計算機方式論

第6章 データ形式

− 負の整数の加算と浮動小数点数

負の整数の加算

- ◆ 符号付き2進数Xの表す負の整数は、符号と絶対値、1の補数、2の補数の3通りあり、それぞれ、表す値(10進数値)を$(X)_{SM}$、$(X)_{1C}$、$(X)_{2C}$とした。
- ◆ 一方、符号無し2進数X,Yは、2進数加算($+_2$と表記)をしたとき、$(X+_2Y)$の値は、それぞれの値$(X)_2$、$(Y)_2$の(10進)加算(+と表記)をした値$(X)_2+(Y)_2$となる。すなわち、$(X+_2Y)_2 = (X)_2+(Y)_2$

```
X      0101      5 (X)2
Y      1001      9 (Y)2
   +2 ______   + _____
       1110     14
```

- ◆ 符号付き2進数X,Yの加算は、どうなるのか？

負の整数の加算 ① 符号と絶対値表現の下での加算

- ◆ 符号と絶対値表現下の整数加算は、2つの2進数X,Yの表す値の10進数加算値を求めるとき、XとYの符号ビットを見て、それに応じて絶対値の和や差を求める。
- ◆ 手順が煩雑な上、加算回路以外にも減算回路を必要とする。

〔注意〕通常の2進数加算、減算を$+_2$、$-_2$と書くとする。
3ビットの2進数加減算なら、$101+_2010=111$、$101-_2010=011$。
一方、10進数−5と2の10進数加算は+と書き、$(-5)+2 = -3$。

```
X  0 1 0 1   +5 (X)SB     X  1 1 0 1   −5 (X)SB     X  1 1 0 1   −5 (X)SB
Y  0 0 1 0   +2 (Y)SB     Y  1 0 1 0   −2 (Y)SB     Y  0 0 1 0   +2 (Y)SB
 +2 ______ + ______        +2 ______ + ______        −2 ______ + ______
     1 1 1                     1 1 1                     0 1 1
   0 1 1 1   +7              1 1 1 1   −7              1 0 1 1   −3
```

同符号の場合 ⇒ 各絶対値を2進数加算$+_2$して同じ符号を付ける

異符号の場合 ⇒ 各絶対値を2進数減算$-_2$して大きい方の符号

負の整数の加算 ②1の補数表現の下での加算

- ◆ ここでは、2進数XとYとの循環桁上げを伴う2進数加算を$X+_2Y$と書くことにする(循環桁上げ(end around carry)とは、符号からの桁上が生じたとき、和にさらに1を加えること)。
XとYの1の補数による符号を考慮した(10進数)値を$(X)_{1C}$と$(Y)_{1C}$とする。このとき、$X+_2Y$の符号を考慮した値は$(X)_{1C}+(Y)_{1C}$になる。すなわち、$(X+_2Y)_{1C} = (X)_{1C}+(Y)_{1C}$。

```
    X    0 1 0 1      +5 (X)1C             X     1 1 0 1     −2 (X)1C
    Y    0 0 1 0      +2 (Y)1C             Y     1 1 0 0     −3 (Y)1C
 +2  __________   + ______             +2  ___________   + ________
X+2Y     0 1 1 1      +7 (X)1C+(Y)1C            1 1 0 0 1
                                                ←←符号ビットへの/からの桁上
                                                 └−−−→+1  循環桁上
                                              _______________________
                                       X+2Y       1 0 1 0    −5 (X)1C+(Y)1C
```

②XとYの符号が同じとき

- Xが正の場合、$(X)_{1C}=(X)_2$、負の場合、Xの1の補数の値を負としたものより、$(X)_{1C}=-\{2^n-1-(X)_2\}$、すなわち、$(X)_2=2^n-1+(X)_{1C}$。
- XとYが正の場合、$(X+_2Y)_2=(X)_2+(Y)_2=(X)_{1C}+(Y)_{1C}$。
  ただし、$X+_2Y$の符号が1になるときはオーバーフロー。
- XとYが負の場合、$(X+_2Y)_2=(X)_2+(Y)_2=2^n-1+(X)_{1C}+2^n-1+(Y)_{1C}$
  循環桁上は、上記の値から2^n-1を引くことを表すので、結果値$2^n-1+(X)_{1C}+(Y)_{1C}$は、その1の補数の値を負とした$(X)_{1C}+(Y)_{1C}$を表す。
  結果の符号が0になるときはオーバーフロー。

X	x_{n-1}	x_{n-2}	・・・	x_0
$+_2$ Y	y_{n-1}	y_{n-2}	・・・	y_0
1	z_{n-1}	z_{n-2}	・・・	z_0
$X+_2Y$	z_{n-1}'	z_{n-2}'	・・・	z_0'

← ←符号への/からの桁上

→+1 循環桁上

循環桁上を伴う2進数の和

- XとYの符号が異なるときも同様に考える。

5

②正のオーバーフローのとき

- 2進数加算$X+_2Y$の際、
  「符号ビットへの桁上げ」と
  「符号ビットからの桁上げ」の
  片方だけが生じるときがオーバーフローとなる。
- X,Y共に正のとき、$0\leqq(X)_{1C},(Y)_{1C}\leqq 2^{n-1}-1$、
  $(X)_2=(X)_{1C}$、$(Y)_2=(Y)_{1C}$から、求める和は

$$0 \leqq (X+_2Y)_2=(X)_2+(Y)_2=(X)_{1C}+(Y)_{1C} \leqq 2^{n-1}-1$$

$(000\cdots00)_2$　　$(001\cdots11)_2$

符号ビットへの桁上げが生じない

オーバーフローは、和がnビットに納まらず、符号ビットが1になる。

$$2^{n-1} \leqq (X+_2Y)_2=(X)_{1C}+(Y)_{1C} \leqq 2^n-2$$

$(010\cdots00)_2$　　$(011\cdots10)_2$

← 符号ビットへの桁上げが生じる ←

符号ビットからの桁上げは生じない

6

②XとYとが異符号のとき

- Xが正,Yが負として、$(X)_2=(X)_{1C}$、$(Y)_2=(2^n-1)+(Y)_{1C}$より、
  $(X+_2Y)_2=(X)_2+(Y)_2=(2^n-1)+(X)_{1C}+(Y)_{1C}$
  オーバーフローは生じない！

$0\leqq(X)_{1C}\leqq 2^{n-1}-1$
$-(2^{n-1}-1)\leqq(Y)_{1C}<0$

- $(X)_{1C}+(Y)_{1C}>0$ のとき
  $2^n\leqq(X+_2Y)_2=(2^n-1)+(X)_{1C}+(Y)_{1C}\leqq 2^n+2^{n-1}-2$ なので、
  符号ビットからの桁上げと符号ビットへの桁上げが共に生じる。
  $(X+_2Y)_2$で(2^n-1)を循環桁上げで無視した値$(X)_{1C}+(Y)_{1C}$が、求める和。

$2^n\leqq(X+_2Y)_2\leqq 2^n+2^{n-1}-2$
$(100\cdots00)_2$　　$(101\cdots10)_2$
←←　　←←
→+1　　→+1
和$(00\cdots01)_2$　　$(01\cdots11)_2$

- $(X)_{1C}+(Y)_{1C}\leqq 0$ のとき
  $-(2^{n-1}-1)\leqq(X)_{1C}+(Y)_{1C}\leqq 0$ なので、
  $2^{n-1}\leqq(X+_2Y)_2=(2^n-1)+(X)_{1C}+(Y)_{1C}\leqq 2^n-1$。
  $(X+_2Y)_2$は符号ビットが1で、
  値$-\{(X)_{1C}+(Y)_{1C}\}$の1の補数より、
  値$(X)_{1C}+(Y)_{1C}$を表す。

$2^{n-1}\leqq(X+_2Y)_2\leqq 2^n-1$
$(010\cdots00)_2$　　$(011\cdots11)_2$

符号ビットからの桁上げと符号ビットへの桁上げは共に生じない。

7

負数の1の補数表現の下での加算例

X	0 1 0 1	+5	$(X)_{1C}$		0 1 0 1	+5
$+_2$ Y	0 0 1 0	+2	$(Y)_{1C}$	$+_2$	0 1 0 0	+4
$X+_2Y$	0 1 1 1	+7	$(X)_{1C}+(Y)_{1C}$		1 0 0 1	オーバーフロー

2進数加算　桁上げ無し　10進数加算　← 符号ビットへの桁上げ

X	1 1 0 1	−2	$(X)_{1C}$		1 1 0 1	−2
$+_2$ Y	1 1 0 0	−3	$(Y)_{1C}$	$+_2$	1 0 0 1	−6
	1 1 0 0 1				1 0 1 1 0	
$X+_2Y$	1 0 1 0	−5	$(X)_{1C}+(Y)_{1C}$		0 1 1 1	オーバーフロー

$(2^n-1)+\{(2^n-1)+(X)_{1C}+(Y)_{1C}\}$

←←符号ビットへの/からの桁上げ　← 符号ビットからの桁上げ

→+1 循環桁上げ　→+1 循環桁上げ

$(2^n-1)+(X)_{1C}+(Y)_{1C}$

8

```
   X    1 1 0 1   -2   (X)1C              0 1 0 1   +5
+2 Y    1 0 1 0   -5   (Y)1C        +2    1 1 0 0   -3
   └───────────────────            └──────────────────
       1 0 1 1 1                          1 0 0 0 1
       ←符号ビットからの桁上げ              ←←
       └──→+1  循環桁上げ                  └──→+1  循環桁上げ
  └────────────────────           └────────────────────
X+2Y   1 0 0 0   -7   (X)1C+(Y)1C          0 0 1 0   +2
       ←符号ビットへの桁上げ

   X    1 0 1 0   -5   (X)1C              0 1 0 1   +5
+2 Y    0 0 1 1   +3   (Y)1C        +2    1 1 1 1   -0 負のゼロ
   └───────────────────            └──────────────────
X+2Y   1 1 0 1   -2   (X)1C+(Y)1C        1 0 1 0 0
       桁上げ無し                          ←←
                                          └──→ +1  循環桁上げ
                                    └────────────────────
                                           0 1 0 1   +5
```


負の整数の加算
③2の補数表現の下での加算

- ここでは、2進数XとYとの符号ビットからの桁上げを無視した2進数加算をX$+_2$Yと書くことにする。
XとYの2の補数による符号を考慮した(10進数)値を$(X)_{2C}$と$(Y)_{2C}$とする。このとき、X$+_2$Yの符号を考慮した値は$(X)_{2C}+(Y)_{2C}$になる。すなわち、$(X+_2Y)_{2C} = (X)_{2C}+(Y)_{2C}$。

```
    X   0 1 0 1   +5  (X)2C              X   1 0 1 1   -5  (X)2C
+2  Y   0 0 1 0   +2  (Y)2C         +2   Y   1 1 0 1   -3  (Y)2C
   └───────────────────                 └───────────────────
X+2Y    0 1 1 1   +7  (X)2C+(Y)2C           1 1 0 0 0
                                            ←←符号ビットへの/からの桁上
                                        無視 1 0 0 0  -8
```


③XとYの符号が同じとき

- Xが正の場合、$(X)_{2C}=(X)_2$、負の場合、Xの2の補数の値を負としたものより、$(X)_{2C}=-\{2^n-(X)_2\}$、すなわち、$(X)_2=2^n+(X)_{2C}$。
- XとYが正の場合、$(X+_2Y)_2=(X)_2+(Y)_2=(X)_{2C}+(Y)_{2C}$。
ただし、X$+_2$Yの符号が1になるときはオーバーフロー。
- XとYが負の場合、$(X+_2Y)_2=(X)_2+(Y)_2=2^n+(X)_{2C}+2^n+(Y)_{2C}$
符号ビットからの桁上は、上記の値の2^nを表すので、それを無視した結果値$2^n+(X)_{2C}+(Y)_{2C}$は、その2の補数の値を負とした$(X)_{2C}+(Y)_{2C}$を表す。
結果の符号が0になるときはオーバーフロー。
- XとYの符号が異なるときも同様に考える。

X	x_{n-1}	x_{n-2}	・・・	x_0
$+_2$ Y	y_{n-1}	y_{n-2}	・・・	y_0
X$+_2$Y	z_{n-1}	z_{n-2}	・・・	z_0

← ← 符号への/からの桁上

③負のオーバーフローのとき

- 2進数加算X$+_2$Yの際、
「符号ビットへの桁上げ」と「符号ビットからの桁上げ」の片方だけが生じるときがオーバーフローとなる。
- X,Y共に負のとき、$(X)_2=2^n+(X)_{2C}$、$(Y)_2=2^n+(Y)_{2C}$。
$(X+_2Y)_2=(X)_2+(Y)_2=2^n+\{2^n+(X)_{2C}+(Y)_{2C}\}$、$2^n$は、符号ビットからの桁上げで、この項は無視するので、残りの項 $\{2^n+(X)_{2C}+(Y)_{2C}\}$ を扱う。
$-2^{n-1} \leqq (X)_{2C},(Y)_{2C} < 0$ なので、$-2^n \leqq (X)_{2C}+(Y)_{2C} < 0$ のときが、求める和。無視した項2^nが符号からの桁上げを表す。符号ビットからの桁上げ(項2^n)が必ず生じる!

$$2^{n-1} \leqq 2^n+(X)_{2C}+(Y)_{2C} < 2^n$$

$(010\cdots00)_2$ … $(011\cdots11)_2$
← 符号への桁上げが生じている ←

オーバーフローは、和がnビットに納まらず、符号ビットが0になる$-2^n \leqq (X)_{2C}+(Y)_{2C} < -2^{n-1}$のとき生じる。このとき、

$$0 \leqq 2^n+(X)_{2C}+(Y)_{2C} < 2^{n-1}$$

$(000\cdots00)_2$ … $(001\cdots11)_2$
符号への桁上げが生じない

負数の2の補数表現の下での加算例

```
     X  0101  +5   (X)2C          0101  +5
 +2  Y  0010  +2   (Y)2C      +2  0100  +4
    └──────────────────       └────────────────
 X+2Y   0111  +7   (X)2C+(Y)2C    1001 オーバーフロー
 2進数加算 桁上げ無し  10進数加算   ← 符号ビットへの桁上げ
     X  1011  -5   (X)2C          1011  -5
 +2  Y  1101  -3   (Y)2C      +2  1100  -4
    └──────────────────       └────────────────
       11000                     10111
       ←←符号ビットへの/からの桁上げ  ←符号ビットからの桁上げ
    無視 1000  -8  (X)2C+(Y)2C  無視 0111 オーバーフロー
```

13

負数の2の補数表現の下での加算例

```
     X  1011  -5  (X)2C          0101  +5
 +2  Y  0011  +3  (Y)2C      +2  1101  -3
    └──────────────────      └──────────────────
        1110  -2  (X)2C+(Y)2C   10010  +2
    桁上げ無し                   ←←符号ビットへの/からの桁上げ
                               無視
```

14

3つの負数表現下の加減算比較

◆加減算を行うとき、

- ◆符号と絶対値表現では、加算回路と減算回路が必要になる。
- ◆補数表現では、減算は補数の加算で行える。
- ◆1の補数表現での減算は、補数化の後の2進数加算で循環桁上げが必要になる。
- ◎2の補数表現での減算は、補数化と2進数加算だけで行える☞次図

15

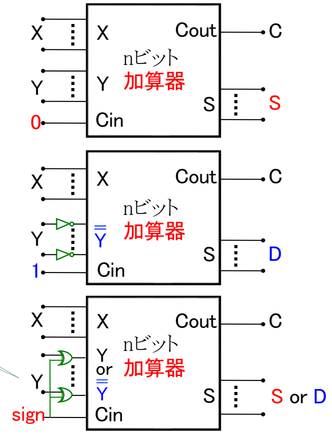
加算器・減算器

nビット加算器 S=X+Y

負数に2の補数表現を使う:
nビット減算器 $D=X-Y=X+\overline{Y}$
補数を加えることで差が求まる

Yi	⊕ sign	=
0	0	0 Yi
1	0	1
0	1	1 $\overline{Y_i}$
1	1	0

負数に2の補数を使う:
nビット加減算器
信号signが、0のとき加算器
1のとき減算器



16

小数点の表現

◆固定小数点方式

◆浮動小数点方式

17

固定小数点方式

◆小数点の位置が予め固定。

①最上位桁(Most Significant Digit)と次の桁の間
小数形計算機

符号	２進数表示

・← 小数点

②最下位桁(Least Significant Digit)の次
整数形計算機

符号	２進数表示

小数点→・

18

浮動小数点方式

◆仮数部(mantissa)、指数部(exponent)、仮数部の符号とから成る。

仮数M、指数E、基数R、仮数の符号Sとしたとき、数値 $(-1)^S M \cdot R^E$ を表す。
(R=2,4,8,10,16)

S	指数部(E)	仮数部(M)

19

浮動小数点方式-正規化

◆正規化(normalization)

$0.0011011 \times 2^5 \Rightarrow \underline{1.1011} \times 2^2$　基数は2

仮数部を標準の形1.xxxxxにすること！

0	0 0 1 0	1 1 0 1 1 0

・←小数点

$-1101100.0 \times 2^{-5} \Rightarrow -\underline{1.1011} \times 2^1$

1	0 0 0 1	1 1 0 1 1 0

20

浮動小数点方式-けち表現と隠しビット

◆けち表現と隠しビット

(economized form and implicit MSB)

正規化済み仮数部:

1. 1010 ⇒ ~~1~~ 1011

正規化済み仮数部の**先頭ビットは常に** 1 なので**省略**

⇒ 仮数部が**1ビット得する！**

0	0 0 1 0	1 1 0 1 1 0

⇩ ・←小数点

0	0 0 1 0	1 0 1 1 0 0

・←小数点

浮動小数点方式-バイアス表現

◆**バイアス表現**(bias)

指数部が負の数

1.1011×2^{-3}

を表したい！

0	0 1 0 0	1 1 0 1 1 0

この表現の指数部4ビットで0～15までの正数を表すが、

負の数を表すため、−7のげた（**バイアス値**7）を履かせて、

$1.1011 \times 2^{4-7} = 1.1011 \times 2^{-3}$

なる数を表すこととする。

これにより、**仮数部** $\times 2^{-7}$～**仮数部** $\times 2^{8}$なる数を表せる。

浮動小数点方式−IEEE規格

IEEE規格

単精度/倍精度、

仮数の負数は**符号と絶対値**表現、

指数(8/11bits)と**仮数**(23/52bits)ともに**2進数**（**基数2**）、

指数は**バイアス表現**(127/1023)、

隠しビット有り

IEEE754の単精度の正規化数

S	指数部(E)	仮数部(M)
	8ビット	23ビット

0<E<255のとき

$(-1)^S\ 2^{E-127} \times (1.M)$:単精度

バイアス値　隠しビット

浮動小数点方式−IBMアーキテクチャ

IBMアーキテクチャ

単精度/倍精度、

仮数の負数は**符号と絶対値**表現、

指数(7bits)と**仮数**(24/56bits)ともに**16進数**（**基数16**）、

指数は**バイアス表現**(64)、

仮数は**1未満**

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

S	指数部(E)	仮数部(M)
	7ビット	24ビット