

コンピュータ・サイエンス1

第7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7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

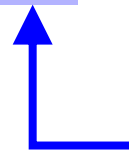
前回の出席問題の解答

- 「やってみよう!」の10進数 $\Leftrightarrow$ 16進数の変換問題の1. 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

$$\begin{array}{r} 16 \overline{) 240} \\ 16 \overline{) 15} \cdots \text{余り: } 0 \\ 0 \cdots \text{余り: } 15 \end{array}$$


15 0



10以上の余りはA～Fに置き換え

➤ $(15)_{10} = (F)_{16}$

解答: (F0)₁₆

前回の質問の解答

第5回授業の出席問題の解答について

- 設問2: 「やってみよう!」の10進数 $\Leftrightarrow$ 2進数の変換問題の4. と6. の計算結果を報告すること
 - 6. の問題: 2進数の「001010101010」を10進数に
誤: 解答: $(682)_2$
正: 解答: $(682)_{10}$
- 設問3: 「やってみよう!」の2進数の足し算の3. の計算結果を報告すること
 - 3. の問題: $10010010 + 11001100$ の結果を10進数で(8ビットでオーバーフローも考慮)
誤: 解答: $(94)_2$
正: 解答: $(94)_{10}$

シフト算

- シフト算: 2進数の桁を左右に移動させることで、かけ算・割り算をする計算方法
 - 左にシフト
 - 2進数の桁を左に移動させる
 - 移動させた桁数分だけ、右側に0をつける
 - Ex. 1011を左に1ビット(1桁分)シフトすると: 10110
 - 右にシフト
 - 2進数の桁を右に移動させる
 - 移動させた桁数分が小数部分になる
 - Ex. 1011を右に1ビット(1桁分)シフトすると: 101.1

音

- 音もビットで表現する

- 音は、高い音・低い音、大きい・小さいをグラフで表現可能
- 音を読み取る装置を使ってグラフ化し、コンピュータに取り込み

 この授業のもっと後で説明

Question!

前回の復習

8進数と16進数[1]

□ コンピュータでの情報: 2進数で扱われる

情報量が多くなると桁数が大きくなって、人間には扱いにくい

Ex.

アルファベットの「N」: 01001110 (8桁)

日本語の「ん」: 1010010011110011 (16桁)

赤色: 111111110000000000000000 (24桁)

人間にとっては扱いにくい

(コンピュータの制御を考えると、人間もコンピュータのように考える必要)



8進数&16進数

8進数と16進数[2]

- 8進数: 数を0～7の8つの数字で表現
- 16進数: 数を0～9とA～Fの16個の文字で表現

- A: 10
- B: 11
- C: 12
- D: 13
- E: 14
- F: 15

覚えよう!

※いろいろな試験で、電卓の持ち込みはできないので、
きちんと自分で計算できるようになろう!

16進数

□ 16進数が特によく使われる

- コンピュータの世界では0～255の数値で表現されるものが多い

Ex. 色

- 色は赤・緑・青の濃淡を混ぜ合わせて表現する
- 赤・緑・青を0～255の256段階の濃淡を混ぜ合わせる

赤成分: 255, 緑成分: 102, 青成分: 153 →

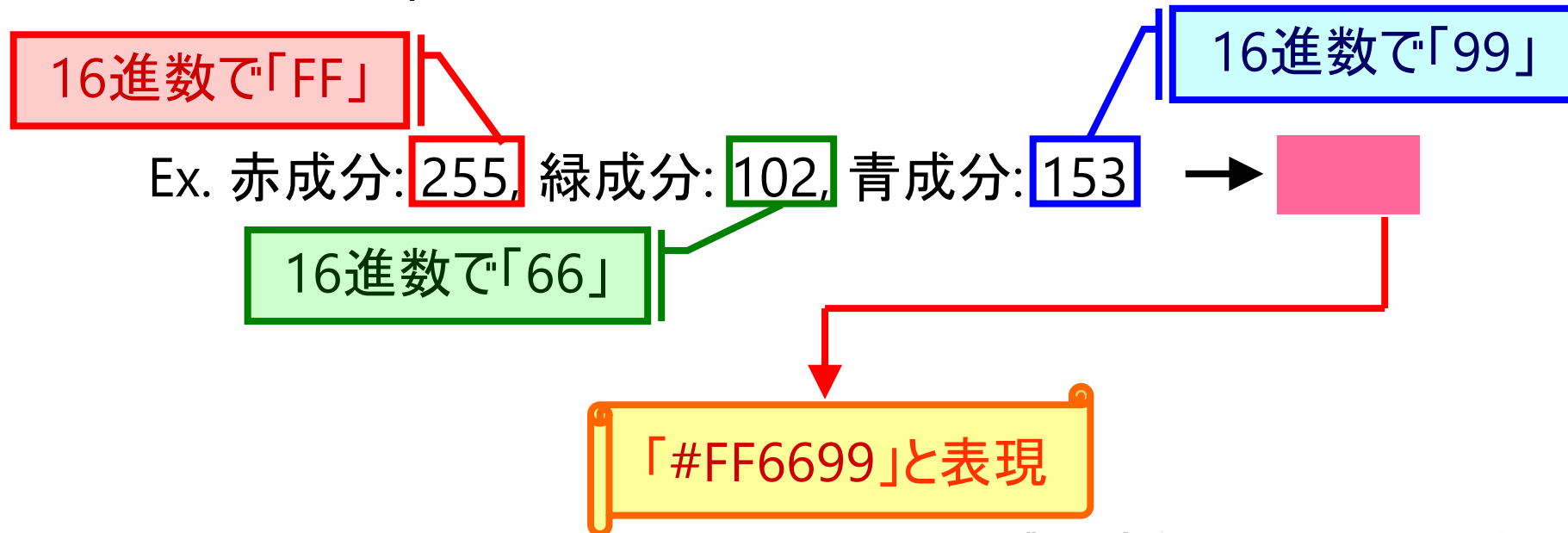


0～255を16進数で表すと0～FFで、ちょうど2桁で表せる

※色のほかに、文字も16進数で表すことが多い
(半角英数: 16進数2桁, 全角: 16進数4桁)

16進数がよく使われる例

- 色の表現: 赤・緑・青の256段階の濃淡で表現
 - それぞれの濃淡の度合いを0～255の数値で表現
 - 濃淡の度合いの数値を16進数で表現
 - 16進数の数値を赤・緑・青の順に並べ、先頭に「#」をつけて色を表現 (色の名前として利用)



※Webページ作成などのときによく使う

10進数を16進数に変換[1]

16進数を求める計算方法

1. 10進数の数を16で割って商1と余り1を計算する
2. 商1を16で割って商2と余り2を計算する
3. 商2を16で割って商3と余り3を計算する
4.

16) 10進数の数
16) 商1...余り1
16) 商2...余り2
.
.
16) .
商n...余りn

商が0になるまで繰り返す
小数の計算はしない

余り1
余り2
.
.
.

時計回りに倒す

余りn-1
余りn

余り
2
余り
1

余り
2
余り
1

余りを余りnから余り1の順に左から並べたものが16進数
(ただし10~15の余りは、A~Fに置き換えること)

10進数を16進数に変換[2]

変換の手順

- 10進数を16で割って商と余りを計算する
- 商を16で割って商と余りを計算する
 - 商が0になるまで繰り返す

1. や2. の手順での余りで、10以上のものがあれば、A～Fに置き換える

- 余りを後に出てきたものから順に並べる
 - 余りを時計回りに倒す

$$\begin{array}{r} 16 \overline{) 240} \\ 16 \overline{) 15} \cdots \text{余り: } 0 \\ 0 \cdots \text{余り: } 15 \end{array}$$

余りを「150」と並べたら...????

- 3桁の16進数(「1」と「5」と「0」という3つの桁がある)?
- 2桁の16進数(「15」と「0」という2つの桁がある)?

区別がつかない!

➤ $(10)_{10} = (A)_{16}$, $(11)_{10} = (B)_{16}$, $(12)_{10} = (C)_{16}$, $(13)_{10} = (D)_{16}$,
 $(14)_{10} = (E)_{16}$, $(15)_{10} = (F)_{16}$ と置き換える

10進数を16進数に変換(例)

10進数の255を16進数に変換

$$\begin{array}{r} 16 \overline{) 255} \\ 16 \overline{) 15} \cdots \text{余り: } 15 \\ \underline{ 0} \cdots \text{余り: } 15 \end{array}$$



$$(15)_{10} = (F)_{16}$$



$$(255)_{10} = (FF)_{16}$$

10進数の2000を16進数に変換

$$\begin{array}{r} 16 \overline{) 2000} \\ 16 \overline{) 125} \cdots \text{余り: } 0 \\ 16 \overline{) 7} \cdots \text{余り: } 13 \\ \underline{ 0} \cdots \text{余り: } 7 \end{array}$$



$$(13)_{10} = (D)_{16}$$



$$(2000)_{10} = (7D0)_{16}$$

→ の方向に余りを並べる

16進数を10進数に変換

□ 16進数→10進数の変換

1. アルファベットを10進数の数になおす
2. 2進数の各桁の上にそれぞれ「16」を書く
3. 1. で書いた「16」の右肩に、右から0, 1, 2, ...と書いていく
4. 16^0 , 16^1 , 16^2 , ...ができていく

1. 7D0 → 7 13 0



2. 16 16 16
7 13 0



右から左に、0, 1, 2, ...と番号をつける

3. 16^2 16^1 16^0
7 13 0

16進数を10進数に変換

□ 16進数→10進数の変換

5. 各桁の上の「 16^n 」と、それぞれの桁の数をかけあわせる

6. 2. の結果を足し合わせる

3.

16^2	16^1	16^0
7	13	0



4.

16^2	16^1	16^0
×	×	×
7	13	0
7×16^2	13×16^1	0



5.

7×16^2	13×16^1	0
足し合わせる		
$7 \times 16^2 + 13 \times 16^1 + 0 = 2000$		

やってみよう!

1. 16進数の「64」を10進数に
2. 16進数の「FA0」を10進数に
3. 16進数の「4E20」を10進数に
4. 16進数の「A3」を10進数に
(2012年度ITパスポート秋季試験問題)

Question!

8進数

8進数

- 16進数ほどではないが、知っておくべき数の表現方法
 - 情報処理技術者試験などには出ることも

10進数を8進数に変換

8進数を求める計算方法

1. 10進数の数を16で割って商1と余り1を計算する
2. 商1を16で割って商2と余り2を計算する
3. 商2を16で割って商3と余り3を計算する
4.

8) 10進数の数
8) 商1...余り1
8) 商2...余り2
.
.
.
8) 商n...余りn

商が0になるまで繰り返す
小数の計算はしない

➡ 余りを余りnから余り1の順に左から並べたものが8進数

余り1
余り2
.
.
.
余りn-1
余りn

時計回りに倒す

余り
2
余り
1
.
.
.

※10進数→X進数の計算は、割る数がXになるだけで、やり方はどのX進数でも同じ

10進数を8進数に変換(例)

10進数の255を8進数に変換

$$\begin{array}{r} 8 \overline{) 255} \\ 8 \overline{) 31} \cdots \text{余り: } 7 \\ 8 \overline{) 3} \cdots \text{余り: } 7 \\ 0 \cdots \text{余り: } 3 \end{array}$$



$$(255)_{10} = (377)_8$$

10進数の2000を8進数に変換

$$\begin{array}{r} 8 \overline{) 2000} \\ 8 \overline{) 250} \cdots \text{余り: } 0 \\ 8 \overline{) 31} \cdots \text{余り: } 2 \\ 8 \overline{) 3} \cdots \text{余り: } 7 \\ 0 \cdots \text{余り: } 3 \end{array}$$



$$(2000)_{10} = (3720)_8$$

→ の方向に余りを並べる

8進数を10進数に変換

□ 8進数→10進数の変換

1. 8進数の各桁の上にそれぞれ「8」を書く
2. 1. で書いた「8」の右肩に、右から0, 1, 2, ...と書いていく
 - $8^0, 8^1, 8^2, \dots$ ができていく

1.

8	8	8	8
3	7	2	0



2.

右から左に、0, 1, 2, ...と番号をつける

³ 8	² 8	¹ 8	⁰ 8
3	7	2	0

8進数を10進数に変換

□ 8進数→10進数の変換

3. 各桁の上の「 8^n 」と、それぞれの桁の数をかけあわせる
4. 2. の結果を足し合わせる

2.

8^3	8^2	8^1	8^0
3	7	2	0



3.

8^3	8^2	8^1	8^0
×	×	×	×
3	7	2	0
3×8^3	7×8^2	2×8^1	0



4.

3×8^3	7×8^2	2×8^1	0
$3 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 0 = 2000$			

足し合わせる

※X進数→10進数の計算は、累乗する数がXになるだけで、やり方はどのX進数でも同じ

やってみよう!

1. 10進数の「200」を8進数に
2. 10進数の「1000」を8進数に
3. 10進数の「555」を8進数に
4. 8進数の「100」を10進数に
5. 8進数の「2734」を10進数に
6. 8進数の「55」を16進数に
(2009年度ITパスポート秋季試験問題)

負数の表現方法

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

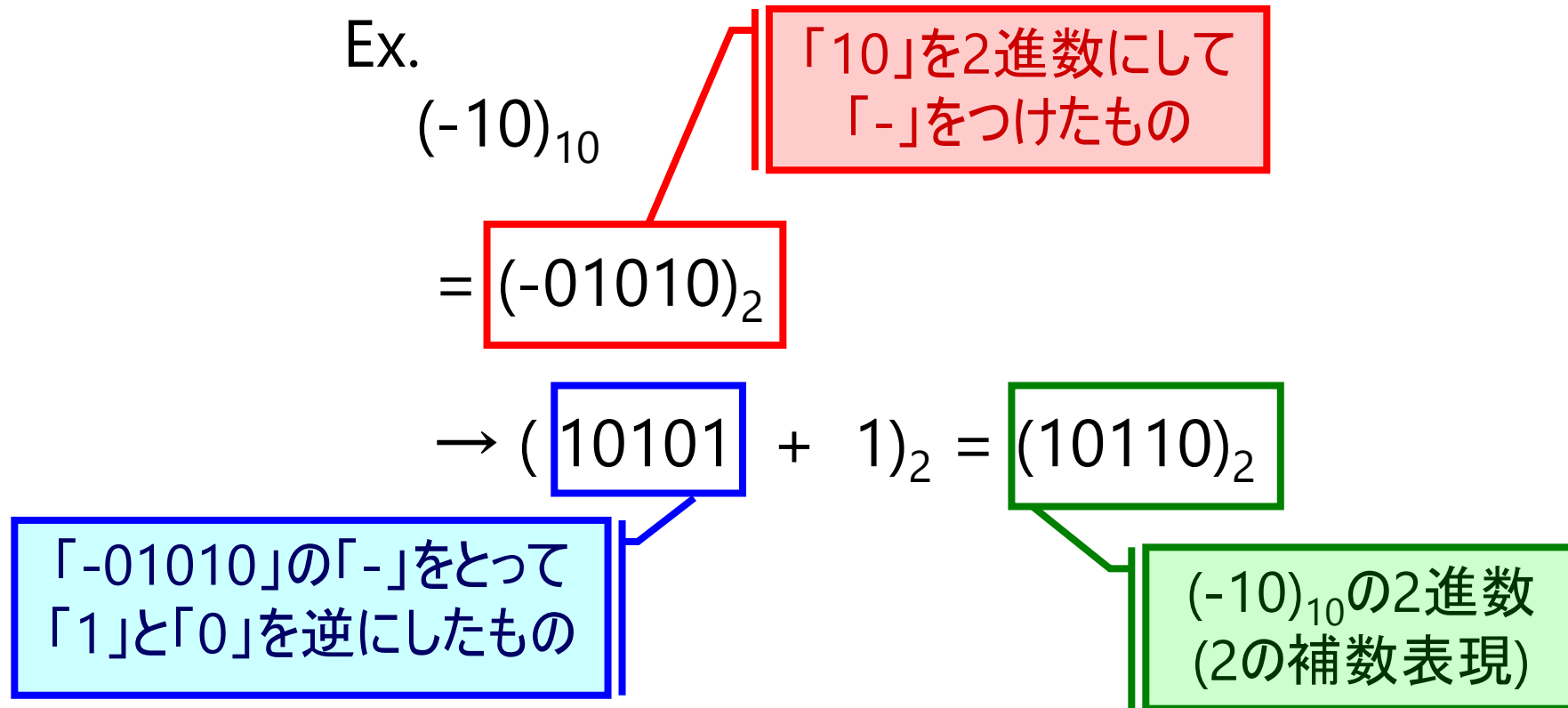
数を表す部分

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)



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
 - $(20)_{10} = (0000010100)_2$
 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)
 3. の結果に1を足し算する

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

-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.

$$(10 + 3)_{10} = (1010 + 0011)_2 = (01101)_2 = (13)_{10}$$

$$(6 - 3)_{10} = (110 + \boxed{101})_2 = (1011)_2 \rightarrow (011)_2 \text{ (オーバーフロー)} = (3)_{10}$$



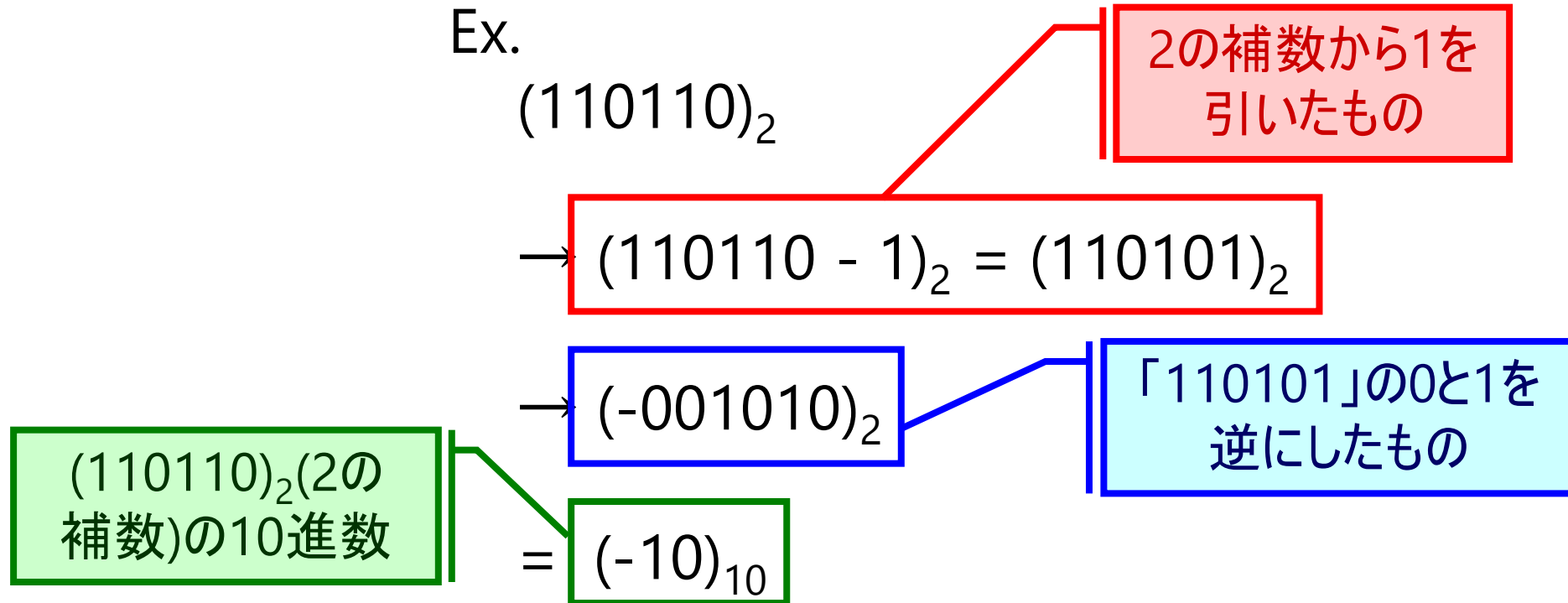
「-3」の2の補数表現



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



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

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

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)

$$\begin{array}{c} 1111101011 \\ \downarrow \\ 0000010100 \end{array}$$

※2の補数→10進数の方法は、10進数→2の補数の逆

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

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

$$\begin{array}{r} 100 \\ - 1 \\ \hline \end{array} \rightarrow \begin{array}{r} 090 \\ - 1 \\ \hline \end{array} \rightarrow \begin{array}{r} 099 \\ - 1 \\ \hline \end{array}$$

引き算の答え: 99

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

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



2(2進数で10)を借りる

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



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



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

やってみよう!

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

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...

0000...00001010

28個の「0」

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...

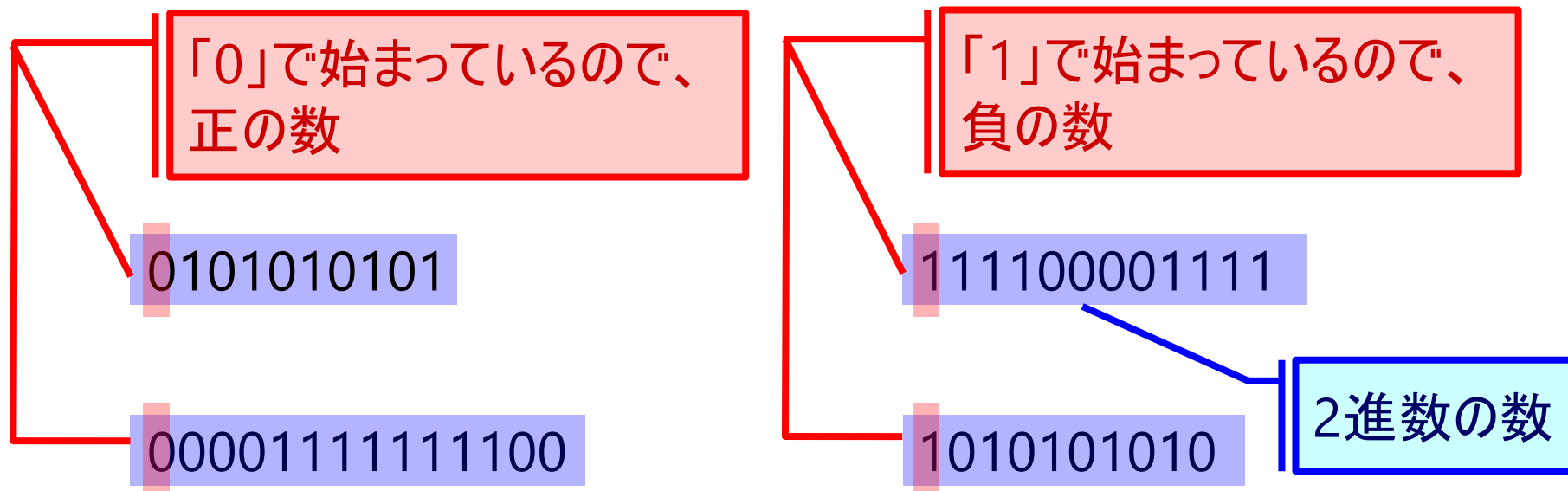
$$\begin{aligned} (-10)_{10} &\rightarrow (10)_{10} \\ &= (0000\dots00001010)_2 \\ &\rightarrow (1111\dots11110101 + 1)_2 = (1111\dots11110110)_2 \end{aligned}$$

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



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)

$$\begin{array}{r} 01110 \\ +) 00101 \\ \hline \end{array}$$

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!

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

桁あふれも考慮すること

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

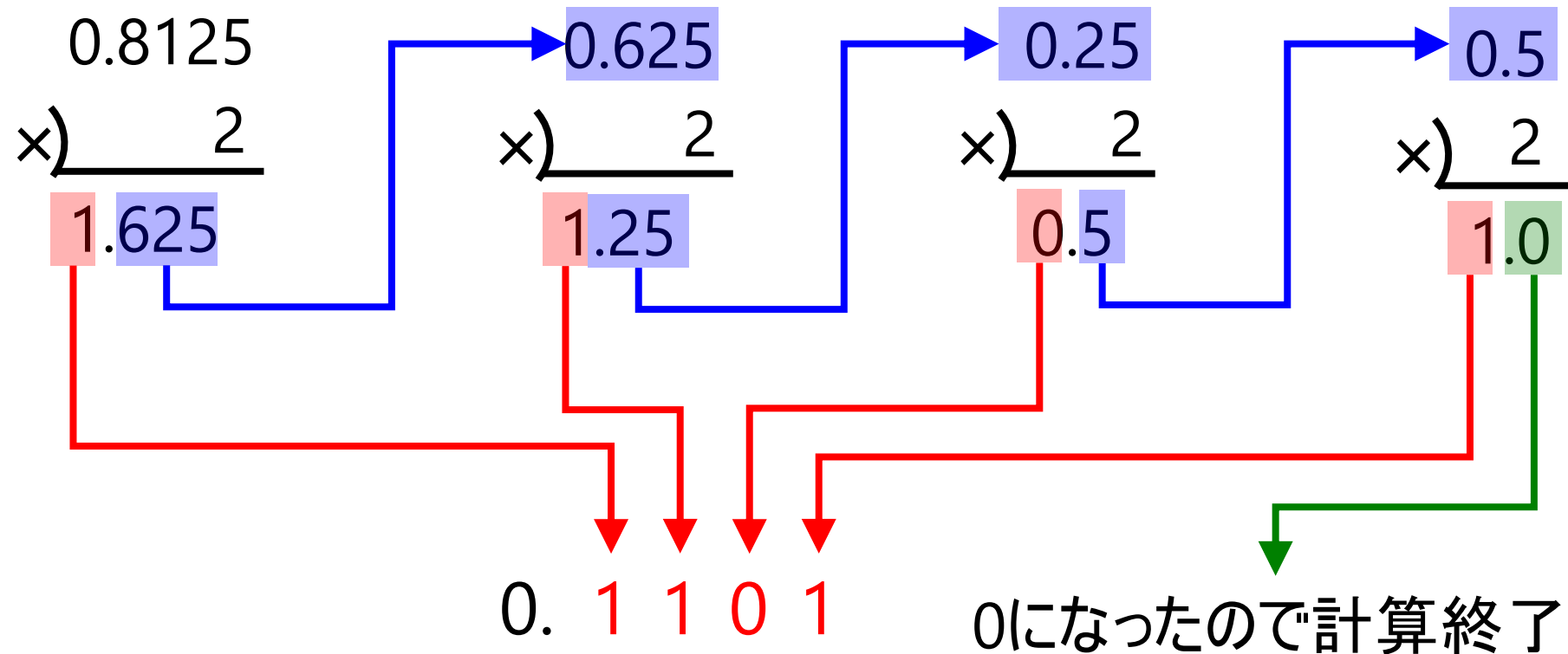
2進数の小数を10進数に直す[1]

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

2進数の小数を10進数に直す[2]

□ 10進数の小数を2進数に直すには?(例1)

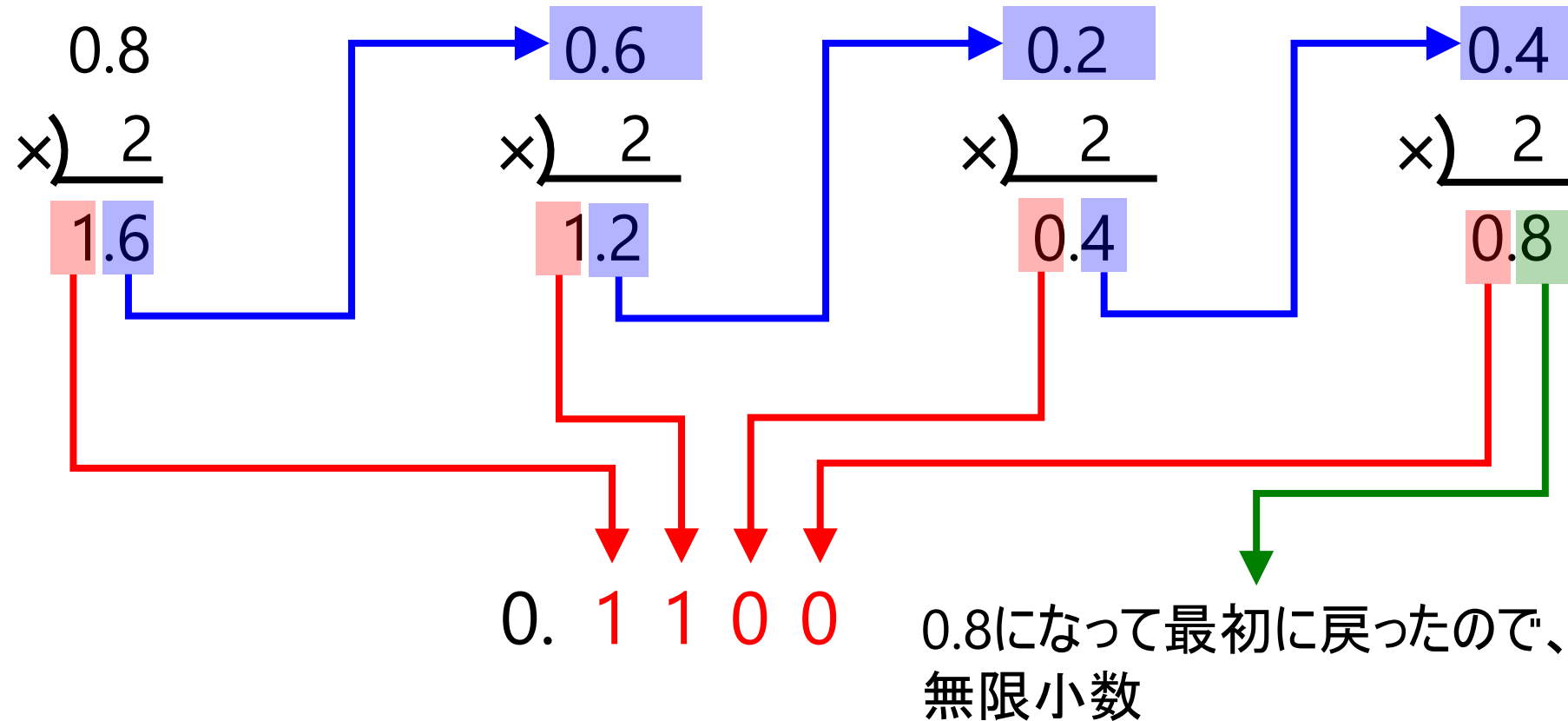
□ 10進数の0.8125を2進数に直す



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

2進数の小数を10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)



$$(3.8)_{10} = (11.110011001100\dots)_2$$

10進数の小数を2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
- $1/2^0, 1/2^1, 1/2^2, \dots$ ができていく

2.

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



3.

左から右に、1, 2, 3, ...と
番号をつける

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

10進数の小数を2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.

$$\begin{array}{cccc} \frac{1}{2^1} & \frac{1}{2^2} & \frac{1}{2^3} & \frac{1}{2^4} \\ 0 & . & 1 & 1 & 0 & 1 \end{array}$$

4.

$$\begin{array}{cccc} \frac{1}{2^1} & \frac{1}{2^2} & \frac{1}{2^3} & \frac{1}{2^4} \\ \times & \times & \times & \times \\ 0 & . & 1 & 1 & 0 & 1 \\ & & & \parallel & & \\ \frac{1}{2^1} & \frac{1}{2^2} & 0 & \frac{1}{2^4} \end{array}$$

5.

$$\begin{array}{c} \boxed{\frac{1}{2^1} \frac{1}{2^2} 0 \frac{1}{2^4}} \\ \text{足し合わせる} \\ \parallel \\ 0.5 + 0.25 + 0.0625 = \boxed{0.8125} \\ \text{1. の整数部分とあわせる} \\ \downarrow \\ 5.8125 \end{array}$$

やってみよう!

- 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)

小数の表現方式

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

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

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

浮動小数点方式[1](p. 10)

□ 小数: $D \times 10^n$ と表現できる

$$\text{Ex. } 0.5 = 5 \times 10^{-1}$$

$$-0.0625 = -6.25 \times 10^{-2}$$

$$0.00000000084 = 8.4 \times 10^{-9}$$

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

浮動小数点方式[2](p. 10)

- 浮動小数点方式:
小数を「 $D \times 10^n$ 」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.

$$20000000000000 = 2 \times 10^{12}$$

$$-42500000000000000000 = -4.25 \times 10^{17}$$

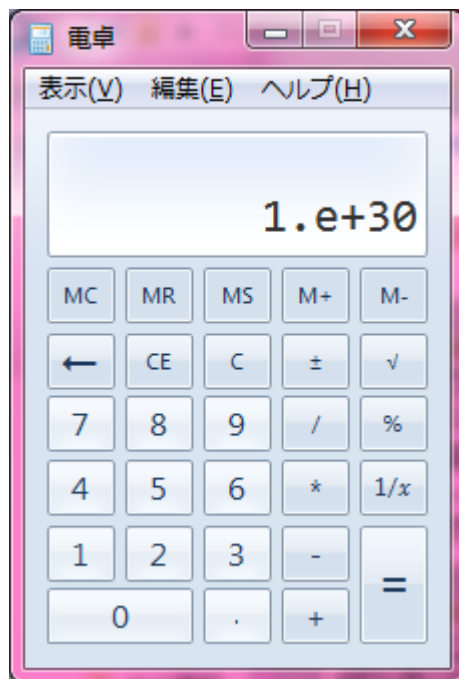
指数部が「+」の数になる



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(オーバフローをなかなかしないモード)で...



Ex. $10000000000000000000 \times 10000000000000000000$
 $= 1.e+30$

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... $1 \div 6$:

コンピュータは「0.16666...666」と考える

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

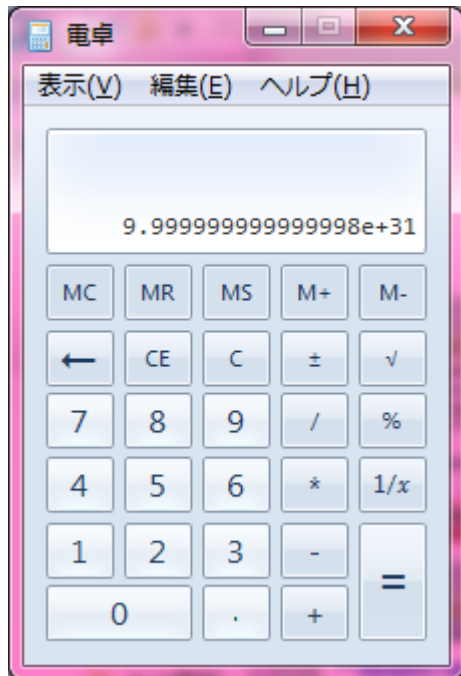


桁落ち

桁落ちの例

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

Ex. 電卓での計算: $9999999999999999999 \times 9999999999999999999$
→ $9.9999999999999998e+31 = 9.9999999999999998 \times 10^{31}$
= $9999999999999999980000...000$ **00**



でも実際に計算すると...

$$\begin{array}{r} 9999999999999999999 \\ \times) 9999999999999999999 \\ \hline \dots 91 \\ +) \dots 91 \\ \hline \dots 01 \end{array}$$

つまり本当の計算では...

$$\begin{array}{l} 9999999999999999999 \times 9999999999999999999 \\ = X.XXXX... \mathbf{01} \end{array}$$

桁落ち

やってみよう!

- 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、 $1 \div 6 \times 100000$ の計算は...

- 「 $1 \div 6$ 」の時点で0.166に桁落ちし、それに100000をかけて、16600
- 1×100000 を7で割ると(計算の順序を変えると)、16666.666
- 本来の $1 \div 6$ に100000をかけると、16666.66666666...

- 
- 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「 $1 \div 6 \times 100000$ 」の計算は、「 1×100000 」をしてから6で割る

やってみよう!

- 0.0000055を浮動小数点方式で表現
- 0.0000000001234を浮動小数点方式で表現
- 456000000000000000を浮動小数点方式で表現
※3つとも仮数部は小数点第2位的小数とすること
- $(10 \div 7) \times 10000$ 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
- $10 \div 7 \times 10000$ 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