1.コンピュータ科学基礎Ⅰ：情報の表現

1.1 数値やデータの表現

コンピュータは、基本的にはすべての情報を、”0”と”1”の値によって表現する。この”0”と”1”で表現できる情報を**ビット**と呼ぶ。

1 ビットとは、一つの”0”と”1”の値 0 で表現できる情報量の最小単位をいう。
8 つのビットを一つにまとめた単位を**バイト**という。

(1) 2 進数／16 進数と基数変換

コンピュータの内部では、情報をビットで表し、”1”または”0”の数値を扱う。そこで、数値を２進数で表し、２進数の一桁が１ビットに相当する。２進数、10 進数、16 進数などの２,10,16 を数の基数と言う。**基数変換**とは、数の基数を異なる基数に変換することをいう。普段の生活では主に 10 進数が使われている。一方、コンピュータでは２進数が基礎になっている。また、２進数では桁数が増えて識別しにくくなるので 16 進数が用いられる。これらの 10 進数、２進数、16 進数の対応を以下に示す。

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

桁の重み

10 進数で、例えば、348.5 についてみると

$$348.5 = 3 \times 10^{2} + 4 \times 10^{1} + 8 \times 10^{0} + 5 \times 10^{-1}$$

となる。

（蛇足ながら、$10^2 = 100$、$10^1 = 10$、$10^0 = 1$、$10^{-1} = 0.1$）

16 進数から 10 進数への変換

16 進数で、例えば、2D.8 を 10 進数へ基数変換すると、

$$\begin{aligned} 2D.8 &= 2 \times 16^1 + D \times 16^0 + 8 \times 16^{-1} \\ &= 2 \cdot 16 + 13 \cdot 1 + 8 \cdot \frac{1}{16} \\ &= 32 + 13 + 0.5 \\ &= 45.5 \end{aligned}$$

2 進数から 10 進数への基数変換

２進数で、例えば、1011.0111 についてみると

$$\begin{aligned} 1011.0111 &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} \\ &= 8 + 2 + 1 + 1 \cdot 0.25 + 1 \cdot 0.125 + 1 \cdot 0.0625 \\ &= 11.43755 \end{aligned}$$

10 進数から 2 進数への基数変換

10 進数で、21.625 についてみると

$$\begin{aligned} 21.625 &= 1 \cdot 2^4 + 5.625 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 5.625 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1.625 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1.625 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0.625 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0.125 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0.125 \\ &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \\ &= 10101.101 \end{aligned}$$

となる。

21.625 を整数部と小数部に分けて、

(整数部)=21

について、２で割った余りと商を求めると、

(余り)=1　--- 1 桁目

(商)=10

(商)=10 を２で割った余りと商を求めると、

(余り)=0　--- 2 桁目

(商)=5

(商)= 5 を２で割った余りと商を求めると、

(余り)=1　--- 3 桁目

(商)= 2

(商)= 2 を２で割った余りと商を求めると、

(余り)=0　--- 4 桁目

(商)= 1

（商)= 1 はこれ以上２で割り切れないのでこれが５桁目となる。

これをまとめると、整数部の２進数は、10101 となる。

小数部 0.625 を２倍すると 1.25 となり整数部と小数部に分けると

(整数部)=1　--- 小数 1 桁目

(小数部)=0.25

小数部 0.25 を２倍すると 0.50 となり整数部と小数部に分けると

(整数部)=0　--- 小数 2 桁目

(小数部)=0.50

小数部 0.50 を２倍すると 1.00 となり整数部と小数部に分けると

(整数部)=1　--- 小数 3 桁目

(小数部)=0　--- 小数 4 桁目

小数部が０になったのでここで終了し、２進数の小数部の 0.1010

10 進数の 21.625 は２進数では 10101.101 と対応する。

10 進数から 16 進数への基数変換

10 進数から 2 進数への基数変換と同様に、整数部／小数部ともに基数である 16 で割り、その剰余を 16 進数における各桁の値としていく。

2 進数から 16 進数への基数変換

$$
\begin{aligned}
11000011.0100 &= 1100\cdot 16^{1}+0011+16^{0}+0100+16^{-1} \\
&= C\cdot 16^{1}+3+16^{0}+4+16^{-1} \\
&= C3.4
\end{aligned}
$$

(2)負の数の表現

コンピュータの内部では”1”と”0”でしか表現できないため、負の値を示すマイナスの符号が使えない。そこで、負の数を表す場合には、**補数**を用いて表現する。補数には「基数の補数」と「基数－1 の補数」があり、基数の補数を負の数の表現として使われる。

n 進数の n の補数

n 進数で表現された数値 x の補数を求めることを考える。ここで、2 桁のみからなる桁数で 10 進数 x = 56 の 10 の補数を求める。

[1] 数 n-1 を桁数分並べる

10 進数を対象にしているので、n-1= 10-1 = 9 となり、56 については、2 桁の 99 をつくる。

[2] 数 n-1 を桁数から対象とする数を引く。

99 – 56 = 43

[3] [2] で得られた値に 1 をたす。

43 + 1 = 44

求めた数とその補数を足すと桁上がりでオーバーフローとなりゼロとなる(桁数が 2 桁における補数ということに注意)。

44 + 56 = 100

いま 2 桁のみで定義がなされているので、これは 00 すなわちゼロとなる。

2 進数の 2 の補数(2 の補数)による負の数の表現

前述のように、コンピュータ内部ではあらゆる値は最終的に 2 進数で表現されるため、負の数も 2 進数の 2 の補数(**2 の補数**)で表されることになる。この 2 の補数による数値の表現が利用される理由には、電子回路によって数値演算回路を構成するにあたって都合が良い点もあげられる。

0100 0001(65)を 2 の補数により負の数(−65)を表すと、

1111 1111 – 0100 0001 + 0000 0001 = 1011 1110 + 0000 0001 = 1011 1111

すなわち、0100 0001 の負の数は 1011 1111 となる。

2 の補数で表現された数の 10 進数への変換は、最上位のビットの持つ重みは負の数となると考えると分かりやすい。例えば、8 ビットの数値の場合、8 ビット目が持つ重みは通常は 2^7 であるが、2 の補数表現の場合は -2^7 となる。よって、先ほどの 1011 1111 は、以下の式で 10 進数である-65 に変換できる。

$$1\cdot(-2^7)+0\cdot 2^6+1\cdot 2^5+1\cdot 2^4+1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0=-65$$

(3)数値の表現範囲

コンピュータ内部では限られた(桁)数（ビット数）の数値を取り扱う。 8 ビット(1 バイト)数で表現できる数値の範囲は、0000 0000 ～ 1111 1111 の 256 種類であり、これをそのまま正の整数と対応させると、10 進数において 0 ～ 255 の数を表現できる。また、2 の補数で負の数表現するとすると、－128 ～ 127 の 256 種類であらわすことができる。

ここで、正の値は先頭ビットが”0”となり、負の値は先頭の値が”1”となるため、先頭ビットを**符号ビット**と呼ぶことがある。

ｎビットで表現できる値の範囲は、

正の数のみ表現する場合　　$0 \leftrightarrow 2^n - 1$

２の補数表現を用いた場合　　$-2^{n-1} \leftrightarrow 2^{n-1} - 1$

となる。

8 ビットの数値表現の範囲(2 の補数表現)

2 進数	16 進数	10 進数
1000 0000	80	-128
1000 0001	81	-127
…	…	…
1111 1111	FF	-1
0000 0000	00	0
0000 0001	01	1
…	…	…
0111 1101	7D	125
0111 1110	7E	126
0111 1111	7F	127

(5)浮動小数点表記

科学技術計算の分野では、非常に大きな数値や小さな数値を扱う必要がある。しかしながら、コンピュータが一度に扱えるビット数には限りがあり、小数点以上を表すためのビットと小数点以下を表すビットの両方を大量に持つことは現実ではない。そこで、限られたビット数（桁数）でよりおおきな数値や小さい数値を表現するために**浮動小数点表記**を用いる。

浮動小数点表記では、数値を

$$(\text{仮数}) \times (\text{基数})^{(\text{指数})}$$

で表現する。

例えば、$1234000000000 = (1234)\times(10)^{(9)}$ の基数を固定し、仮数と指数のみを扱えば、1234 と 9 のみでこの数を再現できる。

浮動小数点表記の正規化

浮動小数点表記を行う場合は、有効桁数を考慮して、正規化を行う必要がある。ここで**有効桁数**とは、意味を持つ数値が存在している桁数を意味する。例えば

$$0.0001\times10^{3} \text{ と } 0.1000\times10^{-1}$$

は同じ量を表す数値であるが、前者の有効桁数は 1 桁、後者は４桁である。そこで、有効桁数を最大にするには、仮数を小数点第１位に移動して指数表現する。この指数表現法を浮動小数点表記による正規化という。上記では右側が正規化表現である。

浮動小数点数の表現形式

浮動小数点の表現形式は様々な規格があるが、以下では標準化されている **IEEE 方式**(IEEE754)について説明する。IEEE 方式では、32 ビットを使って、

1 ビット	8 ビット	23 ビット
仮数部の符号(S)	指数部(E)	仮数部(M)

のビット列で表現する。実際の数値は

$$(-1)^{S}\times(1+M)\times2^{E-127}$$

により変換される。

仮数部の符号(S)

S=0（正)、S=1(負)

指数部(E)

指数の値は E より 127 を引いた値となる。E の最大値は、11111111 であり、10 進数で 255 に対応する。よって、指数の最大値は 255-127=128 となる。また、E の最小値は 00000000 であり、10 進数で 0 に対応する。よって、指数の最小値は 0-127=-127 となる。つまり、指数部では-127 から 128 までの数値をあらわすことができる。

仮数部(M)

仮数部の一桁目が 1 となるように桁を仮数部の左側につめて表現する。例えば 0.00101 は、1.01 と詰める。ここで整数部は常に 1 となるので、残りの桁数 01 のみを 23 ビットの仮数部(M)で記述する。このように表現することにより 23 ビットを用いて 24 ビットの仮数を表現することが可能となる。

$$(28)_{10} = (-1)^{0}\times(1.1100)\times2^{4}$$
$$= (-1)^{0}\times(1+0.1100)\times2^{10000011-01111111}$$

S=0

M=11000…….0

E=10000011

より

仮数部の符号(S)	指数部(E)	仮数部(M)
0	10000011	11000000…0

となる。

(6)誤差

浮動小数点表記では、有効桁数に限りがあるため、正確に表現できずに誤差が発生する。誤差には、次の種類がある。

丸め誤差	仮数部の桁数に限界があることによって、最小桁より小さい部分について四捨五入や切り上げ、切り捨てを行うために生じる誤差。 例： $1.000....00001\times 2^{7}$ -> $1.000....000\times 2^{7}$
情報落ち	絶対値が大きい値に対し、極端に小さい値を加減演算すると、小さいほうの値が計算から欠落することによる誤差を情報落ちという。数多くの計算を行う場合は、絶対値の小さい値から順に加算することによって回避する。 例： $1.110....000\times 2^{3}+1.000....000\times 2^{76} = 1.000....000\times 2^{76}$
桁落ち	絶対値の差が小さい値同士の演算において演算結果の有効桁数が少なくなる誤差。 例： $1.111....000\times 2^{14}-1.110....000\times 2^{14} = 1.000....000\times 2^{11}$ （仮数部の有効桁数 23 桁->20 桁になり、下位に 0 を挿入）
打切り誤差	循環小数などの計算を途中で打ち切ったために発生する誤差。

上記の誤差を減らすため、仮数部や指数部のビット数を増やした倍精度浮動小数点表記や 4 倍精度浮動小数点表記などの規格も存在する。なお、誤差の発生した結果をベースに次の演算を行うことを繰り返すと、誤差の蓄積で結果が大幅に変わってくることがあるので、演算の繰り返す時には注意すること。

1.2 様々なデータ表現

(1) 2 進化 10 進数

10 進数 1 桁 1 桁を個別に 2 進数に変換して、取り扱う方法。

BCD(Binary Coded Decimals)

10 進数の一桁を 4 桁の 2 進数(0000～1001 までを利用)で表す。

BCD コードは符号の概念がない。

例：127 は 0001 0010 0111 という 3 つの BCD で表される。

金額計算などで小数が発生した場合、２進数では、正確に表現できない場合がある。例えば、0.1 を小数で表すと 0.00011011001100...と循環小数になってしまう。そこで、BCD コードで表しておけば、0000.0001 となり正確に表現できる。金融機関で使われることが多いメインフレーム・コンピュータで利用されることが多い。

ゾーン 10 進数

10 進数の 1 桁を 1 バイト（＝8 ビット）、上位 4 ビットをゾーン部、下位 4 ビットを整数部として表現する。

整数部:BCD コード

ゾーン部:JIS コード形式では 0011、EBCDIC 形式では 1111 を割り当てる。

最下位桁のゾーン部(符号部):0 と正の場合は 1100、負の場合は 1101 を割り当てる。

例

JIS コード形式　　123→ 0011 0001 0011 0010 1100 0011
JIS コード形式　　-123→ 0011 0001 0011 0010 1101 0011
EBCDIC コード形式　123→ 1111 0001 1111 0010 1100 0011
EBCDIC コード形式 -123→ 1111 0001 1111 0010 1101 0011

パック 10 進数

BCD（2 進化 10 進数）コードを使い、10 進数の各桁の値を 4 ビットの 2 進数で表現する方法。1 バイト（＝8 ビット）に 10 進数の 2 桁を記憶できる。

符号は最下位バイトの下位 4 ビットに入れる。正負をどのようなビットの並びで表すかは、システムによって異なる。

例：

－123456→ 0000 0001 0010 0011 0100 0101 0110 0111

ただし、最下位 4 ビットの 0110 は負であること示す。

⑵文字の表現

任意のビット数の数値に文字を対応づけ、数値の並びで文字を表現する。改行、記号、空白なども文字として表現される。

例：ASCII コード

..., 48 = "0", 49 = "1", ..., 65 = "A", 66 = "B", 67 = "C", ..., 97 = "a", 98 = "b", ...

72 101 108 108 111 33 = "Hello!"

様々な文字コード

JIS コード	JIS(日本工業規格)で規格化された８ビットの文字コード体系。 英数字、カタカナ、各種記号など 256 種類が定められている。 漢字をあらわすことができる 16 ビットの JIS 漢字コードもあり、 SI（シフトイン）、SO(シフトアウト)という特別コードを使って、16 ビットの漢字コードの始まりと終わりを表す。
シフト JIS コード	JIS コードをもとにした文字コード体系。 16 ビットの漢字コードのデータと英数字、カタカナなどの８ビットコードが混在する場合に識別できる。過去の Microsoft が主に利用。 JIS コードにおける SI, SO なしに８ビットコードと 16 ビットコードを識別できるように､漢字コードの上位８ビットを８ビットコードと干渉しない領域に割り当てている。
ASCII コード(American Standard Code for Information Interchange Code)	ANSI(米国規格協会)で規格化された文字コード体系。 基本は７ビットで、拡張版では８ビットとなっている。 日本語は使用できない。 JIS コードのベースとなった。
EBCDIC (Extended BCD Interchange Code)	IBM が作成したメインフレーム（汎用コンピュータ）向けの文字コード体系。 金融機関等で便利なように、数値に誤差が出ない BCD を採用。
EUC(Extended Unix Code)	UNIX で利用されている可変長ビットの文字体系。 日本語表示も可能。
Unicode	ISO(国際標準機構)で標準化された、世界の多くの文字を表現するための 16 ビット文字コード体系。日本語を表示する文字コードは、JIS, シフト JIS, EUC からこれに移行しつつある。

文字コードの(おおまかな)歴史的な関係

<-古い　　->新しい

ASCII　->　->　->　->

->JIS(8-bit)　->JIS(16-bit)

->シフト JIS

->EUC

->EBCDIC　->　->　->

->Unicode

古くてもずっと利用され続けるコードもある。

(3)他の情報表現の例

他の色々な情報も数値に符号化して表現される。以下は、例を示す。

色の表現

色の三原色である赤(Red)、緑(Green)、青(Blue)の大きさを数値で示す。以下は、最大値を255(符号無し 8 ビットの最大値)とした例である。

赤： (R, G, B) = (255, 0, 0)

黄： (R, G, B) = (255, 255, 0)

黒： (R, G, B) = (0, 0, 0)

ダークブルー： (R, G, B) = (0, 0, 139)

スカイブルー： (R, G, B) = (135, 206, 235)

文字の表現と同様、色の表現にも様々な規格(CMYK など)がある。

画像の表現

画像を区切って画素単位で構成し、画素の情報を左上から順に並べる。以下は、5×8 で表したアルファベット I の白黒画像(2 値画像)の例である。

```
                01110
                00100
I       ->      00100       ->      011000100001000010001110
                00100
                01110
```

カラー画像では、各画素の情報は複数ビットで表すことになる。

(4)情報表現の補助単位

情報の世界では非常に大きな数値や非常に小さい数値を表すことも多々ある。そのような数値は乗数で表現したりもできるが、補助単位を用いればより短く表現できる(特に口述する場合)。以下は、現在の情報の世界でよく使われる補助単位を示す。なお、本来は 10 の乗数を基本として補助単位をつけるが、情報の世界では、2 の乗数で対応する 10 の乗数を近似することもある。

記号	読み方	乗数での表現	(2 の乗数での近似表現)
E	エクサ	10^{18}	($2^{60} \fallingdotseq 1.153\times10^{18}$)
P	ペタ	10^{15}	($2^{50} \fallingdotseq 1.126\times10^{15}$)
T	テラ	10^{12}	($2^{40} \fallingdotseq 1.010\times10^{12}$)
G	ギガ	10^{9}	($2^{30} \fallingdotseq 1.074\times10^{9}$)

M	メガ	10^6	($2^{20} \fallingdotseq 1.048\times10^6$)
k	キロ	10^3	($2^{10} \fallingdotseq 1.024\times10^3$)
m	ミリ	10^{-3}	($2^{-10} \fallingdotseq 0.977\times10^{-3}$)
μ	マイクロ	10^{-6}	($2^{-20} \fallingdotseq 0.954\times10^{-6}$)
n	ナノ	10^{-9}	($2^{-30} \fallingdotseq 0.931\times10^{-9}$)
p	ピコ	10^{-12}	($2^{-40} \fallingdotseq 0.905\times10^{-12}$)

例

音楽用 CD	約 74 分の演奏を記録	650,000,000 バイト = 650M バイト
DVD	約 2 時間 15 分の映像と音声を記録	470,000,000,000 バイト = 4.7G バイト
パソコン CPU	動作周波数が 1GHz	1 秒間に 1,000,000,000 回以上動作できるから、一つの動作に必要な時間は約 1 ナノ秒
	32nm プロセスで製造	最小加工寸法 32 ナノメートルの半導体回路製造技術で製造
京スーパーコンピュータ	演算能力が 10P FLOPS (FLoting point 0Peration per Second)	1 秒間に 10,000,000,000,000,000 回の浮動小数点演算が実行可能

練習問題

第 1 章

問題 1.1 16 進小数 3A.5C を 10 進数の分数で表したものは次のうちどれか。

ア $\frac{939}{16}$, イ $\frac{3735}{64}$, ウ $\frac{14939}{256}$, エ $\frac{14941}{256}$

(基 15 秋午前問 1)

問題 1.2 2 進浮動小数点表示で誤差を含まず表現できる 10 進数はどれか。

ア 0.2, イ 0.3, ウ 0.4, エ 0.5

(基 15 秋午前問 1)

問題 1.3 負数を 2 の補数で表す 8 ビットの数値がある。この値を 10 進数で表現すると -100 である。この値を符号なしの数値として解釈すると、10 進数ではいくつになるか。

ア　28,　　イ　100,　　ウ　156,　エ　228
（基 17 春午前問３）

問題 1.4　16 進小数 2A.4C と等しいものはどれか。
ア　$2^5+2^3+2^1+2^{-2}+2^{-5}+2^{-6}$　　イ　$2^5+2^3+2^1+2^{-1}+2^{-4}+2^{-5}$
ウ　$2^6+2^4+2^2+2^{-2}+2^{-5}+2^{-6}$　　エ　$2^6+2^4+2^2+2^{-1}+2^{-4}+2^{-5}$
（基 17 春午前問 1）

問題 1.5　正の整数の 10 進表示の桁数Ｄと２進数の桁数Ｂとの関係を表す式のうち、最も適切なものはどれか。

ア $D \approx 2\log_{10} B$　イ　$D \approx 10\log_2 B$　ウ　$D \approx B\log_2 10$　エ　$D \approx B\log_{10} 2$

（基 17 春午前問３）

問題 1.6　数値を図に示す 16 ビットの浮動小数点で表すとき、10 進数 0.25 を正規化した表現はどれか。正規化として、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表記する。

S(1 ビット)	e（4 ビット)	f (11 ビット)

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

ア 0 0001 10000000000　イ 0 1001 10000000000　ウ 0 1111 10000000000
エ 0 0001 10000000000
（基 18 春午前問 4）

問題 1.7　数多くの数値の加算を行う場合、全体値の小さいものから順番に計算すると、どの誤差を抑制できるか。
ア　アンダーフロー　イ 打切り誤差　ウ 桁落ち　エ　情報落ち
（基 17 秋午前問 4）

問題 1.8　コンピュータに使われている文字符合の説明のうち適切なものはどれか。
ア　ASCII 符号はアルファベット、数字、特殊文字および制御文字からなり、漢字に関する規定はない。
イ EUC は、文字符号の世界標準を作成しようとして考案された 16 ビット以上の符号体系

であり、漢字に関する規定はない。
ウ　Unicode は文字の１バイト目で漢字かどうかが分かるようにする目的で制定され、漢字と ASCII 符号を混在可能にした符号体系である。
エ　シフト JIS 符号は UNIX における多言語対応の一環として制定され、ISO として標準化されている。
(基 18 春午前問 69)

問題 1.9　10 進数の 0.6875 を２進数で表したものはどれか。
ア　0.1001 イ　0.1011 ウ　0.1101 エ 0.1111
（基 15 春午前問 1）

問題 1.10　数値を図に示す 32 ビットの浮動小数点で表すとき、以下の[1]-[3]を答えよ。

0　　1　　7 8　　31（ビット番号）

S(1 ビット)	e (7 ビット)	f (24 ビット)

S:仮数部の符号(0:正、１:負)
e:指数部（２を基数とし、負は、２の補数で表現）
f:仮数部（２進数、絶対値表示、ビット番号８を小数第１位とする。）

[1]以下のビット列を浮動小数点で表すと、$+(0.11101)\times 2^{(a)} = (b)$ となる。

0	0000011	1110 1000 0000 0000 0000 0000

(a)に関する解答群：ア 0　　イ 1　　ウ 2　　エ 3　　オ 4
(b)に関する解答群：ア 0.725　イ 0.74　ウ 7.25　　エ 7.4　　オ 72.5　カ 74

[2]次の浮動小数点表示の２進数を 10 進数で表現したものはどれか。

0	1111110	0011 0000 0000 0000 0000 0000

ア 3×2^{-6} イ 3×2^{-4}　ウ 3×2^{-1} エ 3×2^{0}　オ 3×2^{1}　カ 3×2^{2} キ 3×2^{122}

[3] [2]の浮動小数点を正規化したビット列はどれか。ここで、正規化とは、仮数部の最上位けた（ビット番号８）が１となるように、指数部と仮数部を調製する操作をいう。

解答群

ア

0	1111011	1100 0000 0000 0000 0000 0000

イ

0	1111100	1100 0000 0000 0000 0000 0000

ウ

0	1111101	1100 0000 0000 0000 0000 0000

エ

0	0000001	1100 0000 0000 0000 0000 0000

オ

0	0000010	1100 0000 0000 0000 0000 0000

（基 13 春午後問 1）

問題 1.11 ８ビットにより符号付き整数を表す。ここで負の数を 2 の補数により表す。
Na=0011 0111, Nb=0001 0101 について
Na-Nb の計算を Nb の２の補数を用いて加算により計算せよ。

問題 1.12 16 ビットに符号付き整数(負の数を 2 の補数により表す)について１０進数で表現できる範囲は[　　] 以上[　　]以下である。

問題 1.13 16 ビットに符号なし整数について１０進数で表現できる範囲は[　　] 以上[　　]以下である。