7　符号化文字集合と文字コード

文字コードは 16 進数で表されることが多いので文字コードに先立って 2 進数、16 進数について述べる。

7.1　2 進数・8 進数・10 進数・16 進数

コンピュータ内部では、IC チップの足 (ゲジゲジ？) にかかる電圧が 0 V か 5 V かで On と Off とを表している。最終的に CPU が理解できるのはこの電圧の高低である。従ってコンピュータの世界では、0 V イコール 0, 5 V イコール 1 として 2 進数が頻繁に使われる。コンピュータの扱える最小の単位が IC チップの足一つにかかる電圧である。これをビット bit という。1 ビットの情報量は一桁の 2 進数に対応する。2 進数はすぐに大きくなってしまうので、8 ビットをまとめて 1 バイト byte を使うこともある。1 バイトは 8 進数で表される情報の単位である。さらに 16 ビットをまとめて 16 進数で表すこともよく行われている。コンピュータが一度に処理できるビットの大きさを使ってコンピュータの性能を表すことがある。東京女子大学の現行の iMac は、64 bits のプロセッサ (中央演算処理装置) が使われている。

10 進数と 2 進数、8 進数、16 進数の相互変換に慣れる必要がある。例えば、10 進数の 43 は、10 の 1 乗が 4 つあり、10 の 0 乗 (すなわち 1) が 3 つあるという意味である。同様にして 43 は $2^5 = 32$ が 1 つと $2^3 = 8$ が 1 つ、$2^1 = 2$ が 1 つ、さらに $2^0 = 1$ が 1 つという意味である。以下のとおり、

$$\begin{aligned} 43 &= 4 \times 10^1 + 3 \times 10^0 && (1)\\ &= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 && (2)\\ &\quad +0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 && (3)\\ &= 5 \times 8^1 + 3 \times 8^0 && (4)\\ &= 2 \times 16^1 + 11 \times 16^0 && (5) \end{aligned}$$

すなわち 10 数の 43 は、2 進数 (binary number) と 16 進数 (hexadecimal number) では $2^0, 2^1, \ldots$ および $16^0, 16^1, \ldots$ の係数を指数の大きい順に並べて表す。16 進数では係数に 10, 11, 12, 13, 14,15 が現れるので、これらを順に A, B, C, D, E, F(または a, b, c, d, e, f) で表す。何進数かを明示するとき、この講義では下つきの添字で表すことにする。したがって、10 進数 43_{10} を 2 進数と 8 進数と 16 進数で表すと、

$$43_{10} = 101011_2 = 53_8 = 2\mathrm{B}_{16} \tag{6}$$

となる[14]。

10 進数、2 進数、8 進数、16 進数の対応表を次に示す。

[14] b を自然数 ($b \geqq 2$) とする。自然数 m を

$$m = c_0 b^n + c_1 b^{n-1} + \ldots + c_n, \quad c_0 \neq 0, \quad 0 \leqq c_j < b \ (j = 1, \ldots, n) \tag{7}$$

と表わしたとき、

$$c_0 c_1 \ldots c_n{}_b \tag{8}$$

を m の b 進表現という。b を基数という。基数 b が文脈から明らかなときは添え字 b は省略する。我々が日常用いている数は $b = 10$ の 10 進数であるが、計算機科学では、$b = 2,\ 8,\ 16$ がよく用いられる。$b = 8$ のときは 8 進数 (octal number) といい、0 から 7 までの文字を用いて表わす。

10 進数	2 進数	8 進数	16 進数
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

2 進数を 4 桁ごとに下の桁から上の表に基づいて 16 進に直すと、16 進数が得られる。逆に、16 進数を 2 進数に変換するときは、各桁を上の表に基づいて 2 進に直す。

2 進数を 3 桁ごとに下の桁から区切って表すと 8 進数になる。逆に 8 進数を 2 進数に変換するときには各桁を 3 桁の 2 進数に変換すれば良い。

7.2　2 進数と 16 進数

2 進数は桁数が大きくなるので (小さい方の位から) 4 桁づつ 1 桁の 16 進数に置き換えることがしばしば行われる。

$1011001_2 = 59_{16}$

$1011001_2 =$ の読み方は「(2 進数) イチゼロイチイチゼロゼロイチ」である。59_{16} の読み方は「(16 進数) ゴキュウ」である。59_{16} は $0x59$ とも書く。読み方は「ゼロエックスゴキュウ」である。

逆に、16 進数を 2 進数に直すには 16 進数 1 桁づつ 4 桁の 2 進数に置き直せばよい。

$A1B2_{16} = 1010000110110010_2$

7.3　10 進数と 16 進数

1 桁に 16 種類の文字 (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) を用い、16 倍毎に桁を繰り上げる。

例 7.1　16 進数 $A1B2_{16}$(=0xA1B2) を 10 進数に変換する。

$\mathrm{A1B2}_{16} = \mathrm{A}_{16}\times16^3+1_{16}\times16^3+\mathrm{B}_{16}\times16^1+2_{16}\times16^0 = 10\times16^3+1\times16^3+11\times16^3+2\times16^0$
$= 10\times4096+1\times256+11\times16+2\times1 = 41394_{10}$

逆に 10 進数 41394_{10} が与えられたとき、41394 を 16 で割り、再び商を 16 で割る。この操作を商が 0 になるまで繰り返す。余りの列を最後の余りが最上位桁になるように並べたものが 16 進表記である。

16	41394	$2=2_{16}$
16	2587	$11=\mathrm{B}_{16}$
16	161	$1=1_{16}$
16	10	$10=\mathrm{A}_{16}$
	0	

$41394_{10} = \mathrm{A1B2}_{16}$

$\mathrm{A1B2}_{16}$ は 0xA1B2 とも書き、「ゼロエックスエーイチビーニ」と読む。

2 進数を固定桁数で表すとき、固定桁数が表示する数の桁数より大きい場合は、先頭を 0 で埋める。例えば 10 進数の 5 を 8 桁の 2 進数で表すと 00000101 となる。固定桁数が表示する数の桁数より小さい場合は、固定桁数を超えた部分は無視する。例えば 10 進数の 261 を 2 進数で表すと 1 0000 0101 となるので、8 桁の 2 進数で表すと (10 進数の 5 と同じ) 0000 0101 となる。

2 進数を 4 桁 (3 桁) ごとに下の桁から上の表に基づいて 16 進 (8 進) に直すと、16 進数 (8 進数) が得られる。逆に、16 進数 (8 進数) を 2 進数に変換するときは、各桁を上の表に基づいて 2 進に直す。8 進数と 16 進数の変換は、2 進数を仲立ちにして行う。

7.3.0.1　10 進数を 16 進数へ　10 進数を 16 進数に変換するには、商を次々に 16 で割り商が 0 になったとき余りを下の桁から並べる。$2010_{(10)}$ を例に取ると

16	2010	$10=\mathrm{A}_{(16)}$
16	125	$13=\mathrm{D}_{(16)}$
16	7	7
	0	

だから、

$$2010_{(10)} = 7\mathrm{DA}_{(16)}$$

となる。これは

$$2010 = 16\times125+10 = 16\times(16\times7+13)+10 \quad (9)$$
$$= 7\times16^2+13\times16+10 \quad (10)$$

より分かる。

7.3.0.2　10 進数を 2 進数へ　10 進数を 2 進数に変換するには、商を次々に 2 で割り、商が 0 になったとき、余りを下の桁から並べる。$2010_{(10)}$ を例に取ると

2	2010	0
2	1005	1
2	502	0
2	251	1
2	125	1
2	62	0
2	31	1
2	15	1
2	7	1
2	3	1
2	1	1
	0	

より、余りを下から並べて

$$2010_{(10)} = 111\ 1101\ 1010_{(2)}$$

となる。分かりやすいように下から 4 桁毎に空白を入れた。

10 進数を 16 進数に変換し、次いで 16 進数を 4 桁の 2 進数に変換しても得られる。16 進数の最高位以外の桁が $7_{(16)}$ 以下のときは、4 桁の 2 進数になるよう先頭を 0 で埋める必要がある。たとえば、10 進整数の 2010 は

$$2010_{(10)} = 7\mathrm{DA}_{(16)} = 111\ 1101\ 1010_{(2)}$$

となる。

10 進数を 8 進数に変換するのも同様である。

演習 7.1 [第 2 種情報処理技術者試験 2000 年度春期午前]

すべて同じ値を表している 2 進数、8 進数、10 進数、16 進数の組合わせはどれか。

	2 進数	8 進数	10 進数	16 進数
ア	111	10	8	8
イ	1010	12	10	A
ウ	1100100	256	100	84
エ	1111111	377	256	FF

注 7.1 第 2 種情報処理技術者試験は情報処理技術者試験 (基本情報技術者) の前身である。

7.4 補数

2 進数を使ってマイナスの数を表す場合に補数が使われる。

1 桁の N 進整数 c と 10 進整数 $d > 0$ に対し、c_d は c が d 個続いた数とする。たとえば、2 進整数として $1_5 0_3 = 11111000$ である。

d 桁の N 進数 x に対し $10_d - x$ を x **の** N **の補数**、$10_d - 1 - x = 1_d - x$ を x **の** $N-1$ **の補数**という。補数を使うことにより、減算を加算で行うことができるので、整数をコンピュータ内部で表現する際に用いられる。

例 7.2 3 桁の 10 進数を考える。123 に対し、10 の補数は 877(= 1000 − 123)　であり、9 の補数は 876(= 999 − 123) である。−123 に対し、10 の補数は 123(= 1000 − (−123) の下 3 桁) であり、9 の補数は 122(= 999 − (−123) の下 3 桁) である。356 − 123 を 3 桁の 10 進数で 10 の補数を用いて計算すると、

$$356 - 123 \equiv 356 + 877 = 1233 \equiv 233$$

となる。ここで、≡ は 3 桁の固定桁数表示で等しいことを表す。

7.4.0.3　2 の補数　補数とは、プラスの値でマイナスの値を表すための工夫である。補数を得るためには、2 進数で表された各桁の数値をすべて反転し、その結果に 1 を加える。

2 進数でマイナスの数を表す方法の一つは、最上位桁を符号のために使うことである。この最上位桁のことを「符号ビット」と読んだりする。符号ビットが 0 のときは正の数であり、符号ビットが 1 のときは負の数であると決めておく。1 は 8 桁の 2 進数では 0000 0001 であるが、マイナス 1 は 2 進数では 1111 1111 となる (すべて反転させて 1 を加える)。

コンピュータは、引き算を行うときに、内部的には足し算として演算する。2 の補数を使うと引き算を足し算として考えることができるので、コンピュータを設計する際に回路が簡単になるというメリットが生まれる。

例えば、8 桁の 2 進数で 1 は 0000 0001 であり、マイナス 1　は (2 の補数で) 1111 1111 である。この両者を足し合わせれば 1 0000 0000 となり、桁上がりを無視すれば 0 となって 1 + (-1) = 0 が計算される。コンピュータの世界では桁上がりを無視するということは良く行われていることである。

上の通り 8 桁の 2 進数で 11111111_2 は -1 を表す。11111110_2 は 1 を引いてから反転させればよい (2 補数を作るときと逆操作なので 1 を引いてから反転させる)。2 の補数表現でマイナスの絶対値がもっとも大きい数は 10000000_2 である。なぜならこの数から 1 を引いて (01111111_2) から、0-1 を反転させると 10000000_2 すなわち 128_{10} である。一方プラスの方で絶対値最大は 01111111_2 であるから 127 である。このように、2 の補数表現ではマイナスの絶対値の方がプラスの絶対値よりも 1 だけ大きくなる。

d 桁 (ビット) の 2 進数 $b(>0)$ を考える。$10_d - b$ (1 の後 0 が d 個続く数から b を引く) が b の 2 の補数 (two's complement) である。0 の 2 の補数は 0 である。

演習 7.2 問 1　10 進数で 39 を 2 進数として表せ。
問 2　-39 を 2 進数で 2 の補数で表せ。
問 3　問 1 と問 2 の答えを足して 0 になることを確かめよ。

8 ビットの 2 進数 $b(>0)$ を考える。b の 2 の補数は、$100000000_{(2)} - b$ である。
b の 2 の補数は

1. すべてのビットを反転 (0 を 1、1 を 0)　(= 1 が d 個続く 2 進数から b を引く)
2. 1 を加える　　(= 1 の後 0 が d 個続く 2 進数から b を引く)

により得られる。

例 7.3 8 ビットの 2 進数 00110101 の 2 の補数は 11001011 である。

1	1	0	0	1	0	1	0	すべてのビットを反転
1	1	0	0	1	0	1	1	1 を加える

7.4.0.4 1 の補数 b を d 桁 (ビット) の 2 進数とする。$1_d - b$ (1 が d 個続く数から b を引く) が b の 1 の補数 (one's complement) である。

8 ビットの 2 進数 $b(>0)$ を考える。b の 1 の補数は、$11111111 - b$ である。b の 1 の補数はすべてのビットを反転 (0 を 1、1 を 0) して得られる。

例 7.4 8 ビットの 2 進数 0011 0101 の 1 の補数は 1100 1010 である。

演習 7.3 [情報処理技術者試験 (基本情報技術者)2002 年度春期午前]

4 ビットの 2 進数 1010 の 1 の補数と 2 の補数の組合わせはどれか。

	1 の補数	2 の補数
ア	0101	0110
イ	0101	1001
ウ	1010	0110
エ	1010	1001

注 7.2 2006 年度以降の情報処理技術者試験の問題冊子・配点割合・解答例・採点講評は主催者の独立行政法人情報処理推進機構が Web

`http://www.jitec.jp/`

で公開している。

情報処理技術者試験 (基本情報技術者) および第 2 種情報処理技術者試験の問題と解答例は

東京理科大学情報処理技術者試験研究

`http://www.rs.kagu.tus.ac.jp/infoserv/j-siken/`

で見ることができる。

2002 年度以降の情報処理技術者試験 (基本情報技術者) の問題と解答例と解説は

`http://情報処理試験.jp/`

で見ることができる。

演習 7.4 問1　1バイトのデータで0のビット数と1のビット数が等しいもののうち，符号なしの2進整数として見たときに最大になるものを，*10* 進整数として表したものはどれか。

ア　*120*　　イ　*127*　　ウ　*170*　　エ　*240*

演習 7.5 [基本情報処理技術者試験 2002 年度秋期午前]

負数を 2 の補数で表す 16 ビットの固定小数点方式で、絶対値が最大である数値を 16 進数として表したものはどれか。

ア 7FFF　　イ 8000　　ウ 8001　　エ FFFF

演習 7.6　*1.* 自分の学生番号の下二桁を *16* 進数と見なして、*10* 進数に変換せよ

2. 自分の学生番号の下二桁を *16* 進数と見なして、*8* 進数に変換せよ

3. 自分の学生番号の下二桁を *16* 進数と見なして、*2* 進数に変換せよ

4. *3.* で求めた *2* 進数の *1* の補数を求めよ
5. *3.* で求めた *2* 進数の *2* の補数を求めよ
6. 自分の携帯電話番号の下二桁を *16* 進数とみなして、*10* 進数に変換せよ
7. 自分の携帯電話番号の下二桁を *16* 進数とみなして、*8* 進数に変換せよ
8. 自分の携帯電話番号の下二桁を *16* 進数とみなして、*2* 進数に変換せよ
9. *8.* で求めた *2* 進数の *1* の補数を求めよ
10. *8.* で求めた *2* 進数の *2* の補数を求めよ
11. *3.* で求めた数に *10.* で求めた数を足し、引き算の答となっていることを確かめよ

演習 7.7 次の表 *9* の空欄を埋めよ *(*ビット表現の欄は、塗り潰すこと*)*。

表 9: 2, 8, 10, 16 進数対応表

10 進数	8 進数	16 進数	2 進数	2 進数のビット表現
0	00	0x00	0	□□□□□□
1	01	0x01	1	□□□□□■
2	02	0x02	10	□□□□■□
3	03	0x03	11	□□□□■■
4	04	0x04		□□□■□□
5	05	0x05		□□□■□■
6	06	0x06	110	□□□■■□
7	07	0x07		□□□□□□
8	010	0x08	1000	□□□□□□
9	011	0x09		□□■□□■
10		0x0a	1010	□□■□■□
11	013	0x0b		□□□□□□
12	014			□□■■□□
13	015	0x0d	1101	□□■■□■
14				□□□□□□
15		0x0f		□□■■■■
16	020	0x10	10000	□□□□□□

演習 7.8 次の n 進数を 10 進数に直せ。

$10110_{(2)} =$

$8f_{(16)} =$

演習 7.9 次の *10* 進数を n 進数に直せ。

$256_{(10)} =$ (16)

$127_{(10)} =$ (2)

演習 7.10 次の *16* 進数を *2* 進数に直せ。*16* 進数の各桁をそれぞれ *4* 桁の *2* 進数に直せば良いことに注意せよ。

$74_{(16)} =$

$68_{(16)} =$

$53_{(16)} =$

$12_{(16)} =$

$93_{(16)} =$

演習 7.11 次の *2* 進数を *8* 進数に直せ。*2* 進数を *3* 桁づつに区切ってそれを *8* 進数に変換すれば良いことに注意せよ。

$01011000_{(2)} =$

$00100010_{(2)} =$

$01100100_{(2)} =$

$00010101_{(2)} =$

$01111010_{(2)} =$

演習 7.12 次の *2* 進数を *16* 進数に直せ。*2* 進数を *4* 桁づつに区切ってそれを *16* 進数に変換すれば良いことに注意せよ。

$01011000_{(2)} =$

$00100010_{(2)} =$

$01100100_{(2)} =$

$00010101_{(2)} =$

$01111010_{(2)} =$

演習 7.13 次の *8* 進数を *2* 進数に直せ。*8* 進数の各桁を *3* 桁の *2* 進数に変換すれば良いことに注意せよ。

$152_{(8)} =$

$364_{(8)} =$

$725_{(8)} =$

演習 7.14 次の *16* 進数の引き算を *2* の補数を使って足し算に直して計算せよ。そしてそれの答えが *10* 進数に直した場合の引き算と一致していることを確かめよ。

$14_{(16)} - 49_{(16)} =$

$22_{(16)} - 65_{(16)} =$

$61_{(16)} - 96_{(16)} =$

$89_{(16)} - 84_{(16)} =$

$56_{(16)} - 25_{(16)} =$

$09_{(16)} - 73_{(16)} =$

$41_{(16)} - 47_{(16)} =$

$12_{(16)} - 95_{(16)} =$

$64_{(16)} - 72_{(16)} =$

$41_{(16)} - 55_{(16)} =$

$82_{(16)} - 58_{(16)} =$

$35_{(16)} - 23_{(16)} =$

7.5　符号拡張

符号拡張とは、例えば 8 桁の 2 進数を 16 桁や 32 桁, 64 桁の 2 進数に変換することである。

正の数の場合拡張は簡単である。単に上位ビットを 0 で埋めれば良い。例えば 01111111_2 は 0000000001111111_2 である。

負の場合には上位ビットすべてを 1 で埋めれば良い。例えば 11111111_2 は 1111111111111111_2 とする。

すなわち符号拡張の場合には、正負の符号に関係なく符号ビット (0 か 1 かである) で上位桁を埋めていけば良い。

7.6　2 進数による少数点表示

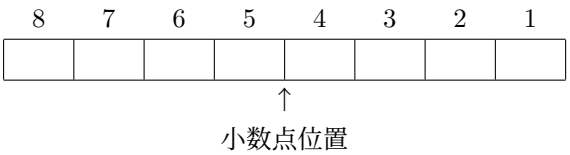
10 進数の小数点、たとえば 0.6_{10} は 10^{-1} が 6 つあることを表している。同様に、0.52_{10} は 10^{-1} が 5 つ, $10^{-2} = 1/100$ が 2 つあることを表している。この関係は 2 進数でも同じである。例えば 0.11 は $2^{-1} = 0.5_{10}$ が 1 つと $2^{-2} = 0.25_{10}$ が 1 つ、すなわち 0.75_{10} を表している。

7.6.1　固定小数点と浮動小数点

小数点第何位で丸めると指定した方法を固定小数点 (fixed point) 方式、有効桁数を指定した方法を浮動小数点 (floating point) 方式という。整数は小数第 1 位を丸めた固定小数点表示と考えることができる。

演習 7.15 [第二種情報処理技術者試験 1999 年度秋期午前]

10 進数 −5.625 を、8 ビット固定小数点形式による 2 進数で表したものはどれか。ここで、小数点位置は、4 ビット目と 5 ビット目の間とし、負数は 2 の補数表現を用いる。



ア 01001100　　イ 10100101　　ウ 10100110　　エ 11010011

例 7.5 10 進小数 123.456 は小数点第 4 位を四捨五入した固定小数点表示、1.23456×10^2 は有効桁数 6 桁の浮動小数点表示である。1.23456 を仮数 (mantissa)、2 を指数 (exponent)、10 を基数 (exponent base) という。

浮動小数点表示の方法は何通りもある。例 7.5 の数は

$$0.0123456 \times 10^4 = 0.123456 \times 10^3 = 1.23456 \times 10^2 = 12.3456 \times 10^1 = 123.456 \times 10^0 = \cdots$$

と表示できる。小数点 (radix) を最初の 0 以外の桁の直後におく方法、上記の場合 1.23456×10^2、を正規化形式 (normalized form) という。

例 7.6 $4.1_{(10)}$ を 2 進小数で表すと、$100.000110011001100110011001100\cdots_{(2)}$ となる。正規化浮動小数点表示は、$1.00000110011001100110011001100\cdots \times 2^2$ である。

たとえば、$0.10000011001100110011001100\cdots \times 2^3$ を仮数部 10 桁で表す場合 11 桁目が 1 だから切り上げて、0.1000001101 が仮数部である。

7.6.2　IEEE(アイトリプルイー) 標準 754

CPU に Intel またはその互換製品が使われている PC、Macintosh、多くのワークステーションでは、浮動小数点は IEEE 標準 754(IEEE Standard 754-1985) の規格[15]が用いられている。

7.6.2.1　浮動小数点数の集合

IEEE 標準 754 では、浮動小数点数として 4 種類の数が用意されている。

$s = 0$ または 1 は符号ビットで $s = 0$ のときプラス、$s = 1$ のときはマイナスである。$p,\quad E_{\min},\quad E_{\max}$ はあらかじめ決められた 10 進整数である。$b_1, b_2, \cdots, b_{p-1}$ は 0 または 1 を表す。

正規化 2 進浮動小数点数 $(-1)^s 2^E \left(1 + b_1 2^{-1} + + b_2 2^{-2} \cdots + b_{p-1} 2^{1-p}\right),\quad E_{\min} \leqq E \leqq E_{\max}$
この数は $s = 0$ のとき $2^E(1.b_1 b_2 \cdots b_{p-1})$ と表し、$s = 1$ のとき $-2^E(1.b_1 b_2 \cdots b_{p-1})$ と表す。

零 $(-1)^s 2^{E_{\min}-1} \left(0 \times 2^{-1} + 0 \times 2^{-2} + \cdots + 0 \times 2^{1-p}\right)$
この数は $s = 0$ のとき 0 と表し、$s = 1$ のとき -0 と表す。

非正規化 2 進浮動小数点数 $(-1)^s 2^{E_{\min}} \left(b_1 \times 2^{-1} + b_2 2^{-2} + \cdots + b_{p-1} 2^{1-p}\right)$
この数は $s = 0$ のとき $2^{E_{\min}}(0.b_1 b_2 \cdots b_{p-1})$ と表し、$s = 1$ のとき $-2^{E_{\min}}(0.b_1 b_2 \cdots b_{p-1})$ と表す。

NaN (非数 Not a Number) $\infty,\quad -\infty,\quad \mathrm{NaN}$

正規化 2 進浮動小数点数の最大値は $b_1 = \cdots = b_{p-1} = 1, E = E_{\max}$ のときで、

$$2^{E_{\max}}(2^0 + 2^{-1} + 2^{-2} + \cdots + 2^{1-p}) = 2^{E_{\max}} \frac{1 - 2^{-p}}{1 - 2^{-1}} = 2^{E_{\max}}(2 - 2^{1-p})$$

である。正の最小数は $b_1 = \cdots = b_{p-1} = 0, E = E_{\min}$ のときで、$2^{E_{\min}}$ となる。

[15]IEEE 標準 754 の日本語訳は
`http://www-amano.aa.cs.keio.ac.jp/members/kawaguti/memofpu/biblio.ht ml`
にある。

非正規化 2 進浮動小数点数の最小値は、$b_1 = \cdots = b_{p-2} = 0, b_{p-1} = 1, E = E_{\min}$ のときで、

$$2^{E_{\min}} 2^{1-p} = 2^{E_{\min}+1-p}$$

となる。

浮動小数点数の演算結果の絶対値が正規化 2 進浮動小数点数の最大値を越えたときオーバーフロー (overflow)、非正規化 2 進浮動小数点数の最小値を下回ったときはアンダーフロー (underflow) という。オーバーフローが生じたときは ∞ または $-\infty$ で、アンダーフローが生じたとき 0 または -0 で置き換える。

IEEE 標準 754 は単精度 (single precision)、倍精度 (double precision)、拡張単精度 (extended single precision)、拡張倍精度 (extended double precision) の 4 通りのシステムがある。拡張精度は、ビット数 79 以上というように標準化されてない。

7.6.2.2　メモリ表現

IEEE 754 の浮動小数点数はメモリには、以下のように格納する。

1. 先頭 1 ビットが符号 (正は 0、負は 1)
2. 正規化 2 進浮動小数点数の仮数部 (mantissa) は先頭ビット (leading digit) の 1 を省略し、1 ビット節約する。指数部 (exponent) はバイアス (bais) と呼ばれる定数を指数に加え非負になるようにし、その値を 2 進表現する。実際の指数 (unbiased exponent) にバイアスを加えたものを下駄履き指数 (biased exponent) という。
3. 非正規化 2 進浮動小数点数の仮数部は先頭ビットの 0 を省略し、1 ビット節約する。指数部は全ビット 0 する。
4. 0 は符号は任意、指数部と仮数部の全ビット 0 とする。
5. 指数部の全ビットが 1 で、仮数部の全ビットが 0 のときは ∞ とする。指数部の全ビットが 1 で、仮数部が非 0 のときは NaN(非数) とする。

したがって、指数部の全ビットが 0 か全ビットが 1 のときは特別の数、そうでないときは正規化浮動小数点数となる。最小の正の正規化数は指数部が 1 で仮数部の全ビットが 0、最大の正規化数は指数部がバイアスの 2 倍で仮数部の全ビットが 1 である。

- 単精度浮動小数点形式

†	指数部	仮数部
	8 ビット	23 ビット

† 符号部 1 ビット
指数部 実際の指数にバイアス $01111111_{(2)} = 127_{(10)}$ を加える。
仮数部 先頭の 1 を省略

- 倍精度浮動小数点形式

‡	指数部	仮数部
	11 ビット	52 ビット

‡ 符号部 1 ビット
指数部 実際の指数にバイアス $01111111111_{(2)} = 1023_{(10)}$ を加える。
仮数部 整数部分の 1 を省略

例 7.7　$4.1_{(10)} = 1.00000110011001100110011001100\cdots_{(2)} \times 2^2$ を IEEE 754 の単精度規格で表す。

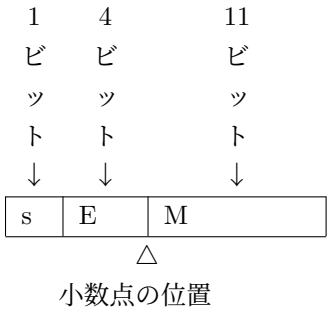
指数部は2+127 = 129 を2進数で表した 10000001 となる。仮数部は、1.00000110011001100110011001100 $\cdots$ の小数第24位が0なので、24位以下を切り捨て、先頭の1. を除いた、00000110011001100110011 である。

符号	指数部	仮数部
0	10000001	00000110 01100110 0110011

16 進数で表すと 4083 3333 となる。

演習 7.16 [第 2 種情報処理技術者試験 2000 年度春期午前]

数値を 16 ビットの浮動小数点で、図に示す形式で表す。10 進数 0.375 を正規化した表現はどれか。ここで正規化とは仮数部の最上位桁が 0 にならないように指数部と仮数部を調節する操作である。



S : 仮数部の符号 (0:正、1:負)
E : 指数部 (2 を基数とし、負数は 2 の補数で表現)
M : 仮数部 (2 進数 絶対値表示)

ア	0	0001	11000000000
イ	0	1001	11000000000
ウ	0	1111	11000000000
エ	1	0001	11000000000

7.7 alphabet の文字コード

alphabet を表現するだけだったら、7 bit で十分である。ISO 646-1991[16] と ANSI X3.4-1986[17] で規定されている 128 文字 (すなわち 7 bit)。128 文字中印刷可能文字は 94 文字。

以下のように入力 (下線部分をタイプ) すると alphabet の文字コードを調べることができる。

```
$ echo -n ''asakawa'' | hexdump -C
0000000 61 73 61 6b 61 77 61 0a
0000008
```

この出力の意味は、最初の 8 桁の数字は無視して 61 = a, 73 = s すなわち a の文字コードが $61_{(16)}$ (16 進数), s の文字コードが $73_{(16)}$ であることを意味している。以下同様に、a = 61, k = 6b, a = 61, w = 77, a = 61 文字列の最後に改行コード ($= 0a_{(16)}$) が一つ入っている。

世界中のコンピュータの中には、7 bit 文字だけしか表示できないものもある。8 bit 目は不要なだけでなく、**危険** であると言われることがある。これが漢字文化圏の人達には問題となった。

[16] ISO: International Organization for Standardization
[17] ANSI: American National Standards Institute

演習 7.17 自分の名前 *(alphabet)* の文字コードを調べて以下の表を埋めよ。

alphabet	*code(16 進数)*	*alphabet*	*code(16 進数)*
例 *a*	*0x61*		

演習 7.18 次の文字のコードを調べて書け。

A=

z=

0=

“ ”*(*空白*)*=

7.8 符号化文字集合

コンピュータでは、文字は 0 と 1 の並びで表される。0 と 1 の並びを 2 進数による数と考えると、文字と数の対応を決めることが必要になる。

文字の集まりを文字集合といい、文字集合に属す文字と数の 1 対 1 の対応まで定めたものが符号化文字集合 (coded character set) である。

文字集合の例

常用漢字表 1981 年 10 月 1 日に内閣告示第 1 号「常用漢字表」により発表された漢字使用の基準。「法令・公用文書・新聞・雑誌・放送等、一般の社会生活で用いる場合の、効率的で共通性の高い漢字を収め、分かりやすく通じやすい文章を書き表すための漢字使用の目安」（同告示）を示す。1945 字からなる。

人名用漢字別表 日本における戸籍に子の名として記載できる漢字のうち、常用漢字に含まれないものを言う。法務省により戸籍法施行規則別表第二として 983 字が指定されている。

学年別漢字配当表 教育漢字、または学習漢字といわれ、小学校 6 年間のうちに学習することが文部科学省によって定められている 1006 字の漢字の総称。『小学校学習指導要領』の付録にある。

出典: フリー百科事典『ウィキペディア（Wikipedia）』
`http://ja.wikipedia.org/wiki/`

7.8.1 ASCII(アスキー)

1963 年アメリカ標準協会 (American Standard Association) が 4、6、7 ビットの符号の標準化を行った。1968 年に 7 ビットに一本化されたのが ASCII 符号 (American Standard Code for Information Interchange 情報交換用米国標準符号) である。1967 年には ISO 符号となり、1969 年には JIS 符号 (JIS X0201) にも採り入れられた。

0 と 1 を 7 個組み合わせて作る符号で、34 個の制御文字 (改行やバックスペースなど)、94 個の図形文字 (記号、数字、アルファベット)、計 128(= 2^7) 文字からなる。

ターミナルエミュレータ上から

```
$ man ascii
```

とするとアスキーコードが 8 進 (octal)、16 進 (hexadecimal)、10 進 (decimal) で表示される。(終了は q を押す。)

アルファベットの大文字 2 文字ないし 3 文字で表わした 00_{16} から 20_{16} までと $7F_{16}$ は、制御文字である。制御文字は、当時の技術であるテレタイプ (伝送装置付きのタイプライター) や紙テープを強く意識している。たとえば、空白はタイプライターでは打つのではなく印字ヘッドを右に移動させるだけであるので制御文字として定義されている。

ASCII の制御文字以外の 21_{16} から $7E_{16}$ のうち (16 進表示で) 23 24 40 5B 5C 5D 5E 60 7B 7C 7D 7E の 12 個を各国の選択にしたのが、ISO 符号 (ISO 646[18]) である。

たとえば、英国 (BS 4730) では 23_{16} を £(ポンド記号) としている[19]。

[18] 1973 年以降、制御文字は ISO 2022 など別の規格に譲った。

[19] 現在英国では、ISO 646 を 8 ビットに拡張した ISO 8859-1 が使われている。ISO 8859-1 にはヨーロッパで使われている à、á などのアクセント付きの文字が 62 文字含まれている。

ビット、バイト、オクテット

コンピュータの内部では、電気信号のオン/オフなど (時間的に不連続な離散量) のデータで処理される。オフを 0、オンを 1 と数値で置き換えると、データは 0 と 1 の列で表わすことができる。
0 と 1 の数をビット (binary digit : bit) で数える。1 文字を表す情報量の単位を 1 バイト (byte) という。通常 1 バイトは 8 ビットであるが、混乱を避けるため 8 ビットは 1 オクテット (octet) とも呼ばれる。

7.8.2 JIS X 0201

1969 年制定の JIS X 0201(1987 年までは JIS C 6220) は、7 ビット版と 8 ビット版がある。8 ビット版では 20_{16} から $7F_{16}$ までは 2 文字[20] を除いて ASCII と一致する。$A1_{16}$ から DF_{16} までは 8 ビット (いわゆる半角) の句読点とカタカナである。

7.8.3 JIS X0208

7 ビット、8 ビット、2 バイトの符号化文字集合からなる。2 バイト符号化文字集合は、6879 文字の図形文字 (特殊文字、数字、ラテン文字、平仮名、片仮名、ギリシャ文字、キリール文字、漢字、罫線素片) とそれらのビット組合わせの対応を規定する。漢字には第 1 水準漢字集合 2965 文字、第 2 水準漢字集合 3390 文字、計 6355 文字を含む。

2 バイトの符号化文字集合は、各文字には区と呼ばれる 1 から 94 までの番号と、点と呼ばれる 1 から 94 までの番号がふられていて、区と点の組合わせで、文字の対応を取る。16 区 1 点の文字「亜」の区点は「16-01」と表す。

区を列、点を行とする $94 \times 94 (= 8836)$ 文字を成分とする行列である。

点 \ 区	1	···	16	···	94
1			亜		
2			唖		
⋮					
94			蔭		

7.8.4 JIS X0212

JIS X0208 に含まれない 6067 字の文字集合で 1990 年に制定された。JIS 補助漢字とも呼ばれる。

[20] ASCII では $5C_{16}$ はバックスラッシュ\ であるのに対し、JIS X0201 では円記号¥である。

7.8.5　JIS X0213

JIS X0208 の拡張を目的とした文字コード規格で 2000 年に制定され 2004 年に改正された。文字の位置を面区点で表すところが JIS X0208 と異なる。

JIS X0213 の 1 面には JIS X0208 の文字をすべて収録し空き領域にも文字を埋め込んでいる。2 面では JIS X0212 で使用している区点は避けている。

7.8.6　ユニコード (Unicode)

世界中の文字を単一の文字コードにしようということで作られた符号化文字集合である。非営利団体のユニコードコンソーシアム[21](The Unicode Consortium) により開発調整が進められている。1993 年に ISO でも ISO/IEC 10646 の一部として標準化された。

7.8.7　ISO 10646

ISO 10646[22]は世界中の文字を一意で表示することを目標とする 31 ビットの文字コードである。国際符号化文字集合 (UCS:Universal multiple-octet coded Character Set) という。最初の 7 ビットが群 (Group)、次の 8 ビットが面 (Plane)、その次の 8 ビットが区 (Row)、最後の 8 ビットが点 (Cell) である。00000021 から 0000007E までは、ISO 646(ASCII) の 21 から 7E までに一致するので、ISO 646 の拡張になっている。

00 群 00 面を基本多言語面 (BMP、Basic Multilingual Plane) とよび Unicode と同じコードになっている。

ISO 10646 の漢字は、00004E00 から 00009FA5 までに 20902 文字、00003400 から 00004DB5 までに 6582 文字の追加、00020000 から 0002A6D6 に 42711 文字追加されている。中国・台湾・日本・韓国・ベトナムの文字を字形によってコードをふっている。5 欄併記の形式を取っており、1 つのコードで最大 5 つの字形が併記されている。

7.9　文字コード

同じ符号化文字集合を対象としても、コンピュータ上で文字を利用する際、異なるバイト表現をとる文字符号化方式 (character encoding scheme) がある。ここでは、文字符号化方式を文字コードとよぶ。

7.9.1　日本語コード

日本語のコード化は 5 通りの方法が広く用いられている。

[21]ゼロックス社が提唱し、Microsoft、Apple、IBM、ジャストシステム、サン・マイクロシステムズ、などが参加する

[22]JIS X 0221 は ISO/IEC 10646 Part 1 の翻訳である。

符号化文字集合	文字コード	通称	使用例
ASCII, JIS X 0201, JIS X 0208	ISO-2022-JP †	JIS	電子メール
JIS X 0201、JIS X0208	Shift_JIS ‡	SJIS	古い Windows
ASCII、JIS X 0208	EUC-JP††	EUC	古い UNIX
ISO 10646	UTF-8	UTF-8	Web
	UTF-16	UTF-16	Mac OS X、Linux、Windows

†JIS X 0213-2004 に対応した ISO-2022-JP-2004 ができている。
‡JIS X 0213-2004 に対応した Shift_JIS-2004 ができている。
††JIS X 0213-2004 に対応した EUC-JIS-2004 ができている。

7.9.2 ISO 2022-JP

7.9.2.1 ASCII(アスキー文字) 20_{16} から $7E_{16}$ は ASCII または JIS X0201 をそのまま使う。

7.9.2.2 漢字 JIS X 0208 漢字の 1 バイト目は、区の番号に 32_{10} を加えた数、2 バイト目は点の番号に 32_{10} を加えた数である。

例 7.8 16 区 1 点の「亜」の 1 バイト目は $48_{10} = 30_{16}$、2 バイト目は $33_{10} = 21_{16}$ である。したがって、3021_{16} である。

7.9.2.3 文字列 JIS X 0201 に入るとき、ESC-(-J(アスキーコードで、1B 28 4A) とする。
JIS X 0208 の 2 バイト文字にはいる前に、ESC-$-B(アスキーコードで、1B 24 42) とする。
ASCII に戻るときは、ESC-(-B(アスキーコードで、1B 28 42) とする。
ESC-(-J、ESC-$-B、ESC-(-B はエスケープシークエンス escape sequence と呼ばれる。初期状態では ASCII であるので、ASCII から始まるときはエスケープシークエンスはつけない。文字列の最後は必ず ASCII に戻る。

例 7.9 「a 亜」(ASCII の英小文字 a、漢字の亜) を ISO-2022-JP で表すと、

61 1B 24 42 30 21 1B 28 42

となる。

7.9.3 日本語 EUC

7.9.3.1 ASCII(アスキー文字) 20_{16} から $7E_{16}$ は ASCII をそのまま使う。

7.9.3.2 漢字 ASCII コードは先頭のビットが 0 なので、EUC コード (Extended Unix Code) では JIS X 0208 の 2 バイト文字 (JIS 漢字という) を表す場合 ISO 2022-JP の先頭のビットを 1 にする。JIS 漢字のコードに 8080_{16} を加えると得られる。
JIS X 0212 の補助漢字は 1 バイト目に制御文字 $8F_{16}$ を置くので 3 バイト用いる。

例 7.10 「亜」の JIS コードは 3021_{16} であるので、8080_{16} を加えた $B0A1_{16}$ が EUC コードである。

7.9.3.3 文字列 アスキー文字はそのまま、漢字は JIS 漢字の先頭ビットを 1 に変えて並べる。

EUC コードは、エスケープシークエンスを使わないので ISO-2022-JP に比べ必要なメモリが節約になる。

例 7.11 「a 亜」を EUC-JP で表すと、61 B0 A1 となる。

演習 7.19 「東女 TWCU」の ISO-2022-JP コードと EUC-JP コードを求めよ。ただし、「東」の区点は 37-76、「女」の区点は 29-87 である。また、「女」と「T」の間に空白はない。

7.10 ツール

7.10.1 文字コードの相互変換

漢字コードの相互変換は、Mac OS X では Jedit で保存の際文字コードを指定することにより行える。JIS を指定して保存すると、アルファベットや記号は、ASCII ではなく JIS X 0201 としているので、エスケープシークエンスは、1B 28 4A である。

ターミナルで ISO-2022-JP、EUC-JP、Shift JIS、UTF-8 は、`iconv` または `nkf` というプログラムで相互変換を行うこともできる。*input-file* という漢字コードが ISO-2022-JP のファイルを UTF-8 に変換するには、以下のように行う。

ISO-2022-JP を UTF-8 に変換

```
$ iconv -f ISO-2022-JP -t UTF-8 ファイル名 > output-file
```

利用できる文字コードは `iconv -ℓ` により表示できる。

```
$ nkf --utf8 ファイル名> output-file
```

7.10.2 漢字の区点を調べる

Mac OS X ではことえりの文字パレットで Unicode、UTF8、JIS X0213 面-区-点、Shift JIS(JIS X0208) を見ることができる。

文字パレットの表示

文字パレットは、メニューバーの「あ」「A」または星条旗をクリックして表示されるプルダウンメニューの「文字パレット」をクリックして表示できる。

例 7.12 例えば、「東」の区点は次のように調べる。
「表示: 日本語」→「画数」→　4 画「木」→「部首別」→　4 画「東」→「JIS 面-区-点」1-37-76
JIS X0213 の 1 面 37 区 76 点である。JISX0208 では 37 区 76 点となる。

プルダウンメニューに「文字パレット」が無い場合は以下のような操作で表示できる。

1. [”言語環境”を開く]
2. 入力メニューにある「文字パレット」にチェックをつける。
   (US にもチェックをつけると星条旗になる。)
3. ”言語環境”を閉じる。

多くの漢和辞典 (たとえば、電子辞書に入っている「漢字辞典」「漢字源」など) にも区点と JIS、Shift JIS コードが載っている。

参考文献

[1] 安岡孝一、日本における最新文字コード事情 (前編)、システム/制御/情報、Vol 45, No. 9, 528-435
http://kanji.zinbun.kyoto-u.ac.jp/~yasuoka/publications/ISCIE2001.pdf

[2] 安岡孝一、日本における最新文字コード事情 (後編)、システム/制御/情報、Vol 45, No. 12, 687-694

[3] 安岡孝一・安岡素子、符号化文字集合の歴史と体系、インターネット時代の文字コード、pp.1-62、共立出版、2002

[4] フリー百科事典『ウィキペディア（Wikipedia）』http://ja.wikipedia.org/wiki/