数との基数変換

【整数の場合】例）１１１０１$_{(2)}$を１０進数に変換する



$$11101_{(2)}$$
$$=1\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$$
$$=16+8+4+1$$
$$=29_{(10)}$$

【小数の場合】例）０．１１$_{(2)}$を１０進数に変換する



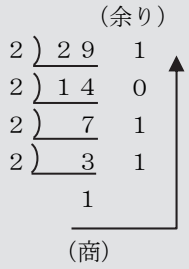
$$0.11_{(2)}$$
$$=1\times2^{-1}+1\times2^{-2}$$
$$=\frac{1}{2}+\frac{1}{4}$$
$$=0.75_{(10)}$$

（イ）１０進数から２進数へ

【値が整数の場合】

２進数との基数変換（１）

例）２９$_{(10)}$を２進数に変換する



商と余りを求めた逆の順番に並べると

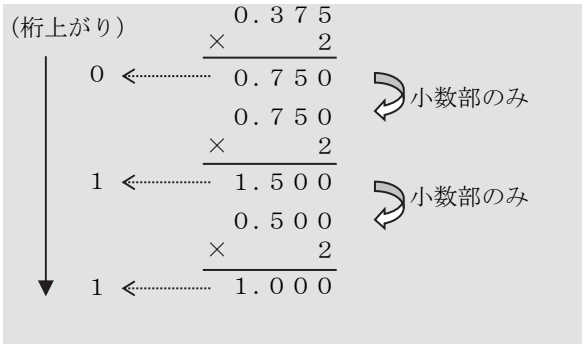
２９$_{(10)}$＝１１１０１$_{(2)}$

となる

【値が小数の場合】

２進数との基数変換（２）

例）０．３７５$_{(10)}$を２進数に変換する



整数部への桁上がりを並べると

０．３７５$_{(10)}$＝０．０１１$_{(2)}$

となる

(c)　２進数と１６進数の間での基数変換

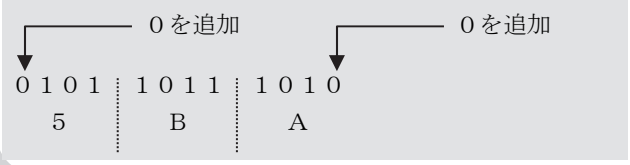
２進数と同様に１６進数もよく使用される。１６進数の場合、２進数との基数変換が簡単に行える。そのため、１０進数と１６進数の間での基数変換を行う場合、間に２進数をはさみ、１０進数から２進数に変換したうえで１６進数に変換する、もしくは１６進数から２進数に変換したうえで１０進数に変換する、という変換をよく行う。

（ア）２進数から１６進数へ

２進数から１６進数へ変換するときは、小数点を境にして４桁ずつ区切ってブロックとし、ブロックごとを１桁の１６進数に置き換える。

１６進数との基数変換（１）

例）１０１１０１１．１０１$_{(2)}$を１６進数に変換する



１０１１０１１．１０１$_{(2)}$

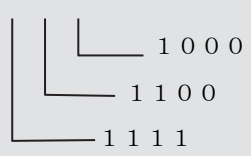
＝５Ｂ．Ａ$_{(16)}$

（イ）１６進数から２進数へ

１６進数から２進数へ変換する場合は、１６進数の各桁を２進数４桁に変換し、先頭から並べる。

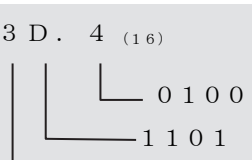
１６進数との基数変換（２）

【整数の場合】例）ＦＣ８$_{(16)}$を２進数に変換する



ＦＣ８$_{(16)}$＝１１１１１１００１０００$_{(2)}$

【小数の場合】例）３Ｄ．４$_{(16)}$を２進数に変換する。



３Ｄ．４$_{(16)}$＝００１１１１０１．０１００$_{(2)}$

削除

＝１１１１０１．０１$_{(2)}$

１．１．２　数値の表現

（１）コンピュータでの情報の表現

コンピュータでは数値、文字、画像などのさまざまな情報を取り扱う。まずコンピュータが取り扱う情報の表現方法を説明する。

(a)　情報の表現

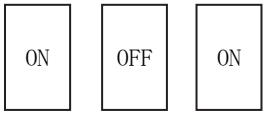
（ア）ビット

コンピュータにおける記憶は、“１”と“０”の組み合わせで成り立っている。この“１”と“０”が情報を表現する最小の単位で、**ビット**（bit：binarydigit）と呼ぶ。ビットは“１”と“０”のみしか扱わないので、多くの場合これを２進数の数値として、またはそれを変換して１６進数の数値として表現する。

ビットを複数並べることによって、多くの情報を表現することができる。１ビットでは“１”と“０”の２つの情報だけしか表現できないが、２ビットあれば、“１１”、“１０”、“０１”、“００”の４つの情報を表現できる。

ビットと情報

コンピュータの記憶はスイッチが
複数並んでいると考える



１　　０　　１

ＯＮを１、ＯＦＦを０とすると３つのスイッチで１０１という情報を保持している

このスイッチが数字を記憶しているとすれば１０１$_{(2)}$＝５$_{(10)}$

この３つのスイッチで５$_{(10)}$を記憶していることになる

つまり、ｎビットあれば、2^nのパターンを表現することができることになる。

（イ）バイト

現在のコンピュータでは、人間の使用する文字や数字などの情報を表現するときは多くの場合、８ビットを１つの単位として用いる。これを**バイト**（byte）と呼び、１バイトは2^8＝２５６通りの情報を表現できる。

（ウ）ワード

コンピュータで一度に取り扱うことのできる情報の大きさを、**ワード**（word：語）と呼ぶ。ワードの大きさはコンピュータの性能によって異なる。例えば、一度に取り扱えるデータの量が１６ビットのコンピュータであれば、そのコンピュータの１ワードは１６ビットで、一度に取り扱えるデータの量が３２ビットのコンピュータであれば、そのコンピュータの１ワードは３２ビットということになる。

(b)　大きさの単位

コンピュータで扱う数値は非常に大きい、または非常に小さい値になることがある。記憶装置に記憶するデータ量では大きな値が使われ、一方でコンピュータの動作時間を表すには小さい値が使われることが多い。そのため、単位に接頭辞と呼ばれる大きさを示すアルファベットをつけることで、表現する桁数を減らすことができる。

例えば、１０Ｋ（キロ）では、１０，０００となり、１万を表す。

接頭辞

接頭辞	記号	大きさ
エクサ	E	10^{18}＝１００京
ペタ	P	10^{15}＝１０００兆
テラ	T	10^{12}＝１兆
ギガ	G	10^{9}＝１０億
メガ	M	10^{6}＝１００万
キロ	K	10^{3}＝１０００
なし		10^{0}＝１

接頭辞	記号	大きさ
ミリ	m	10^{-3}＝１０００分の一
マイクロ	μ	10^{-6}＝１００万分の一
ナノ	n	10^{-9}＝１０億分の一
ピコ	p	10^{-12}＝１兆分の一

（２）コンピュータでの数値の表現

コンピュータではビットで情報が保持される。コンピュータで数字を取り扱う場合には、ビットが“1”と“0”のみを使用するため、２進数での表現方法がコンピュータでは使用される。

一方で、２進数では人間が使用するにはなじみが薄くわかりにくいため、人間にとってわかりやすい１０進数をビットで表現する方法も使用されている。

(a)　１０進数の表現

（ア）ＢＣＤコード

数字を表現する時に１０進数で表現される方法としては、**２進化１０進コード（Binary Coded Decimal：ＢＣＤコード）**が使われる。

ＢＣＤコードは１０進数の各桁をそれぞれ４ビットの２進数で表現する方式である。

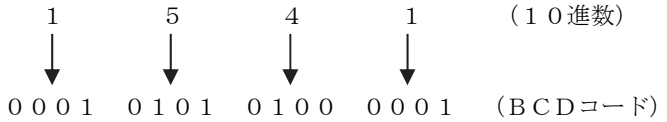
ＢＣＤコード

例）１５４１$_{(10)}$を表現する

【２進数で表現した場合】

１５４１$_{(10)}$　＝　１１０００００１０１$_{(2)}$

【ＢＣＤコードを使った場合】



実際にはこのＢＣＤコードを使用して、**ゾーン１０進数**、**パック１０進数**と呼ばれる表現方法が使用される。

（イ）ゾーン１０進数

ゾーン１０進数は１０進数１桁を１バイト（８ビット）で表現する形式である。各桁を表現する１バイトのうち下位４ビットはＢＣＤコードで表現した数値を入れる。上位４ビットにはゾーンビットと呼ばれる数値を入れる。ただし、桁のうち最下位の桁の上位４ビットは符号を表現する符号ビットをいれる。

ゾーンビットに使用する値は、コンピュータで使用される文字コードによって異なる。文字コードについては第２章で説明する。

ゾーン１０進数

ここではゾーンビットは「１１１１ (2)」、符号ビットは正ならば「１１００ (2)」、負ならば「１１０１ (2)」を入れるものとする。

例）－１９８６ (10) を表現する

	1		9		8		6
ゾーンビット	BCDコード	ゾーンビット	BCDコード	ゾーンビット	BCDコード	符号ビット	BCDコード
1111	0001	1111	1001	1111	1000	1101	0110
F	1	F	9	F	8	D	6

よって、－１９８６ (10) はゾーン１０進数ではＦ１Ｆ９Ｆ８Ｄ６ (16) と４バイトで表現される。

（ウ）パック１０進数

パック１０進数は、１０進数１桁を４ビットで表現する方式である。つまり、１バイトで２桁を表現できる。各桁をゾーン１０進数と同様にＢＣＤコードで表現するが、最下位桁の後ろに符号ビットをつけて符号を表す。ただし、ビット数が８の倍数にならない場合（バイトで表現できない場合）は、０００００ (2) を先頭に挿入することで、８の倍数にする。

パック１０進数はゾーン１０進数に比べてゾーンビットを省略して詰めており、少ないバイト数で多くの桁を表現できる。この省略して詰めていることを**パック**されていると呼ぶ。つまり、ゾーン１０進数をパックしたものがパック１０進数であると言える。また逆にゾーン１０進数はパック１０進数に対してパックされていない(**アンパック**されている)とも言える。そのため、ゾーン１０進数のことをアンパック１０進数とも呼ぶ。

パック１０進数

ここでは符号ビットは正ならば「１１００$_{(2)}$」、負ならば「１１０１$_{(2)}$」を入れるものとする。

例１）１５４１$_{(10)}$を表現する

１　　５　　４　　１

BCDコード　BCDコード　BCDコード　BCDコード　符号ビット

0000	0001	0101	0100	0001	1100
0	1	5	4	1	C

よって、１５４１$_{(10)}$はパック１０進数では０１５４１Ｃ$_{(16)}$と３バイトで表現される。

例２）－１９８$_{(10)}$を表現する

１　　９　　８

BCDコード　BCDコード　BCDコード　符号ビット

0001	1001	1000	1101

よって、－１９８$_{(10)}$はパック１０進数では１９８Ｄ$_{(16)}$と２バイトで表現される。

(b)　２進数での数値の表現

（ア）正の数の表現

ビットではｎビットあれば、2^n通りの表現ができる。例えば８ビットならば００００００００～１１１１１１１１の表現が可能になる。このビットの並びを、２進数として考えれば８ビットでは正の整数として０$_{(10)}$～２５５$_{(10)}$の数値が表現できることになる。

つまり、ｎビットある場合、正の整数は０～2^n-1までの値を表現できることになる。

（イ）負の数の表現（符号付き絶対値方式）

ｎビットで負の数を表現する方法には２通りある。１つは、符号をつける方式で、**符号付き絶対値方式**と呼ぶ。

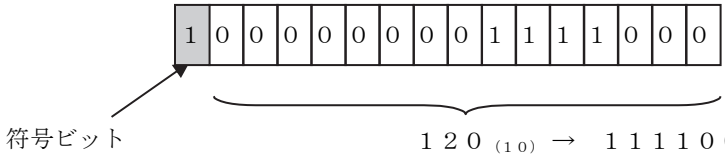
符号付き絶対値方式では一番先頭のビットを符号として扱う。先頭のビットを符号ビットと呼び、正の数ならば０を、負の数ならば１をこのビットに入れ、残りのビットで数値を表す。

符号付き絶対値方式で表すことができる数の範囲は、１ビットを符号ビットとして扱うため、ｎビットの場合－（$2^{n-1}-1$）～$2^{n-1}-1$までとなる。

符号付き絶対値方式

例）１６ビットで、－１２０$_{(10)}$を表現した場合

－１２０$_{(10)}$＝１０００００００１１１１０００$_{(2)}$



符号ビット

（負の数を表す１を入れる）

１２０$_{(10)}$ →　１１１１０００$_{(2)}$

（ウ）負の数の表現（補数を用いる方式）

【補数の定義と２進数の補数表現】

補数とは、とある数に対して加算した場合桁上がりが起きる最小の数のことである。この補数を使うことによって符号を使わず負の数を表現することができるようになる。

補数を使って負の数を表す場合、ある整数の補数がその数の負の数となる。例えば、＋４$_{(10)}$という数字がある場合、この＋４の補数Ｘが－４を意味する数値となる。

補数には“基数の補数”と“基数－１の補数”がある。ここでは２進数の“**２の補数**”と“**１の補数**”の２つ補数の求め方と、それを使った計算を説明する。

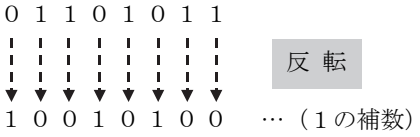
・１の補数（基数－１の補数）

基数－１の補数は､その桁の数の最大値から減算することによって求めることができる。

２進数では基数－１の補数は、１の補数である。１の補数は減算を使わなくても、各桁のビットを反転（１を０に、０を１に）することで求めることができる。

１の補数

例）０１１０１０１１$_{(2)}$の１の補数を求める

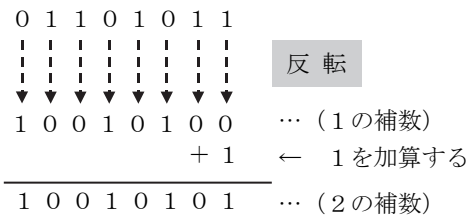


・２の補数（基数の補数）

基数の補数は、基数－１の補数に１を加算することで求めることができる。

２の補数

例）０１１０１０１１$_{(2)}$の２の補数を求める



【補数による表現範囲】

２進数で補数を使用すると、負の数は先頭のビットが１になるため、補数を使用した場合の先頭のビットは事実上の符号ビットと同じ扱いになる。

また、１の補数（基数－１の補数）では定義上、０の表現方法が２通り（＋０と－０）でき、演算が複雑になるため、通常は２の補数（基数の補数）を使って負の数を表現する。

従って、８ビットで２の補数を使用して表現できる数値の範囲は、１０進数での－１２８（-2^7）～＋１２７（2^7-1）までとなる。ｎビットある場合の数値の範囲は、次のようになる。

２の補数の範囲

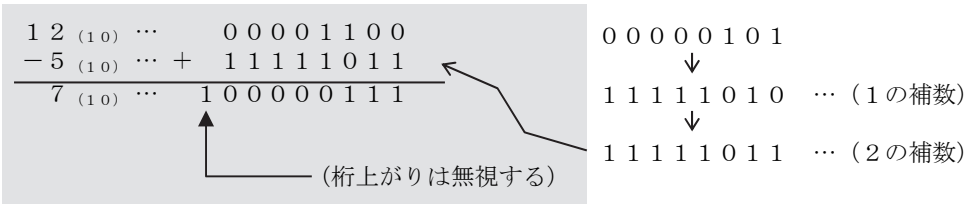
$$-2^{n-1} \sim 2^{n-1}-1$$

【２の補数を使った計算】

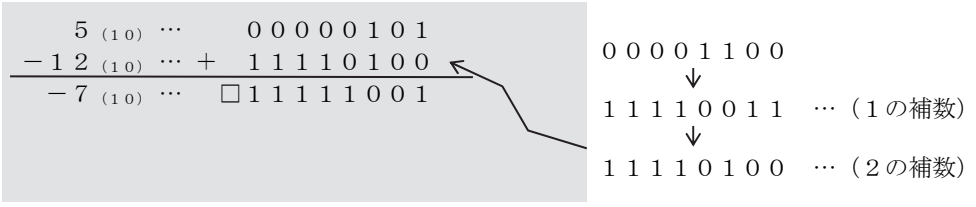
補数を使うことで減算を加算で行うことができるようになる。例えば、ａ－ｂという減算を行う場合、ｂの補数を取ることで、ａ+（－ｂ）として加算の形になる。加算で減算が行えるということは、コンピュータが演算を行う機能として加算機能だけあればよく、その分単純な仕組みでよいということになる。

２の補数を使った計算

例１）　００００１１００ $_{(2)}$ －００００００１０１ $_{(2)}$ を求める



例２）　００００１０１ $_{(2)}$ －００００１１００ $_{(2)}$ を求める



(c)　小数の表現

ビットでは“０”か“１”の表現しかできないため、“小数点”を表現することができない。よって、ビットで小数を表現するためには、小数点の位置などを考慮した表現が必要となる。少数の表現には“固定小数点数”による表現と、“浮動小数点数”による表現がある。

（ア）固定小数点数

固定小数点数は、小数点の位置が固定的に決められている方式である。使用するビットの長さは１ワードで１６ビット、３２ビット、６４ビットのいずれかとなる。また、負の数は２の補数を使用して表すことが一般的である。

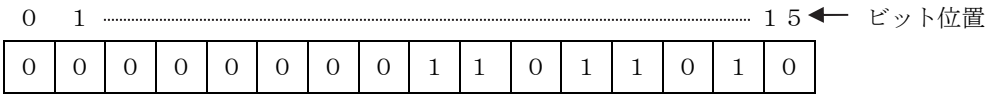
小数点の位置を固定的に決定するが、その位置はコンピュータによって異なる。小数点を最下位桁の次や、最上位桁とその次の桁の間に置く場合が一般的である。

固定小数点数（１）

例）２１８$_{(10)}$を１６ビット固定小数点数で表す

２１８$_{(10)}$＝１１０１１０１０$_{(2)}$

となる。従って、次のように表現できる。



▲ 小数点の位置

小数点の最下位桁の次に小数点を配置した場合は“固定小数点数”とは呼ぶものの、（ｂ）の（ウ）で説明した２の補数を用いた整数と同じになる。つまり、整数を表現するということは、小数点の最下位桁の次に小数点を配置した固定小数点数による表現である、とも言える。

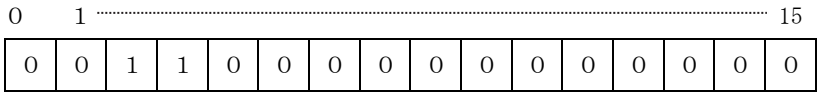
また、小数点を最上位桁とその次の桁の間に置く場合は、次のようになる。

固定小数点数（２）

例）０．３７５$_{(10)}$を１６ビット固定小数点数で表す

０．３７５$_{(10)}$＝０．０１１$_{(2)}$

となる。従って、次のように表現できる。



▲ 小数点の位置

（イ）浮動小数点数

固定小数点数で表すことのできる範囲はビット数により決まってしまい、使用できるビット数はそのコンピュータで使用するワードによって決まる。そのため、科学技術計算で使用するような非常に大きな数値や、非常に小さな数値を扱うことがワードのビット数の制限によりできなくなる。そこで、小数点を固定しない**浮動小数点数**と呼ばれる方式を使用する。

浮動小数点数では数値を“仮数”と“指数”に分けて表現することで、より広い範囲の数値を取り扱うことができるようになる。

浮動小数点数（１）

浮動小数点数の表現では、数値を次のように表す。

$$X_{(n)} = A \times n^{B}$$

例）５３８．２５$_{(10)}$を浮動小数点数で表現する

$$538.25_{(10)} = 538.25 \times 10^{0}$$
$$= 53.825 \times 10^{1}$$
$$= 5.3825 \times 10^{2}$$
$$= 0.53825 \times 10^{3}$$

それぞれ、０．５３８２５ → 仮数、１０ → 基数、３ → 指数と呼ぶ。

０．５３８２５の部分を仮数部、10^{3}の部分を指数部と呼ぶ。

コンピュータでは、仮数を１０進数ではなく１６進数または２進数で表現し、基数は２または１６を使うことが多い。例えば、１６進数の場合の表現は次のようになる。

浮動小数点数（２）

例）２１Ａ．４を１６進数の浮動小数点数で表す

$$21A.4_{(16)} = 21A.4 \times 16^{0}$$
$$= 21.A4 \times 16^{1}$$
$$= 2.1A4 \times 16^{2}$$
$$= 0.21A4 \times 16^{3}$$

上図のように、浮動小数点数では小数点が左に１つ移動すると指数が＋１され、逆に小数点が右に移動すると指数が－１される。

また、浮動小数点数をビットで表記するには、エクセス６４形式と、ＩＥＥＥ７５４形式がある。なお、ＩＥＥＥ（電気電子学会：the Institute of Electrical and Electronics Engineers）は通信、電気、電子に関する学会で、様々な規格の標準化を行っている。

この形式で浮動小数点数を表記するには、まず仮数の**正規化**を行う。エクセス６４形式では１６進数で０．ｘｘｘのように１以下の値で、小数点第１位に０以外の数値があるようにする。一方、ＩＥＥＥ７５４形式では、２進数に直した後、１．ｘｘｘのように１以上２未満の、先頭に１がある値にする。

正規化

例）２６．２５$_{(10)}$を浮動小数点数にするため正規化する

【エクセス６４】

まず、１６進数に直す。２６．２５$_{(10)}$＝１Ａ．４$_{(16)}$

１Ａ．４　＝１Ａ．４ × １６^{0}　＝０．１Ａ４ ×１６^{2}

【ＩＥＥＥ７５４】

まず、２進数に直す。２６．２５$_{(10)}$＝１１０１０．０１$_{(2)}$

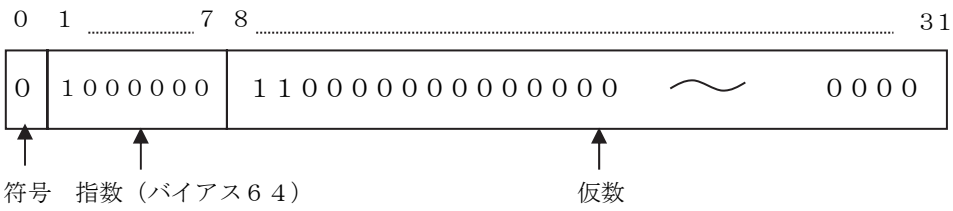
１１０１０．０１ × ２^{0}　＝１．１０１００１×２^{4}

浮動小数点数では仮数については正規化を行う。一方、指数は負の数を表現するためバイアスを使用する。エクセス６４、ＩＥＥＥ７５４どちらとも指数は２の補数表示を使わず、特定の値を足して表記する。この特定の値をバイアスと呼ぶ。例えば、エクセス６４では指数に６４を足して表記する。つまり指数が１だった場合は６５、－１だった場合は６３として表記する。

【エクセス６４】

エクセス６４では基数として１６進数を使い、一般的に使用されるエクセス６４は３２ビットで符号ビット１ビット、指数７ビット、仮数２４ビットとなっている。エクセス６４の“６４”は指数に使用するバイアスの値を示す。

エクセス 64 形式 浮動小数点



・符号　：　１ビット。正の値のときは０、負の値のときは１である。

・指数　：　７ビット。基数として１６を使用した場合の指数。バイアスとして６４を足した値となる。

・仮数　：　２４ビット。正規化した仮数のうち、０．ｘｘｘｘ$_{(16)}$のｘｘｘｘの部分。

例１）上図の浮動小数点数は

符号…０のため正

指数…１０００００００$_{(2)}$＝６４$_{(10)}$。バイアスとして６４を足しているため０となる

仮数…１１００～$_{(2)}$＝Ｃ$_{(16)}$。よって、０．Ｃ$_{(16)}$

よって、０．Ｃ$_{(16)}$　×　１６^{0}　＝　０．Ｃ$_{(16)}$　＝<u>０．７５</u>$_{(10)}$

例２）－１２８．５$_{(10)}$を浮動小数点数で表す

－１２８．５$_{(10)}$＝－１００００００００．１$_{(2)}$＝－８０．８$_{(16)}$

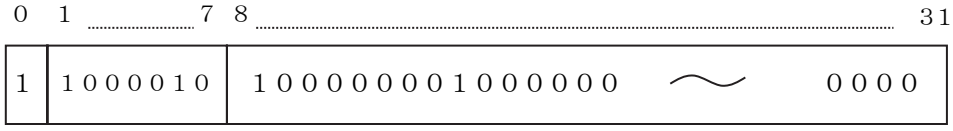
＝－０．８０８$_{(16)}$　×　１６^{2}

符号…負のため１

指数…２にバイアスの６４を足して６６。よって、１０００００１０$_{(2)}$

仮数…０．８０８$_{(16)}$の８０８$_{(16)}$＝１０００００００１０００$_{(2)}$

よって、下図のような表記になる。

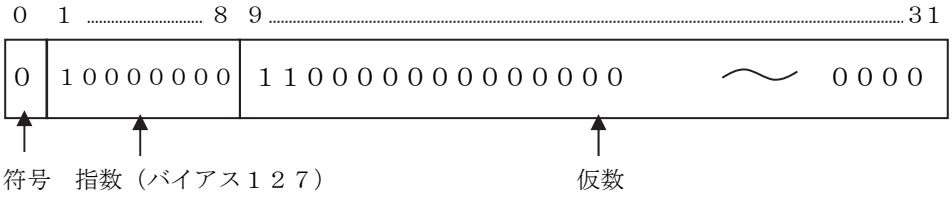


【ＩＥＥＥ７５４】

ＩＥＥＥ７５４での浮動小数点数の形式は、基数として２進数を使う。ビット数は、３２ビット、６４ビット、１２８ビットがある。３２ビットの浮動小数点を**単精度浮動小数点数**、６４ビットを**倍精度浮動小数点数**、１２８ビットを四倍精度浮動小数点数とも呼ぶ。単精度、倍精度、四倍精度では表示できる値の範囲が大幅に異なり、また誤差の範囲も変わる。

ここでは単精度浮動小数点数と、倍精度浮動小数点数を説明する。

ＩＥＥＥ７５４形式 単精度浮動小数点数



・符号　：　１ビット。正の値のときは０、負の値のときは１である。

・指数　：　８ビット。基数として２を使用した場合の指数。バイアスとして１２７を足した値となる。

・仮数　：　２３ビット。正規化した仮数のうち、１．ｘｘｘｘ$_{(2)}$のｘｘｘｘの部分となる。

例）－１３０．５$_{(10)}$を浮動小数点数で表す

－１３０．５$_{(10)}$＝－１０００００１０．１$_{(2)}$

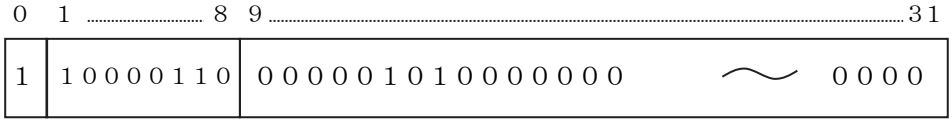
＝－１．０００００１０１$_{(2)}$ × 2^{7}

符号…負のため１

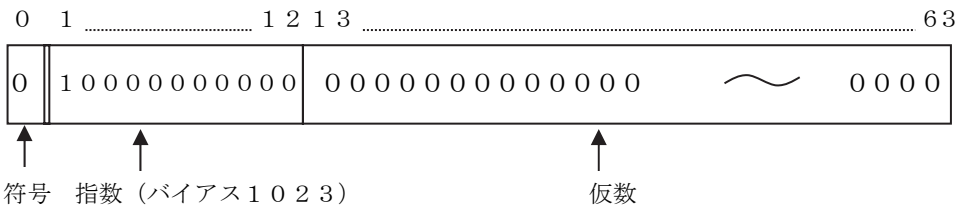
指数…７にバイアスの１２７を足して１３４。よって、１０００００１１０$_{(2)}$

仮数…１．０００００１０１$_{(2)}$の小数点以下の部分で０００００１０１$_{(2)}$

よって、下図のような表記になる。



ＩＥＥＥ７５４形式 倍精度浮動小数点数



・符号　：　１ビット。正の値のときは０、負の値のときは１である。

・指数　：　１１ビット。基数として２を使用した場合の指数。バイアスとして１０２３を足した値となる。

・仮数　：　５２ビット。正規化した仮数のうち、１．ｘｘｘｘ$_{(2)}$のｘｘｘｘの部分となる。

ただし、小数点をコンピュータで扱う場合には、実際の値と表現された値との間に差異が生じる場合がある。この差異のことを**誤差**と呼ぶ。誤差については、「１．１．３」で説明する。