コンピュータ・サイエンス1

第8回
コンピュータでの情報の扱い方(4)

人間科学科コミュニケーション専攻
白銀 純子

第8回の内容

- コンピュータでの情報の扱い方(4)

前回の出席問題の解答

- 数を6桁の2進数で表しているコンピュータがある。このとき、以下の数のなかで、オーバーフローしている数はどれか、全て答えなさい。
  1. 00110 ←5桁
  2. 11001 ←5桁
  3. 1010111 ←7桁
  4. 010101 ←6桁
  5. 111000 ←6桁
  6. 101 ←3桁
  7. 1111111111 ←10桁

解答: 3, 7

前回の出席問題の解答

- 設問2:「やってみよう」の10進数⇔16進数の変換問題の1. と4. の計算結果を報告すること
  - 1. の問題: 10進数の「240」を16進数に

16) 240
16) 15・・・余り: 0
0・・・余り: 15

15 0

10以上の余りはA～Fに置き換え
- $(15)_{10} = (F)_{16}$

解答: $(F0)_{16}$

前回の出席問題の解答

- 設問2:「やってみよう!」の10進数⇔16進数の変換問題の1. と4. の計算結果を報告すること
  - 1. の問題: 16進数の「64」を10進数に

16^1 16^0
× ×
6 4

$6 \times 16^1 + 4 \times 16^0$
$= 6 \times 16 + 4 \times 1$
$= 96 + 4$
$= 100$

解答: $(100)_{10}$

前回の質問の回答

24進数・32進数

- XX進数はいくつでもあり
- コンピュータ関係で出てくるのは、2進数・16進数がほとんど

練習問題の解答

- 授業の資料のページの「練習問題」の欄
  - 練習問題を出した次の回の授業の前までに掲載
  - 練習問題を出した回の「練習問題」の欄のリンク先に掲載
    - Ex. 第7回で出した問題の解答は、第7回の資料の「練習問題」の欄のリンク

Question!

前回の復習

負の数の表現[1](p. 9)

- コンピュータでの計算は、全て足し算
  - コンピュータでの計算は、電気回路で実行
    - 電気回路: 電気が通る線を組み合わせて、様々な処理をするためのもの (コンピュータを構成する最も基本的な部品)
  - 足し算, 引き算, かけ算, 割り算をするには、それぞれのために専用の回路が必要
    - 足し算専用回路, 引き算専用回路, かけ算専用回路, 割り算専用回路

経済的に良くない

足し算専用回路(加算器)を
組み合わせて他の計算をカバー

負の数の表現[2](p. 9)

- 足し算の組み合わせで他の計算も行う
  - 引き算: 「a-b」を、「a+(-b)」(bを負数と考える)
  - かけ算: 足し算の繰り返しとして計算
  - 割り算: 引き算の繰り返しとして計算

コンピュータでも負数を扱う
- 方法1: 真数表現
- 方法2: 2の補数表現

真数表現(p. 9)

- 数を表す2進数に符号(+ or -)を表す1ビットを付加
  - 数の先頭のビットで符号を表す
  - 0が「+」, 1が「-」を表す

「符号ビット」と呼ぶ

10進数	2進数
-3	1 0 1 1
-2	1 0 1 0
-1	1 0 0 1
-0	1 0 0 0
+0	0 0 0 0
+1	0 0 0 1
+2	0 0 1 0
+3	0 0 1 1

符号ビット

数を表す部分

0が「+0」と「-0」の
2種類できてしまう

具合が悪いので
真数表現はあまり使われない

2の補数とは?

- そもそも「補数」とは: ある数Nに足したときに桁が1つ上がる数
  - 2進数に限らず、どんなN進数であっても、「補数」は存在
  - 2進数での補数が「2の補数」

5桁の数XXXXXに対して...
- XXXXX + YYYYY = 100000
  となるとき、「XXXXX」の補数が「YYYYY」
- Ex. 2進数で「01001」だと...
  01001 + 10111 = 100000 なので、「01001」の補数は「10111」

2の補数をマイナスの数として扱うと...
- 5桁の計算の例: $(01001)_2 + (10111)_2 = (100000)_2 \rightarrow (00000)_2$ (オーバーフロー)
  - ✓ $(01001)_2 = (9)_{10}$, $(10111)_2 = (-9)_{10}$ なので、$(9)_{10} + (-9)_{10} = (0)_{10}$
    なので、計算がきちんとできている!

2の補数表現[1](p. 9)

- 負の数Nを、正の数N(2進数)の0と1を反転させて1を加えた数で表現する方法
  - 0と正の整数(自然数)は、そのまま表現(この計算はしない)

Ex.
$(-10)_{10}$

「10」を2進数にして
「-」をつけたもの

$= (-01010)_2$

$\rightarrow (10101 + 1)_2 = (10110)_2$

「-01010」の「-」をとって
「1」と「0」を逆にしたもの

$(-10)_{10}$の2進数
(2の補数表現)

2の補数表現[2](p. 9)

- 2の補数 = 負の数を2進数で表現したもの(コンピュータの世界では)
- 計算方法(例: -20を10桁の2進数に直す)
  1. 2の補数に直したい10進数のマイナスを取り除く
     - $(-20)_{10} \rightarrow (20)_{10}$
  2. 1. の結果を2進数に直す
     (この時点で桁数あわせ!! 後で桁数あわせをすると、合わせ方を間違えやすい)
     - $(20)_{10} = (0000010100)_2$
  3. 2. の結果の0と1を逆にする(0の桁を1、1の桁を0にする)

0 0 0 0 0 1 0 1 0 0

1 1 1 1 1 0 1 0 1 1

2の補数表現[2](p. 9)

- 2の補数 = 負の数を2進数で表現したもの(コンピュータの世界では)
- 計算方法(例: -20を10桁の2進数に直す)
  4. 3. の結果に1を足し算する

```
  1111101011
+)         1
  1111101100
```

-20を2進数に直した結果
(2の補数 = 2進数での負の数の表現)

2進数での負の数の表現では、
「-」の記号はつけない

2の補数表現の利点(p. 10)

- 引き算(符号付きの足し算)をそのまま足し算として処理できる

Ex.
$(10 + 3)_{10} = (1010 + 0011)_2 = (01101)_2 = (13)_{10}$
$(6 - 3)_{10} = (110 + \boxed{101})_2 = (1011)_2 \rightarrow (011)_2$ (オーバーフロー) $= (3)_{10}$

「-3」の2の補数表現

真数表現: 符号付きの足し算を処理するには、別の回路が必要
(単純に足すことはできない)

2の補数を10進数に変換[1]

- 2の補数から1を引き、0と1を反転させて10進数になおして「-」をつける
  - 負の数を2の補数に変換するときの逆
  - この計算は、負の数だけ

Ex.

$(110110)_2$

2の補数から1を引いたもの

$\rightarrow (110110 - 1)_2 = (110101)_2$

「110101」の0と1を逆にしたもの

$\rightarrow (-001010)_2$

$(110110)_2$(2の補数)の10進数

$= (-10)_{10}$

 19

2の補数を10進数に変換[2]

- 計算方法(例: 1111101100を10進数に直す)

1. 2の補数から1を引き算する

$$\begin{array}{r} 1111101100 \\ -)\quad\quad\quad\quad 1 \\ \hline 1111101011 \end{array}$$

2. 1. の結果の0と1を逆にする(0の桁を1、1の桁を0にする)

1111101011

↓

0000010100

※2の補数→10進数の方法は、10進数→2の補数の逆

 20

2の補数を10進数に変換[2]

- 計算方法(例: 1111101100を10進数に直す)

1. 2. の結果を10進数に直す
   - $(0000010100)_2 = (20)_{10}$
2. 3. の結果に-(マイナス)をつける
   - $(20)_{10} \rightarrow (-20)_{10}$

1111101100を10進数に直した数

※2の補数→10進数の方法は、10進数→2の補数の逆

 21

2進数の引き算[1]

- 10進数の引き算だと…
  - ある桁の引かれる数が引く数より小さければ、1つ大きな桁から10を借りる
    - 10を借りる: 貸した桁から1を引き、借りた桁に10を足す

10を借りる　　　　10を借りる

$$\begin{array}{r} 1\ 0\ 0 \\ -)\quad\ \ 1 \\ \hline \end{array} \Rightarrow \begin{array}{r} 0\ 10\ 0 \\ -)\quad\quad 1 \\ \hline \end{array} \Rightarrow \begin{array}{r} 0\ 9\ 10 \\ -)\quad\quad 1 \\ \hline 0\ 9\ 9 \end{array}$$

引き算の答え: 99

 22

2進数の引き算[2]

- 2進数の引き算だと…
  - ある桁の引かれる数が引く数より小さければ、1つ大きな桁から $(10)_2$(10進数で2)を借りる
    - 2を借りる: 貸した桁から1を引き、借りた桁に2を足す

2(2進数で10)を借りる　　2(2進数で10)を借りる

$$\begin{array}{r} 1\ 0\ 0 \\ -)0\ 0\ 1 \\ \hline \end{array} \Rightarrow \begin{array}{r} 0\ 2\ 0 \\ -)0\ 0\ 1 \\ \hline \end{array} \Rightarrow \begin{array}{r} 0\ 1\ 2 \\ -)0\ 0\ 1 \\ \hline 0\ 1\ 1 \end{array}$$

引き算の答え: 011

※コンピュータ的には引き算はしないので、人間が2の補数→10進数の計算をするための引き算

 23

やってみよう[5]

1. -25を2の補数10桁で表現
2. -32を2の補数10桁で表現
3. 2の補数10000を10進数で表現
4. 2の補数1011000を10進数で表現

 24

Question!

正の数と負の数の見分け方[1]

- 大前提: 数を表す2進数の桁数は決まっている
  - 普通のコンピュータで32桁(or 64桁)

ということは…例えば$(10)_{10}$は、コンピュータ的には…

$\underbrace{\boxed{0000....0000}}_{28個の「0」}1010$ と考えている

※授業のスライド中では32桁分も書けないので、そのときどきで適当なところで割愛

正の数と負の数の見分け方[2]

- 負の数(2の補数)の計算方法: 負の数Nを、正の数N(2進数)の0と1を反転させて1を加える

コンピュータ的には32桁で数を表すので…

$(-10)_{10} \rightarrow (10)_{10}$
$= (0000...00001010)_2$
$\rightarrow (\boxed{1111...1111}0101 + 1)_2 = (1111...11110110)_2$

28個の「0」も全て「1」に反転される

➡ 負の数は結果的に一番大きな桁(一番左の桁)が「1」になる

➡ 一番大きな桁(一番左の桁)が「0」であれば正の数、「1」であれば負の数として扱う

正の数と負の数の見分け方[3]

- 2進数を見たときに…(2の補数を考える場合)
  - 「2の補数を考える」という場合は、先頭の桁を見て、正の数か負の数かを判断
  - 「2の補数を考える」と書かれていない場合は、負の数を考えなくてOK

「0」で始まっているので、正の数

0101010101

00001111111100

「1」で始まっているので、負の数

111100001111

1010101010

2進数の数

正の数と負の数の見分け方[3]

- 2進数で表された数は、一番大きな桁(一番左の桁)が「0」であれば正の数、「1」であれば負の数
- 10進数で表された数は、普通に正の数、負の数として計算
  - 正の数であれば、割り算だけで2進数に変換
    - Ex.「+5」または「5」と書かれていれば、割り算だけで2進数に変換
  - 負の数であれば、2の補数の方法で2進数に変換
    - Ex.「-5」と書かれていれば、2の補数の方法で2進数に変換

桁あふれ(オーバーフロー)(p. 11)

- 2の補数に関係した桁あふれ(オーバーフロー)が起こりうる

Ex. 2進数5桁の計算(10進数で14+5の計算)

```
   0 1 1 1 0
+) 0 0 1 0 1
-------------
   1 0 0 1 1
```

先頭の桁が1になってしまった

➡ 先頭の桁が1の場合は、2進数で負の数として扱う

1 0 0 1 1

負の数を表す

➡ 計算結果: $(-13)_{10}$(負の数)

2の補数に関係した桁あふれ(オーバーフロー)

負数込みの計算の考え方

- 計算結果を見て…(オーバーフローがどーのと考えるのではなく!)
  1. 計算結果が指定の桁数を超えていれば、超えた分の桁を削除(桁数あわせ)

     桁数を超えた部分 = 削除

     4桁の2進数の計算結果: $(100110)_2$ ➡ 計算結果: $(0110)_2$
  2. 計算結果の先頭の桁が0か1かで、正か負を判断
     - 正の数(先頭の桁が0)であれば、割り算だけで10進数に直す
     - 負の数(先頭の桁が1)であれば、2の補数の方法で10進数に直す

     正の数と判断

     4桁の2進数の計算結果: $(0101)_2$ ➡ 計算結果: $(5)_{10}$

     負の数と判断

     4桁の2進数の計算結果: $(1010)_2$ ➡ 計算結果: $(-6)_{10}$

桁あふれ[まとめ][1]

- 桁あふれの分類(その1)
  - 足し算等の何らかの計算の結果、コンピュータが扱うことのできる数の桁数の限界を超えてしまう場合
    - Ex. 4桁の数「0110+0110+0110」の計算
      →本来の計算結果は「10010」で5桁になってしまうので、5桁目が無視されてコンピュータが出す結果は「0010」
      →コンピュータが出す結果と本来の結果が違うことになる現象

ある意味、コンピュータの性能の限界を超えてしまうということで、その結果として、本来の計算結果とは違う結果がでる現象

※人間やコンピュータがミスをした、という現象ではない

桁あふれ[まとめ][2]

- 桁あふれの分類(その2)
  - 足し算等の何らかの計算の結果、数の正と負が違ってしまう場合
    - Ex. 4桁の数「0110+0110」の計算
      →本来の計算結果は「1100」で、1桁目が1なので、コンピュータは計算結果を負の数(-4)として取り扱い
      →本来の計算結果は正の数(12)
      →コンピュータが出す結果と本来の結果が違うことになる現象

本来の計算結果は正(負)の数なのに、計算結果が負(正)の数になってしまう現象

※人間やコンピュータがミスをした、という現象ではない

やってみよう[6]

1. (+10)+(+8)を5桁の2の補数として計算し、10進数として表現
2. (-10)+(+8)を5桁の2の補数として計算し、10進数として表現

オーバーフローも考慮すること

Question!

小数の表現方法

整数以外の数を表現するには?

- コンピュータが表現できる数: 整数, 小数
- 整数以外の数
  - 小数
  - 分数
  - n乗根(平方根, 立方根, etc)
  - π(円周率)
  - etc.

全て小数として表現

10進数の小数を2進数に直す[1]

1. 10進数の整数部分は、通常の方法で2進数に直す
2. 10進数の小数部分に2をかけ算する
3. 2. の結果、整数部分を小数点第1桁にする
4. 3. の結果、小数部分をまた10進数の数とする
   - 小数部分が0になるまで繰り返す
     - ✓ 無限小数になることも(10進数できりのいい数も2進数では無限小数になりえる)
   - 2. の整数部分を小数点第2桁、第3桁、…と置いていく
   - 1. の整数部分と2. の整数部分を並べたものが2進数での小数になる

10進数の小数を2進数に直す[2]

- 10進数の小数を2進数に直すには?(例1)
  - 10進数の0.8125を2進数に直す

0.8125 ×) 2 → 1.625
0.625 ×) 2 → 1.25
0.25 ×) 2 → 0.5
0.5 ×) 2 → 1.0

0. 1 1 0 1　　0になったので計算終了

$(0.8125)_{10} = (0.1101)_2$

10進数の小数を2進数に直す[3]

- 10進数の小数部分を2進数に直すには?(例2)
  - 10進数の3.8を2進数に直す(整数部分の3は2進数で11)

0.8 ×) 2 → 1.6
0.6 ×) 2 → 1.2
0.2 ×) 2 → 0.4
0.4 ×) 2 → 0.8

0. 1 1 0 0　　0.8になって最初に戻ったので、無限小数

$(3.8)_{10} = (11.110011001100....)_2$

2進数の小数を10進数に直す[1]

- 例: 101.1101 (整数部分101は10進数で5)

1. 2進数の整数部分は、通常の方法で10進数に直す
2. 10進数の小数部分各桁の上に「1/2」を書く
3. 2. で書いた「1/2」の「2」の右肩に左から1, 2, 3, …と書いていく
   - $1/2^0, 1/2^1, 1/2^2$, …ができていく

2. $\frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}\ \frac{1}{2}$
0 . 1 1 0 1

左から右に、1, 2, 3, …と番号をつける

3. $\frac{1}{2^1}\ \frac{1}{2^2}\ \frac{1}{2^3}\ \frac{1}{2^4}$
0 . 1 1 0 1

2進数の小数を10進数に直す[2]

4. 各桁の上の「$1/2^n$」と、それぞれの桁の数をかけあわせる
5. 4. の結果を足し合わせ、1. の整数部分とあわせる

3. $\frac{1}{2^1}\ \frac{1}{2^2}\ \frac{1}{2^3}\ \frac{1}{2^4}$
0 . 1 1 0 1

4. $\frac{1}{2^1}\ \frac{1}{2^2}\ \frac{1}{2^3}\ \frac{1}{2^4}$
× × × ×
0 . 1 1 0 1
=
$\frac{1}{2^1}\ \frac{1}{2^2}\ 0\ \frac{1}{2^4}$

5. $\frac{1}{2^1}\ \frac{1}{2^2}\ 0\ \frac{1}{2^4}$
足し合わせる
=
0.5 + 0.25 + 0.0625 = 0.8125
1. の整数部分とあわせる
5.8125

やってみよう[7]

- 2進数1.101を10進数に変換
  (2010年度ITパスポート春季試験問題)
- 2進数に変換した時、有限小数で表現できる10進数は、
  以下のうちどれか
  - 0.1
  - 0.2
  - 0.4
  - 0.5

  (2012年度ITパスポート秋季試験問題)

 43

小数の表現方式

 44

小数を表現する方法(p. 10)

- 固定小数点方式
- 浮動小数点方式

 45

固定小数点方式[1](p. 10)

- 小数部分の桁数がnの場合: 2進数では、小数部分が$1/2^n$刻みで表現

nを大きくすると、それだけ小数部分を細かく表現可能

※ただし、実際コンピュータは小数も2進数で考えているが、
人間が考えるときの便宜上、10進数で考えることが多い

 47

固定小数点方式[2](p. 10)

- 小数を表す桁数が決まっていると…

Ex. 右から2桁を小数部分とすると…
$(2 \div 1000)_{10} = (0.002)_{10} \rightarrow (0.00)_{10}$
小数を正確に表現できない

何桁分の小数部分を持っているかは数値によって異なる
=固定小数点方式で小数を表せる場合は少ない

浮動小数点方式

 48

浮動小数点方式[1](p. 10)

- 小数: $D \times 10^n$と表現できる
  - Ex. 1: $0.5 = 5 \times 10^{-1}$
  - Ex. 2: $-0.0625 = -6.25 \times 10^{-2}$
  - Ex. 3: $0.0000000084 = 8.4 \times 10^{-9}$

どの数でも「$\times 10^n$」の「10」の部分は同じ(実際のコンピュータでは「$\times 2^n$」

小数を「$D \times 10^n$」の形と考え、「D」と「n」だけ記憶しておく

 49

浮動小数点方式[2](p. 10)

- 浮動小数点方式:
  小数を「D×10ⁿ」と考え、「D」と「n」を記憶することで小数を表す方式

Ex.
D = 6.25, n = -3の場合: 0.00625
D = 6.25, n = -2の場合: 0.0625
D = 6.25, n = -1の場合: 0.625
D = 6.25, n = 0の場合: 6.25

nの数値が何かで、小数点が仮数の中を動くように見えるから「浮動小数点」と名づけられた

D: 仮数部
n: 指数部
と呼ぶ

浮動小数点方式[3](p. 10)

- 仮数部
  - 符号は「0」が「+」、「1」が「-」
  - 固定小数点方式
- 指数部
  - 符号は「0」が「+」、「1」が「-」(ただし、2の補数表現とは別の特殊な形で表現される)

大きな数の表現(p. 10)

- 浮動小数点方式を利用して表現

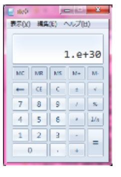
Ex.
- 2000000000000 = 2×10^{12}
- -425000000000000000 = -4.25×10^{17}

指数部が「+」の数になる

コンピュータは「2」と「+12」, 「-4.25」と「+17」を記憶しておく
(実際には、「$\times 10^n$」ではなく「$\times 2^n$」で表現)

大きな数の表現[例](p. 10)

- Windowsの電卓のあるモード(オーバーフローをなかなかしないモード)で…



Ex. 1000000000000000 × 1000000000000000
= 1.e+30

1.0×10^{30} という意味
(浮動小数点方式での表現)

※オーバーフローしにくいモードは、大きな数を浮動小数点方式で表現する
= それだけたくさんの桁がある数を表現できるので、オーバーフローしにくい

桁落ち(1)

- 小数部分が無限のものを扱えるわけではない
  - 例えば割り算で割り切れない数や円周率

➡ 小数部分を適当なところで切り捨てる(四捨五入ではない)

例えば…1÷7:
コンピュータは「0.142857…142」のように考える
本来はこの後も無限に続く

コンピュータが扱える小数の桁数:
「有効桁数」と呼ぶ

桁落ち(2)

- 小数部分が無限のものは適当なところまでで切り捨てられる

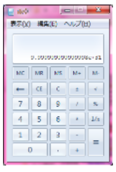
本来の数よりも、小数の桁数が小さくなってしまう現象

桁落ち

桁落ちの例

- Windowsの電卓のあるモード(オーバーフローをなかなかしないモード)で…

Ex. 電卓での計算: 9999999999999999 × 9999999999999999
→ 9.999999999999998e+31 = 9.999999999999998 × 10^{31}
= 99999999999999980000....00000



でも実際に計算すると…

$$\begin{array}{r} 9999999999999999 \\ \times)\underline{9999999999999999} \\ \ldots 91 \\ +)\underline{\ldots 91} \\ \ldots 01 \end{array}$$

つまり本当の計算では…
9999999999999999 × 9999999999999999
= x.xxxx…01

桁落ち

ミニ実習

- スマートフォンの電卓で、大きな数のかけ算をしてみよう!
  - 結果が浮動小数点表示になるか?
  - 桁落ちしているか?

を確認しよう!

桁落ちが起こると…

- 数が本来の数よりも小さくなってしまう
  - 微妙な数値が必要な場合には要注意
- 桁落ちをした数に大きな数をかけると、本来の数に大きな数をかけたときとの差が大きくなる

例えば…有効桁数が小数点第3位とすると、1÷7×100000の計算は…

- コンピュータ: 「1÷7」をして0.142に桁落ちし、それに100000をかけて、14200
- コンピュータ: 1×100000を7で割ると(計算の順序を変えると)、14285.714
- 人間: 本来の1÷7に100000をかけると、14285.714285….

- コンピュータで計算をするときは、計算の順番に注意(割り算はなるべく後にすること)
- 例えば「1÷7×100000」の計算は、「1×100000」をしてから7で割る

やってみよう[8]

1. 0.0000055を浮動小数点方式で表現
2. 0.000000001234を浮動小数点方式で表現
3. 456000000000000を浮動小数点方式で表現
   ※3つとも仮数部は小数点第2位の小数とすること
4. (10÷7)×10000を計算
   - 小数点第2位までが有効桁数
   - 桁落ちを考えて計算すること
5. 10÷7×10000を計算
   - 小数点第2位までが有効桁数
   - 桁落ちの影響がなるべく少ないように計算すること

Question!