基本情報技術Ｉ

2. 符号，算術演算

菊池浩明

講義目標

- 教科書
  - 1-1, 3. 整数の表現, 4. 小数, 5. 算術演算, 6. シフト演算, 7. 誤差
- N進数の負の数を表現できる
- N進数の加減算と乗算とシフト演算ができる
- 誤差の違いが分かる.

宿題（次回小テスト範囲）

- 1章練習問題 (p.56)
  - 7 (パリティ), 8（ビットパターン）
  - 4 (シフト演算), 5, 6 （ビット演算）, 9. 誤差

3. 整数の表現(負数)

- nビット整数 (integer)
  - 符号なし整数 $[0, 2^n - 1]$ (unsigned)
  - 符号付き整数 $[-2^{n-1}, 2^{n-1}-1]$ (signed)
  - 参考) Java

種別	型	n	値域
符号なし	char	2	[0, 65535]
符号整数	byte	1	[-128 , 127]
	short	2	[-32768, 32767]
	int	4	$[-2^{31}, 2^{31}-1]$
	Long	8	$[2^{63}, 2^{63}-1]$

整数の減算

- 3ビットの加算のみで減算ができる
  - 例） 3 + 6 =

$$\begin{array}{r} 3 \\ +\ \ 6 \\ \hline 9 \end{array} \qquad \begin{array}{r} 011 \\ +\ \ 110 \\ \hline 1001 \end{array} \qquad \begin{array}{r} 3 \\ -\ \ 2 \\ \hline 1 \end{array} \qquad \boxed{110=\underline{-}2}$$

表1.2 負数の表現

- （3ビット）

MSB		LSB
1	1	0

10進数	2進数	符号付整数	符号
7	111	−1	−
6	110	−2	
5	101	−3	
4	100	−4	
3	011	3	+
2	010	2	
1	001	1	
0	000	0	

10進数	2進数	符号付整数	符号	
3	011	3	+	最大
2	010	2		
1	001	1		
0	000	0		
7	111	−1	−	
6	110	−2		
5	101	−3		
4	100	−4		最小

補数の求め方

- 例）n = 8bit, b = 10
  - (1) $-b = 2^n - b$
    $= 2^8 - 10 = 256 - 10 = 246$
  - (2) $b = 10_{(10)} = 00001010_{(2)}$

    1の補数（ビット反転) 11110101

    $+1$

    2の補数 (-b) $= 11110110 = 246$

    (1), (2)どちらでやっても同じ.

正負の変換

(1) $256 - b$

(2) $\sim b^{(反転)} + 1$

$\boxed{b = 10}$ ⇄ $\boxed{-b = 246}$

(1) $256 - (-b)$

(2) $\sim(-b)^{(反転)} + 1$

例1

- N = 8の時，次の負数を求めよ.
  - -1 = (16)
- N=16の時,
  - -1 = (16)
- N=8の時,
  - 最小値 = (16)
  - 最大値 = (16)

例2

- 2の補数で表現された負数$11010110_{(2)}$の「絶対値」を求めよ.

  $11010110_{(2)}$の1の補数 =

  1の補数+1 =

- $-42_{(10)}$を8ビットの2の補数表現して，10進数で表せ.

4. 小数点数の表現

- 固定小数点表示
  - 小数点の位置を固定した表現方法
  - 符号bit + 整数部 + 小数部

- 利点
  - 高速, 実装が容易
  - 精度が悪い. 丸め誤差

例3

- $X = -6.625_{(10)}$ を，4ビット整数部，4ビット小数部の固定小数点で表現する．
  - b = 6.625 = $110.101_{(2)}$
  - b * 2^4 = 0110 1010
  - $-(b*2^4)$ = 10010101 + 1 = 1001 0110
  - $-(b*2^4)/2^4$ = 1001.0110

  - 小数部を整数化して考えるとよい．（実際は何もしない）

例4 (固定小数点数）

- 整数部4ビット，小数部4ビットの固定小数点数 B=1010.0101$_{(2)}$を10進数で表せ.
  - ❑B*2^4の1の補数　　=
  - ❑　　　　2の補数　　=
  - ❑2の補数/2^4　　=　　　　　　　　(2)
  - 　　　　　　　　　=　　　　　　　　(10)
  - ❑注) 小数部 .0101がそのまま.5$_{(10)}$にならない.

浮動小数点 (floating-point format)

- 定義
  - 小数点の位置を固定しない小数の表現.
  - 符号S＋仮数M＋指数E
  - 例）－ 0.125 x 2^6
    S = 1(負),
    M = .125 = $.001_{(2)}$ = 1/8
    E = 6 = $110_{(2)}$

正規化

- 表現方法
  - .001 x 2^7
  - .01 x 2^6
  - .1 x 2^5 (正規化: 仮数の最初のビット1)
- 有効数値
  - 数値の持つ有効な桁数
  - 例）賛成は, 33.3333 % であった.（実際は, 3人中の一人. 有効数値は1ケタ）
  - 正規化は有効桁数を最大化する.

例5（正規化）

- X = 0.375を，正規化して，浮動小数点数で表せ．

S (符号）	E (指数）	M (仮数）
1 bit	4 bit	11 bit

  - ❑ $X = 0.011_{(2)}$より，
    S = 0, E = 0, M = 01100000000
  - ❑ 正規化: $X = 0.11 \times 2^{-1}$
    S =　, E =　, M =

参考

- 問
  - 0.0123 [m]を表すのはどれが適切か？
  - a) 0.123 $\times 10^{-1}$,
    b) 1.23 $\times 10^{-2}$
    c) 12.3 $\times 10^{-3}$
    d) 123 $\times 10^{-4}$

参考) IEEE 754

IEEE 754 の浮動小数点形式

http://pc.nikkeibp.co.jp/pc21/special/gosa/eg4.shtml

・基数は2。基数はデータには含めない。
・仮数は1以上2未満にそろえる。これを正規化と言う。
・0は指数と仮数の全ビットを0にする。
・仮数と指数は2進数で表現する。
・符号は正を0、負を1で表す。

● 単精度 浮動小数点形式(32ビット)

	指数部 (8ビット)	仮数部 (23ビット)

↑
符号部(1ビット)

指数部 … 実際の指数に127を足す。
仮数部 … 整数部分の1を省略する。

● 倍精度 浮動小数点形式(64ビット)

	指数部 (11ビット)	仮数部 (52ビット)

↑
符号部(1ビット)

指数部 … 実際の指数に1023を足す。
仮数部 … 整数部分の1を省略する。

5. 算術演算

- 加算
  - けた上がり (キャリー carry)

$$
\begin{array}{r}
18_{(10)} \\
+\ 25_{(10)} \\ \hline
43_{(10)}
\end{array}
\qquad
\begin{array}{r}
1101_{(2)} \\
+\ 0101_{(2)} \\ \hline
10010_{(2)}
\end{array}
$$

  - オーバーフロー:　桁上がりの結果，数値範囲を超えてしまうエラー

減算

- ボロー(borrow)

$$-0101 = 1010+1 = 1011$$

$$\begin{array}{r} 55_{(10)} \\ -\ 18_{(10)} \\ \hline 37_{(10)} \end{array} \qquad \begin{array}{r} 1110_{(2)} \\ -\ 0101_{(2)} \\ \hline 1001_{(2)} \end{array} \qquad \begin{array}{r} 1110_{(2)} \\ +\ \ 1011_{(2)} \\ \hline 11001_{(2)} \end{array}$$

- アンダーフロー：演算の結果，数値表現範囲の下限を下回ってしまう.

N進数の減算

- ボロー
  - Nが下の桁に降りてくる.

$$\begin{array}{r} 131_{(8)} \\ -\ 45_{(8)} \\ \hline 64_{(8)} \end{array} \qquad \begin{array}{r} 131_{(7)} \\ -\ 45_{(7)} \\ \hline 53_{(7)} \end{array} \qquad \begin{array}{r} 131_{(9)} \\ -\ 45_{(9)} \\ \hline {}_{(9)} \end{array}$$

6. 乗除算(シフト演算)

- 10進数の乗算

$$
\begin{array}{rl}
31_{(10)} & \\
\times\ \underline{12}_{(10)} & \\
62_{(10)} & =31*2 \\
\underline{+31}_{(10)} & =31*10 \\
372_{(10)} &
\end{array}
$$

- 2進数の乗算

$$
\begin{array}{rl}
1101_{(2)} & \\
\times\ \underline{011}_{(2)} & \\
1101_{(2)} & =1101*1 \\
\underline{+1101}_{(2)} & =1101*10 \\
100111_{(2)} &
\end{array}
$$

$$
\begin{aligned}
&=b_4b_3b_2b_1*1 \\
&+b_4b_3b_2b_1*10
\end{aligned}
$$

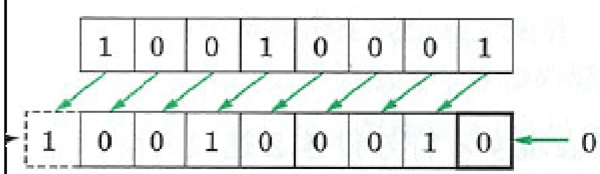
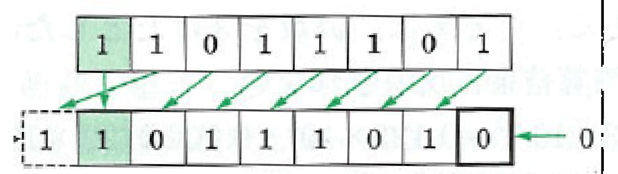
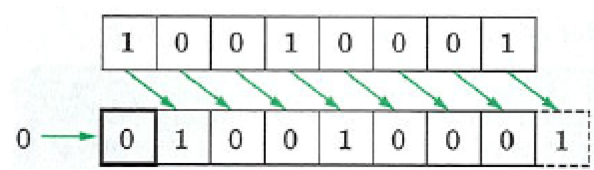
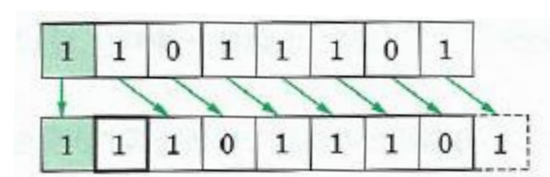
例5 (シフト演算)

- 10進数
  - $123_{(10)} \times 10_{(10)} = 1230$
  - $123_{(10)} \times 100_{(10)} = 12300$
  - $123_{(10)} / 10_{(10)} = 12.3$
- 2進数
  - $1101_{(2)} \times 2_{(10)} =$
  - $1101_{(2)} \times 4_{(10)} =$
  - $1101_{(2)} / 2_{(10)} =$

負数の演算

- $-35_{(10)}$ x 2 = -70　(8ビット)
  - -35 =(-100011)+1 = 11011100+1
    = 11011101　　　　　　　　　　　　(2の補数)
  - 11011101　x 10 = 1 10111010　（左シフト）
  - $-10111010 = 01000101 + 1 = 1000110 = +70_{(10)}$
- $-35_{(10)}$ / 2 = -17.5
  - 11011101 / 10 = 11101110　　　　(右シフト)
  - $-11101110 = 00010001 + 1 = 10010 = +18_{(10)}$

シフト演算

	論理シフト	算術シフト （符号付)
左シフト (2倍)		
java	byte a = -35; DD a << 1 -70	a << 1 実際のアーキテクチャーでは存在しない
右シフト(1/2倍)		
java	a >>> 1 7FFFFFEE	a >> 1 -18

例6

- x = 0.FEDCの4倍
  - 4倍 = $100_{(2)}$ 倍=左2ビットシフト
  - x = 0.1111 1110 1101 1100
  - 4x = 11.11 1110 1101 1100

    =

例7

- 次の演算結果をもとめよ.
  1. $52_{(10)} << 1$
  2. $52_{(10)} << 2$
  3. $52_{(10)} >>> 1$
  4. $52_{(10)} >> 1$
  5. $-52_{(10)} >> 1$
  6. $-52_{(10)} >>> 1$

7. 誤差のモンダイ

- 1
  1. int a = 3;
  2. int b = 4;
  3. println(a == a+b);
- 2
  1. double c = 1.25e18;
  2. double d = -1.25e-5;
  3. println(c == c+d);

情報落ち

- 3
  1. byte e = 100;
  2. byte e2 = (byte)(e + e);
  3. println(e2);

オーバフロー

- 4
  1. byte e = -100;
  2. byte e4 = (byte)(e + e);
  3. println(e4);

アンダーフロー

誤差の整理

種類	定義	例
情報落ち	絶対値の差がある加減算で小さな値が「落ちる」	1.25e18 -1.25e-5
けた落ち	値がほぼ等しい演算で有効桁数が減る	1.25e-5 - 1.24999e-5
打切り誤差	演算を一定の桁数で打ち切る	1/3 = .333334
丸め誤差	表現桁数より小さな値を「丸めて」切り捨てる	.999999 = 1.0
オーバーフロー	演算結果が数値範囲を上回る	100+100 = -56
アンダーフロー	演算結果が数値範囲を下回る	-100-100 = +56

誤差の大きさ

- 相対誤差
  - 定義: (絶対誤差)/真値
  - 例）真 x = 100; 計算結果 y = 101
  - 絶対誤差: |x – y| = +1
  - 相対誤差: 1/100 = 0.01

まとめ

- 整数xをある値から引いて表現された負数をxの（　　　　　　　　）といい，ビット反転された（　　　　　　　　）+1で求められる.
- 小数点の位置を固定した表現方法を（　　　　　　）という．浮動小数点表示の -6.25×10^3 ，-を（　　　），6を（　　　）数，3を（　　　）数という.
- xを2倍した値は（　）シフトで得られる．右シフトには符号なしの（　　　　　）と符号ありの（　　　）がある.
- 絶対値の差がある演算で生じる誤差に（　　　　　　）がある．LSBより小さな桁を切り捨てる誤差を（　　　　）誤差という.