2019. 5.20

電子計算機工学

Ibaraki Univ. Dept of Electrical & Electronic Eng.

Keiichi MIYAJIMA

コンピューターにおける数表現

２進数

コンピュータの内部でのデータは

- 電圧の高･低
- スイッチのオン･オフ
- 磁化されている･いない

etc･･･

これらを１と０で表す

２進数で表現されている

２進数の表記

１０進数	２進数
０	０
１	１
２	１０
３	１１
４	１００
５	１０１

１０進数	２進数
６	１１０
７	１１１
８	１０００
９	１００１
１０	１０１０
・・・	・・・・

2^n のところで桁上がりが発生する

r進数

r進数とは･･･

10進数の場合

$$145.32 = 1\times 10^2 + 4\times 10^1 + 5\times 10^0 + 3\times 10^{-1} + 2\times 10^{-2}$$

2進数の場合

$$1101.01 = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2}$$

以後、r進数を区別するため

10進数: 145.32_{10}　　2進数: 1101.01_2

r: 基数

bit

ビット(bit):２進数における１桁の情報

1101.01_2 (6bit の情報)

MSb (most significant bit)

LSb (least significant bit)

16進数

2進数では桁が多くて大変なので
4桁づつひとまとめにして16進数を用いる

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

10進数	2進数	16進数
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

r進数から10進数へ

２進数から10進数

$$
\begin{aligned}
1101.01_2 &= 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} \\
&= 8 + 4 + 1 + 0.25 \\
&= 13.25_{10}
\end{aligned}
$$

16進数から10進数

$$
\begin{aligned}
\mathrm{BC.5}_{16} &= 11\times 16^1 + 12\times 16^0 + 5\times 16^{-1} \\
&= 176 + 12 + 0.3125 \quad (16^{-1} = 0.0625) \\
&= 188.3125_{10}
\end{aligned}
$$

１０進数からr進数へ

１０進数から２進数

23.75_{10} を2進数へ

23

$$
\begin{array}{rl}
2\,\underline{)\ 23} & \cdots 1 \\
2\,\underline{)\ 11} & \cdots 1 \\
2\,\underline{)\quad 5} & \cdots 1 \\
2\,\underline{)\quad 2} & \cdots 0 \\
\quad 1 &
\end{array}
$$

下から順に並べる

$23_{10} = 10111_2$

0.75

$$
\begin{array}{r}
0.75 \\
\times \quad\ \ 2 \\
\hline
1.50
\end{array}
\qquad
\begin{array}{r}
0.5 \\
\times \quad 2 \\
\hline
1.0
\end{array}
$$

順に並べる

$0.75_{10} = 0.11_2$

$23.75_{10} = 10111.11_2$

10進数からr進数へ

10進数から16進数

23.75_{10} を16進数へ

$$
\begin{array}{r}
23 \\
16)\underline{23} \cdots 7 \\
1
\end{array}
\qquad\qquad
\begin{array}{r}
0.75 \\
0.75 \\
\times \quad 16 \\
\hline
\boxed{12.}00
\end{array}
$$

下から順に並べる

順に並べる

$23_{10} = 17_{16}$ $\qquad 0.75_{10} = 0.C_{16}$

$$23.75_{10} = 17.C_{16}$$

２進数と１６進数の関係

$$23.75_{10} = 10111.11_2 = 17.C_{16}$$

１６進数は２進数を４桁づつまとめたもの

$$10111.11_2$$

$$0001\ 0111.1100_2$$ ４桁ずつ区切る

$$1_{16} \quad 7_{16} \quad C_{16}$$ それぞれの区切りごとに１６進数に直す

$$17.C_{16}$$

数の表現

コンピュータで表現できる数の範囲は、数値を表現するために利用できるビット数で決まる

⇩

一度に利用できるビット数：ワード（word）

例）　最近のPCならば64ビット

数の表現

正の整数:

１ワードが8ビットならば

$$0000\ 0000_2 = 0_{10}$$

$$\wr$$

$$1111\ 1111_2 = 255_{10}$$

では「負の数」はどうするか？

負の数の表現

絶対値表現：最初の1ビットを＋，－の符号として使用

１ワードが8ビット

$$00010010_2 = 18_{10}$$

$$10010010_2 = -18_{10}$$

利点：人間にとってわかりやすい　しかし･･･

4ビットの加算と減算を考えると

加算）
$$\begin{array}{rl} & 0101_2 = 5_{10} \\ + & 1010_2 = -2_{10} \\ \hline & 1111_2 = -7_{10} \end{array}$$

減算）
$$\begin{array}{rl} & 0101_2 = 5_{10} \\ - & 1010_2 = -2_{10} \\ \hline & 1011_2 = -3_{10} \end{array}$$

欠点：単純な回路構成では計算ができない

負の数の表現（補数表示）

補数：任意の数Nに対する補数には基数をrとすると、r-1の補数とrの補数が存在する。

例）１０進数なら９の補数と１０の補数
２進数なら１の補数と２の補数

一般に基数rの正数 N に対するr-1の補数 C_{r-1} は

$$C_{r-1} = r^n - r^{-m} - N$$

ここで、n は N の整数部の桁数、m は小数部の桁数

負の数の表現（補数の例）

例：10進数326の9の補数は

$$C_{r-1} = r^{n} - r^{-m} - N$$

326_{10}の9の補数$= 10^{3} - 10^{-0} - 326 = 999 - 326 = \boxed{673_{10}}$

つまり、326の9の補数とは、整数部3桁で表現できる最大の数999になるために326にいくつ加えればよいかに相当する。

例２：10進数0.36の9の補数は

0.36_{10}の9の補数$= 10^{0} - 10^{-2} - 0.36 = 0.99 - 0.36 = \boxed{0.63_{10}}$

負の数の表現（補数の例）

例：2進数101の1の補数は

$$C_{r-1} = r^n - r^{-m} - N$$

101_2 の1の補数$= 2^3 - 2^{-0} - 101_2 = 111_2 - 101_2 = \boxed{010_2}$

つまり、101の1の補数とは、整数部3桁で表現できる最大の数111になるために101にいくつ加えればよいかに相当する。

しかし、よく見れば1と0を反転させただけ

負の数の表現（rの補数）

一般に基数rの正数 N に対するr-1の補数 C_{r-1} は

$$C_{r-1} = r^n - N$$

ここで、n は N の整数部の桁数、m は小数部の桁数

負の数の表現（rの補数の例）

例：10進数326の10の補数は

$$C_{r-1} = r^n - N$$

$$326_{10} \text{の10の補数} = 10^3 - 326 = 1000 - 326 = \boxed{674_{10}}$$

つまり、326の10の補数とは、9の補数に1加えたもの

例：10進数0.36の10の補数は

$$0.36_{10} \text{の10の補数} = 10^0 - 0.36 = 1 - 0.36 = \boxed{0.64_{10}}$$

負の数の表現（rの補数の例）

$$C_{r-1} = r^n - N$$

例:２進数１０１の２の補数は

101_2 の２の補数 $= 2^3 - 101_2 = 1000_2 - 101_2 = \boxed{011_2}$

つまり、１０１の２の補数とは、１の補数に１加えたもの

例:２進数０.１０１の２の補数は

0.101_2の２の補数 $= 2^0 - 0.101_2 = 1.000_2 - 0.101 = \boxed{0.011_2}$

なぜ、補数表現なるものを用いるのか？

補数による演算

補数を用いることにより

減算も加算で表現可能

１０進数による例：$784_{10} - 762_{10} = 22_{10}$

減数（負の数）762の１０の補数は　２３８

$$
\begin{array}{r}
784_{10} \\
+\ \ 238_{10} \\
\hline
\boxed{1}022_{10}
\end{array}
$$

桁上げ分を無視すると答えの ２２ が出てくる

補数による演算

１０進数による例： $784_{10} - 813_{10} \qquad = -29_{10}$

減数（負の数）813の１０の補数は　１８７

$$
\begin{array}{r}
784_{10} \\
+\ 187_{10} \\
\hline
\boxed{}\,971_{10}
\end{array}
$$

桁上げが起こらないならば、負の値なので、９７１の１０の補数を求める　→　-29_{10}

補数による演算（2進数の場合）

条件：
- 符号を含めて1ワード8ビットとする
- 加算においてオーバーフローは起こらないものとする

数値が8ビットで表現できる範囲を超えてしまうこと

この場合：＋127～－128

例）

$$0010\ 0101_2 = 37_{10}$$

$$1001\ 1101_2 = -29_{10}$$

補数による演算（2進数の場合）

（正の数）＋（負の数）の例：$34_{10} - 22_{10} = 12_{10}$

$34_{10} = 0010\ \ 0010_2$

$22_{10} = 0001\ \ 0110_2$

22は負の数なので最初の1ビットを1にして、残りの7ビットで2の補数をつくる

$-22_{10} = 1110\ \ 1010_2$

$34_{10} = 0010\ \ 0010_2$

加算

$$
\begin{array}{rr}
 & 0010\ \ 0010_2 \\
+ & 1110\ \ 1010_2 \\
\hline
\boxed{1} & 0000\ \ 1100_2
\end{array}
= 12_{10}
$$

桁上げ分を無視すると答えの 12 が出てくる

補数による演算（2進数の場合）

（正の数）＋（負の数）の例： $34_{10} - 48_{10} = -14_{10}$

$34_{10} = 0010\ \ 0010_2$

$48_{10} = 0011\ \ 0000_2$

48は負の数なので最初の1ビットを1にして、残りの7ビットで2の補数をつくる

$-48_{10} = 1101\ \ 0000_2$

$34_{10} = 0010\ \ 0010_2$

加算

$$\begin{array}{r} 0010\ \ 0010_2 \\ +\ \ 1101\ \ 0000_2 \\ \hline \boxed{1111\ \ 0010_2} \end{array} = -14_{10}$$

最初の1ビットが1なので負の数、残りの7ビットの2の補数をとると答えの −14 が出てくる

補数による演算（2進数の場合）

（負の数）＋（負の数）の例: $-34_{10} - 13_{10} = -47_{10}$

$34_{10} = 0010\ 0010_2$　$13_{10} = 0000\ 1101_2$

34と13は負の数なので最初の1ビットを1にして、残りの7ビットで2の補数をつくる

$-34_{10} = 1101\ 1110_2$
$-13_{10} = 1111\ 0011_2$

加算

$$\begin{array}{r} 1101\ 1110_2 \\ +\ 1111\ 0011_2 \\ \hline \boxed{1}\ 1101\ 0001_2 \end{array} = -47_{10}$$

桁上げを無視する。8ビットのうち最初の1ビットが1なので負の数、残りの7ビットの2の補数をとると答えの —47が出てくる

補数を用いる利点

例にあげてきたように単純なルールに基づく加算のみで、減算も計算できる。

↓

回路が簡単になる

↓

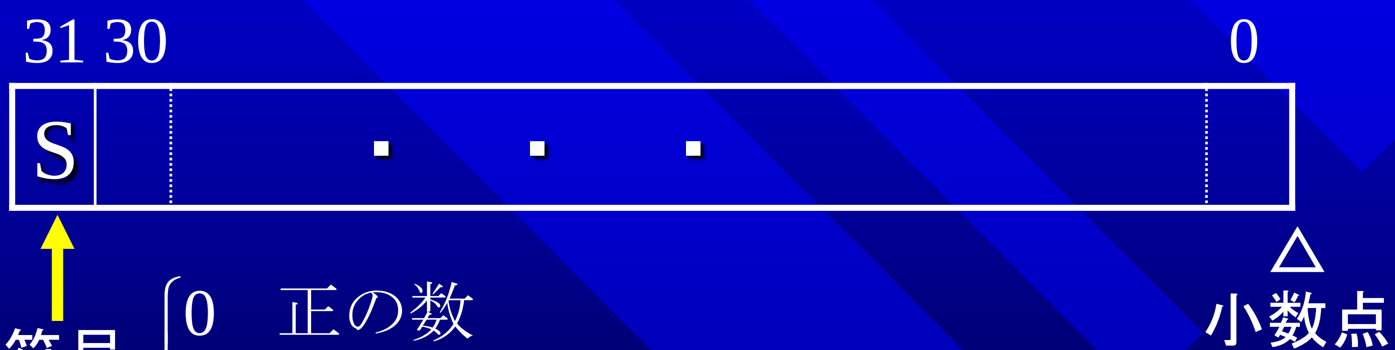
より高集積、より高速な回路

数値データ

コンピュータ内部ではデータは2進数で表現される

固定小数点表示:

小数点を特定の位置に固定し、数値によって動かさない。



符号
0 正の数
1 負の数（2の補数で表される）

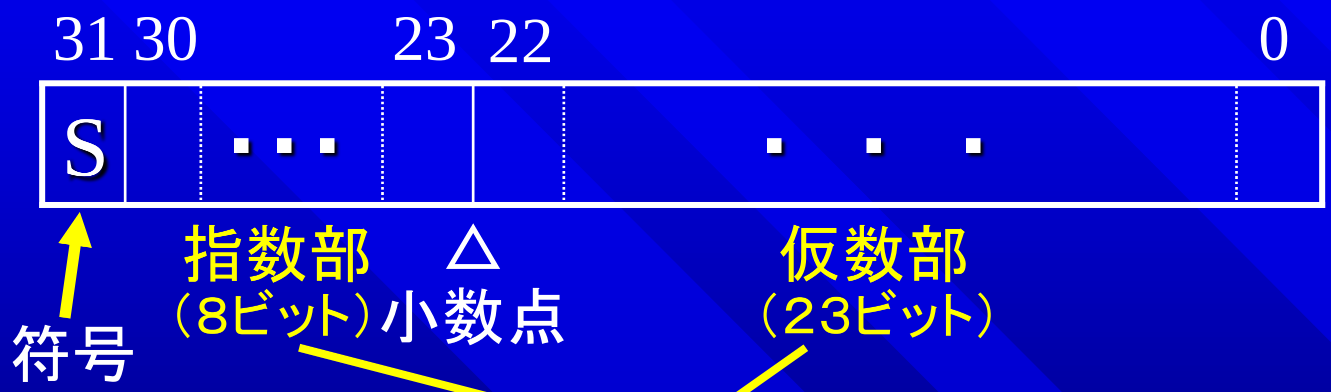
整数型のデータを表すのに用いられる。

この場合、10進数で -2^{31} から $2^{31}-1$ までが表現できる

数値データ

浮動小数点表示:

指数表示の概念を用いる(以下はIEEE方式)



$$N = a \times 2^{n}$$

極めて大きな数値や、小さな数値を表すのに用いられる。

なお、a は2進数で次式を満足するように正規化される

$$2^{-1} \leq |a| < 1$$

数値データ

２進化１０進コード：

１０進数の各桁を4ビットの２進数に変換して表す。

例）243_{10}と-243_{10}をパック形式で表すと

	2	4	3	正
243_{10}	0010	0100	0011	1100

	2	4	3	負
-243_{10}	0010	0100	0011	1101

論理データと文字データ

論理データ:

ビットごとに論理演算の対象となる。ビットの値が1の時は真、0の時は偽を表す。

通常は8ビットや16ビットに複数個の論理値をパックし、1つのデータとして取り扱う。

文字データ:

英数字、カナ文字、特殊文字などコンピュータのコード体系に定められている文字を表す。

（WindowsではS-JISコードで日本語が表される）

本日のまとめ

コンピュター上でのデータ表現

1. ２進数と16進数
2. 負の数の表現(補数)
3. 補数による演算
4. 数値データの表現

   固定小数点表示、浮動小数点表示

本日の課題１

１．$131_8 - 45_{16} = \left(\qquad\right)_{10}$

( )に入る数字はいくらか？

２．ある自然数Xを2進数で表現すると、1と0が交互に並んだ2n桁の2進数1010･･･10となった。このとき、Xに関して以下の式が成立する。その理由を述べなさい。 （基本情報 改題）

$$X + \frac{X}{2} = 2^{2n} - 1$$

３． $81_{10} - 54_{10}$ の計算を符号を含む8桁の2進数に直し、2の補数を用いた加算によって計算し、求めなさい。
（答えは符号を含む8桁の2進法によって書くこと）

本日の課題２

４．次の24ビットの浮動小数点で表現できる最大値を表すビット列を、16進数として表したものはどうなるか。理由も付けて回答せよ。なお、ここでこの形式で表現される値は

$(-1)^S \times 16^{E-64} \times 0.M$ である。

（基本情報）

0	1 〜 7	8 〜 23
S	E	M