

An abstract graphic featuring several curved lines in red, brown, and grey, with small black and grey dots scattered along and around them. The background has faint horizontal orange lines.

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(3)

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(3)

前回の出席問題の解答

- 設問1: 12個の情報を表現するためには、最低何ビットの0と1が必要か

nビットで 2^n 個の情報を表現できるので...

- $2^2 = 4$ → 足りない!
- $2^3 = 8$ → 足りない!
- $2^4 = 16$ → 足りる!

解答: 4ビット

前回の出席問題の解答

- 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

- 4. の問題: 10進数の「135」を10桁の2進数に

$$\begin{array}{r} 2 \overline{) 135} \\ 2 \overline{) 67} \cdots \text{余り: } 1 \\ 2 \overline{) 33} \cdots \text{余り: } 1 \\ 2 \overline{) 16} \cdots \text{余り: } 1 \\ 2 \overline{) 8} \cdots \text{余り: } 0 \\ 2 \overline{) 4} \cdots \text{余り: } 0 \\ 2 \overline{) 2} \cdots \text{余り: } 0 \\ 2 \overline{) 1} \cdots \text{余り: } 0 \\ 0 \cdots \text{余り: } 1 \end{array}$$



$$(135)_{10} = (10000111)_2 \xrightarrow{\text{「10桁」の指定がついているので...}} \text{解答: } (0010000111)_2$$

前回の出席問題の解答

- 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

2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
×	×	×	×	×	×	×	×	×	×	×	×
0	0	1	0	1	0	1	0	1	0	1	0

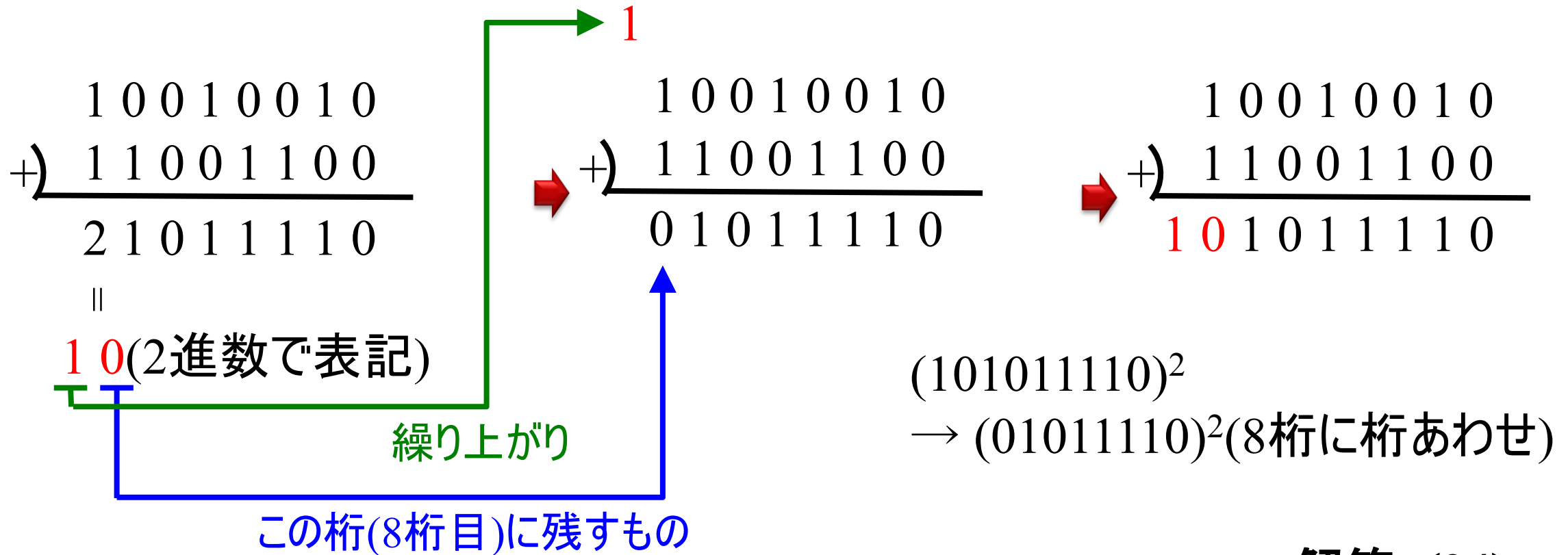


$$\begin{aligned} &0 + 0 + 2^9 + 0 + 2^7 + 0 + 2^5 + 0 + 2^3 + 0 + 2^1 + 0 \\ &= 512 + 128 + 32 + 8 + 2 \\ &= 682 \end{aligned}$$

解答: $(682)_{10}$

前回の出席問題の解答

- 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)



解答: $(94)_{10}$



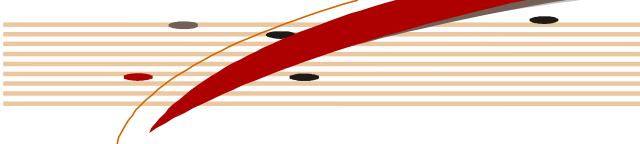
前回の質問の回答

- 2^n : 2をn回かけた数 = nが1つ増えるごとに、2倍になっていく
 - 2^{n-1} : 2をn-1回かけた数であり、かつ 2^n を2で割った数

...

- 2^3 : 2^4 を2で割った数
- 2^2 : 2^3 を2で割った数
- 2^1 : 2^2 を2で割った数
- 2^0 : 2^1 を2で割った数

$$20 = 2^1 \text{を} 2 \text{で割った数} = 2^1 \div 2 = 2 \div 2 = 1$$


$$(1+1+1)_2$$

- $(1)_2 = (1)_{10}$ なので... $(1+1+1)_2$ の計算は、 $(1+1+1)_{10}$ の計算をして、計算結果を2進数に直したもの

$$(1+1+1)_{10} = (3)_{10} = (11)_2$$

割り算でオーバーフロー

- オーバーフロー (over flow)

- over: 余った, 超えた, etc.
- flow: 流れる, 流す, 流れ, etc.

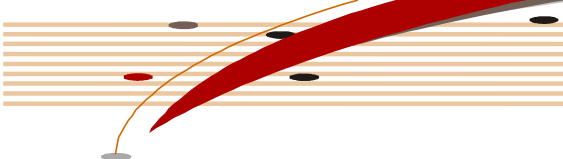
 多くなりすぎてあふれた、という感じの意味

整数ベースだと...

- かけ算: 数が大きくなっていく
- 割り算: 数が小さくなっていく



- かけ算にはオーバーフローがある
- 割り算にはオーバーフローはない
 - ✓ 小数点以下の部分が表現しきれなくなる現象はある
(桁落ち, 後日の内容)



Question!



前回の復習

2進数を10進数に変換

● 単純に...

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

1.

2	2	2	2	2	2
1	1	1	0	1	0



2.

右から左に、0, 1, 2, ...と番号をつける

⁵	⁴	³	²	¹	⁰
2	2	2	2	2	2
1	1	1	0	1	0

※ 2^n : 2をn回かけ算する

➤ Ex. 2^3 : $2 \times 2 \times 2 = 8$

2進数を10進数に変換

● 単純に...

3. 各桁の上の「 2^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.

2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0



3.

2^5	2^4	2^3	2^2	2^1	2^0
×	×	×	×	×	×
1	1	1	0	1	0

||

2^5	2^4	2^3	0	2^1	0
-------	-------	-------	---	-------	---



4.

2^5	2^4	2^3	0	2^1	0
-------	-------	-------	---	-------	---

足し合わせる

||

$$2^5 + 2^4 + 2^3 + 0 + 2^1 + 0 = 58$$

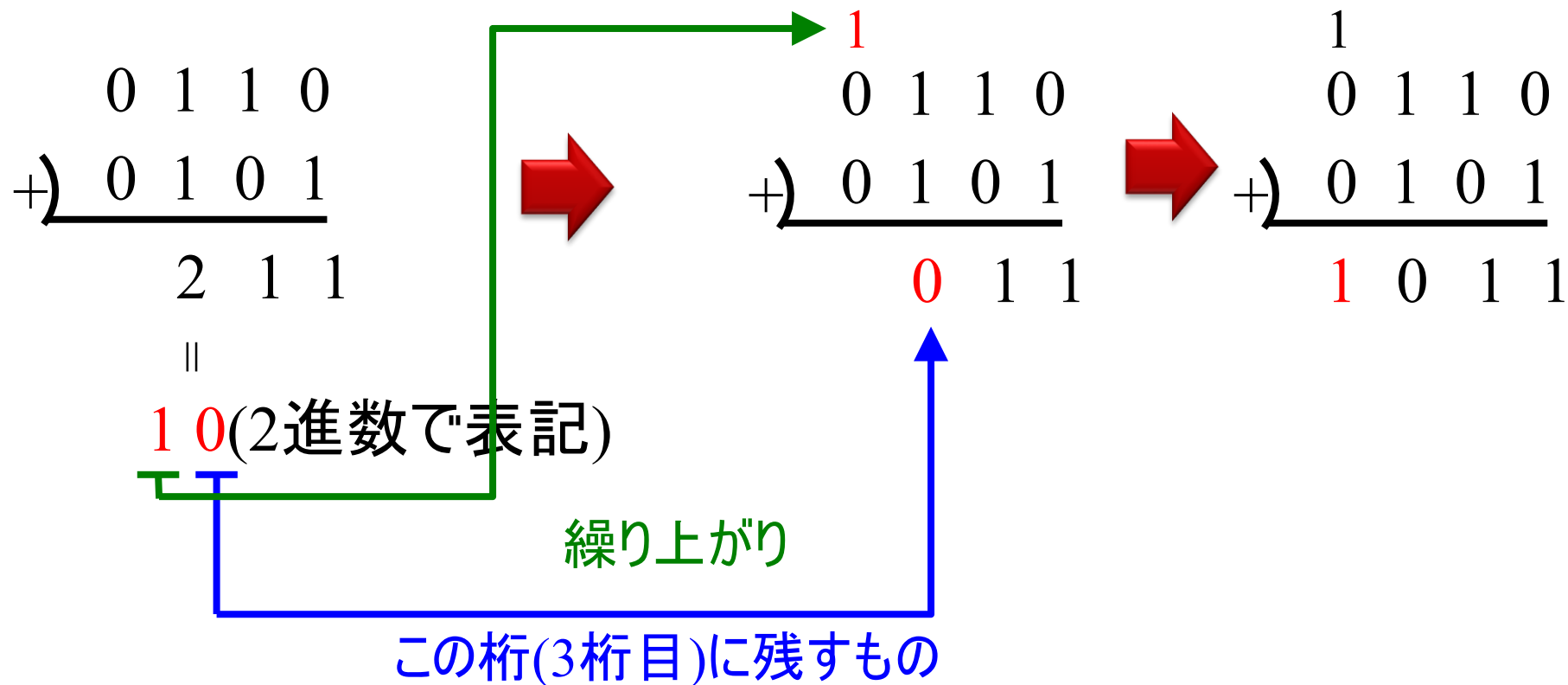
足し算をする方法[1](p. 6)

- 10進数での1桁の足し算
 - たくさん($10 \times 10 = 100$)のパターンが存在
 - $1+1, 1+2, 1+3, \dots 2+1, 2+2, 2+3, \dots \dots 8+6$ (繰り上がり1), $8+7$ (繰り上がり1),
... ..
- 2進数での1桁の足し算
 - 4通り
 - 足した結果が2になると繰り上がり1(2進数では10進数の2を「10」と表すため)
 - $0+0, 0+1, 1+0, 1+1$ (繰り上がり1)

足し算をする方法[2](p. 6)

- 基本的な2進数の足し算の方法は10進数と同じ

0110(10進数で6)と0101(10進数で5)の足し算



➡ 計算結果: 1011(10進数で11)

桁あふれ(オーバースロー)[1]

- コンピュータでは数を表すビット数(2進数の桁数)は固定されている
 - 計算の結果、決まった桁数を超えると...?

Ex. 数を4ビット(4桁)で表す場合:

1110(10進数で14)と0101(10進数で5)の足し算

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

桁あふれ(オーバーフロー)[2]

Ex. 数を4ビット(4桁)で表す場合

1110(10進数で14)と0101(10進数で5)の足し算

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

↑ 5ビット目(5桁目, 決められた桁数を越えてしまった部分)

➡ 決められた桁数を越える = 2進数の各桁を入れる箱の数足りなくなる

➡ 決められた桁数を越えた部分は無視される(捨てられてしまう)

~~1~~ 0 0 1 1

↑ 無視される(捨てられる)

➡ 計算結果: 0011(10進数で3)

計算結果が決められた桁数を越えること:
桁あふれ(オーバーフロー)

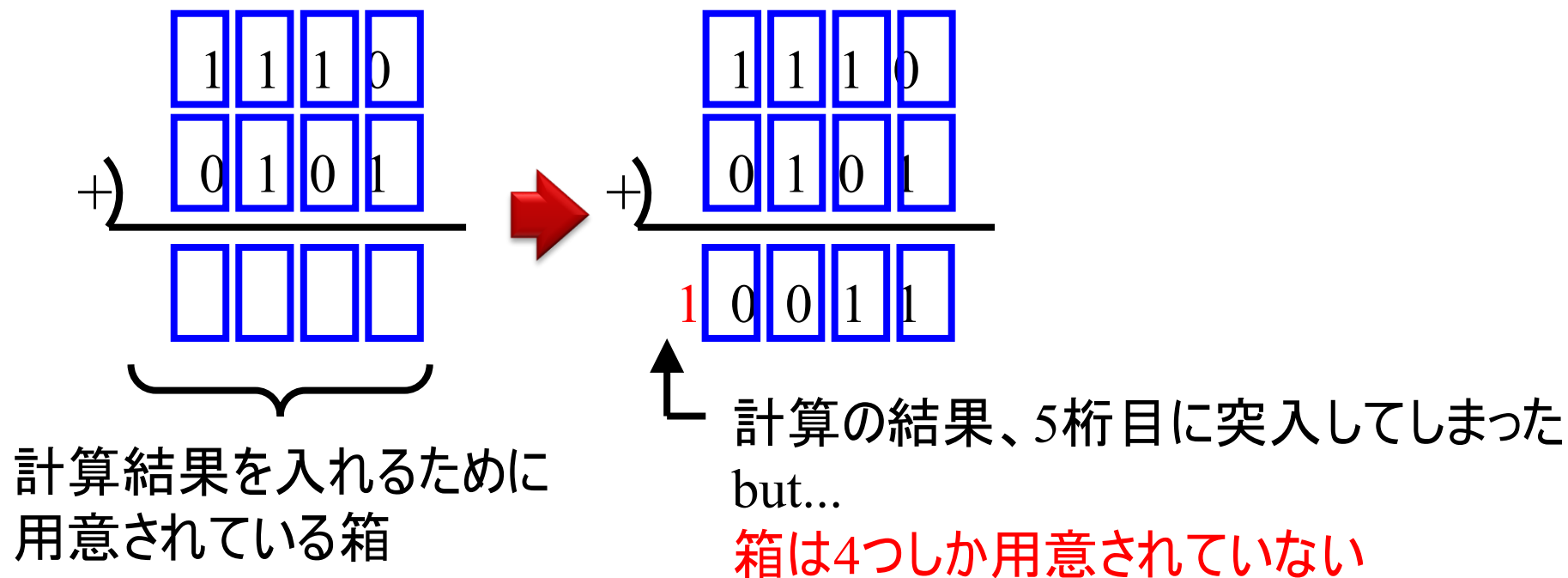
桁あふれ(オーバーフロー)の扱い[1]

- コンピュータでは、2進数の各桁を、1つずつ箱に入れて扱っている、というイメージ
 - 各桁を入れる箱の数に限りがある
 - Ex. 数を4ビットで表す = 数を4桁で表す(2進数の各桁を入れる箱の数が4個)
 - **どのような計算をしたとしても、箱の数は変更されない**
 - 計算結果を入れるための箱は、計算に使う数と同じ個数しか用意されない
 - Ex. 数を4ビットで表すときに、 $(1110 + 0101)_2$ の計算結果も4ビットでしか表現できない(箱は4個しかない)

本来の計算結果(人間が自分の手で行った計算結果)とコンピュータが行った計算結果(Ex. 電卓などの計算結果)が違ってしまいう現象

桁あふれ(オーバーフロー)の扱い[2]

- 2進数の各桁を入れる箱は、小さい桁(右の桁)の分から用意される



5桁目は無視されるので計算結果は $(0011)_2$

計算結果の解答

- 試験問題などで、「オーバーフローを考慮する」と問題文に書いてある場合 (書いてなければ考えなくてOK)
 - 問題で求められた計算をして結果を出す(2進数で計算するか、または10進数で計算して2進数になおすか、どちらか)
 - 問題文の桁数の指定をチェックし、1. の計算結果の桁数を合わせる
 - (必要であれば)2. の結果を10進数になおす

Ex. 2進数4桁の数1110+0101の計算結果を10進数で答えなさい。
ただし、オーバーフローを考慮すること

1. 計算をする

✓ $1110 + 0101 = 10011$

2. 問題文に「4桁」という指定がある

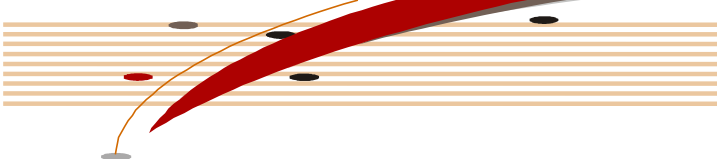
✓ 1. の計算結果は10011(5桁) → 0011 (4桁にあわせた)

3. 10進数に直す

✓ $(0011)_2 = (3)_{10}$



解答: 3



コンピュータでの情報量

バイト[1](p. 8)

- コンピュータでの情報量:
情報を表現する「0」と「1」の数 = **ビット**

コンピュータの世界では、「0」と「1」を8個単位で扱うことが多い

Ex.:

半角英数の文字: 8個の「0」と「1」で構成 (8個 × 1)

全角の文字: 16個の「0」と「1」で構成 (8個 × 2)

画像などの色: 24個の「0」と「1」で構成 (8個 × 3)

8ビットで1つの単位: **バイト(byte)**

バイト[2](p. 8)

- 1バイト(byte) = 8ビット(bit)
 - 半角英数1文字(8ビット): 1バイト
 - 全角1文字(16ビット): 2バイト
 - 画像などの色1つ(24ビット): 3バイト

バイト[3](p. 8)

- 現実世界: 1000で1つの単位
 - 1000: 1K (1000m = 1Km)
- コンピュータの世界では 2^{10} で1つの単位
 - 1Kbyte(キロバイト, KB): 1024byte
 - 1Mbyte(メガバイト, MB): 1024Kbyte
 - 1Gbyte(ギガバイト, GB): 1024Mbyte
 - 1Tbyte(テラバイト, TB): 1024Gbyte

便宜上、1KB = 1000byte, 1MB = 1000KB, etc. とすることもある



8進数と16進数

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!

8進数と16進数[3]

Ex.

アルファベットの「N」

2進数: 01001110

8進数: 116

16進数: 4E

日本語の「ん」:

2進数: 1010010011110011

8進数: 51163

16進数: 5273

赤色:

2進数: 111111110000000000000000

8進数: 77600000

16進数: FF0000

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 →

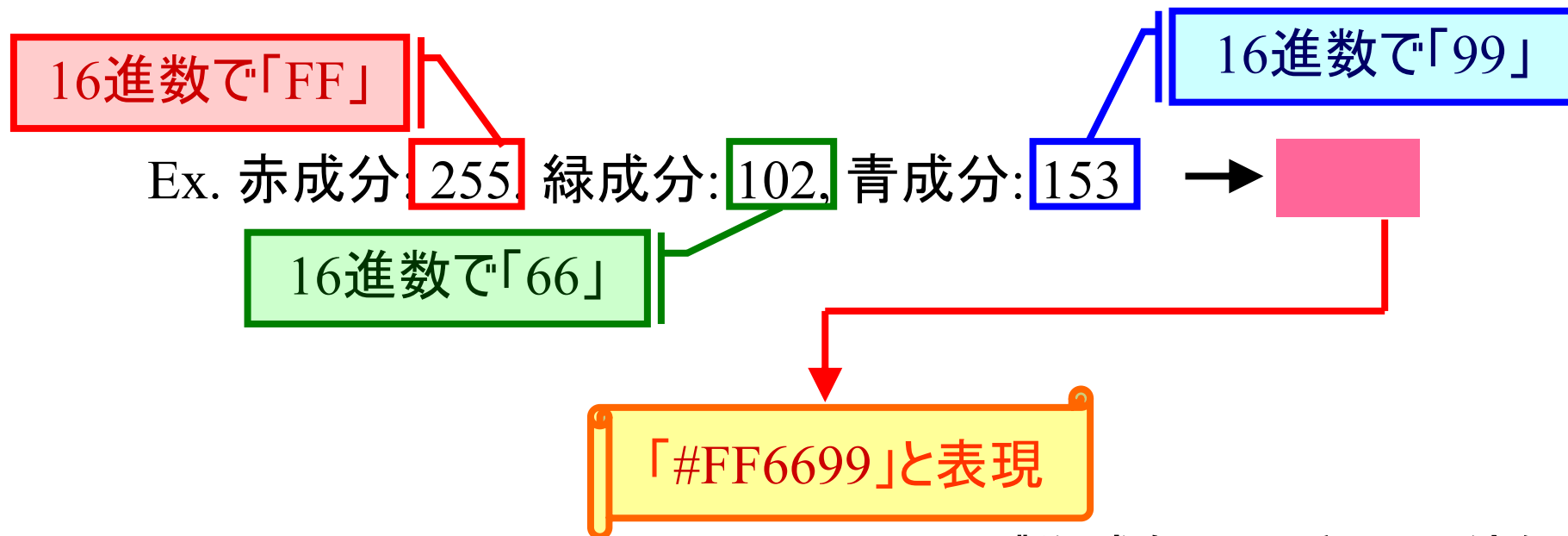


0～255を16進数で表すと0～FFで、ちょうど2桁で表せる

※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

やってみよう!

- 授業の資料のページに行って...

1. 「PECO STEP」のページに行く
2. 「カラーコード」の入力欄に、16進数の色の名前を入力する
 - 色の名前は適当で良い(1文字目が「#」、2文字目以降が1～9とa～fの文字)
 - 色の名前のアルファベットは大文字でも小文字でも良い
3. 下側の領域に、その色が表示される
 - R・G・Bの欄に、赤・緑・青の濃淡の10進数
 - 「色の見本」の欄に、2. で入力した色

※必要な色を探すときは、「色見本」などのキーワードで検索すると良い

10進数を16進数に変換

● 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.

$$\begin{array}{r} 16 \overline{) 10\text{進数の数}} \\ 16 \overline{) \quad \quad \quad \text{商1} \cdots \text{余り1}} \\ 16 \overline{) \quad \quad \quad \text{商2} \cdots \text{余り2}} \\ \quad \cdot \\ \quad \cdot \\ 16 \overline{) \quad \quad \quad \cdot} \\ \quad \quad \quad \text{商n} \cdots \text{余りn} \end{array}$$

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_n
余り_{n-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(例)

10進数の255を16進数に変換

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



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



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

10進数の2000を16進数に変換

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



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



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

→ 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[3]

1. 10進数の「240」を16進数に
 2. 10進数の「3000」を16進数に
 3. 10進数の「50000」を16進数に
 4. 16進数の「64」を10進数に
 5. 16進数の「FA0」を10進数に
 6. 16進数の「4E20」を10進数に
 7. 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)



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

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

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.

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n
余りn-1
.
.
.
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



右から左に、0, 1, 2, ...と番号をつける

2.

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
足し合わせる			
$3 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 0 = 2000$			

※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



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	符号ビット
-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	

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)

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

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

-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

Ex.

$(110110)_2$

2の補数から1を
引いたもの

→ $(110110 - 1)_2 = (110101)_2$

→ $(-001010)_2$

「110101」の0と1を
逆にしたもの

$(110110)_2$ (2の
補数)の10進数

= $(-10)_{10}$

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

$$\begin{array}{r} 100 \\ - 1 \\ \hline \end{array} \rightarrow \begin{array}{r} 0 \text{ } 10 \text{ } 0 \\ - 1 \\ \hline \end{array} \rightarrow \begin{array}{r} 0 \text{ } 9 \text{ } 10 \\ - 1 \\ \hline 0 \text{ } 9 \text{ } 9 \end{array}$$

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

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



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



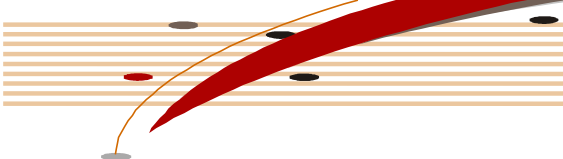
引き算の答え: 011

※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

とを考えている

28個の「0」

※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...

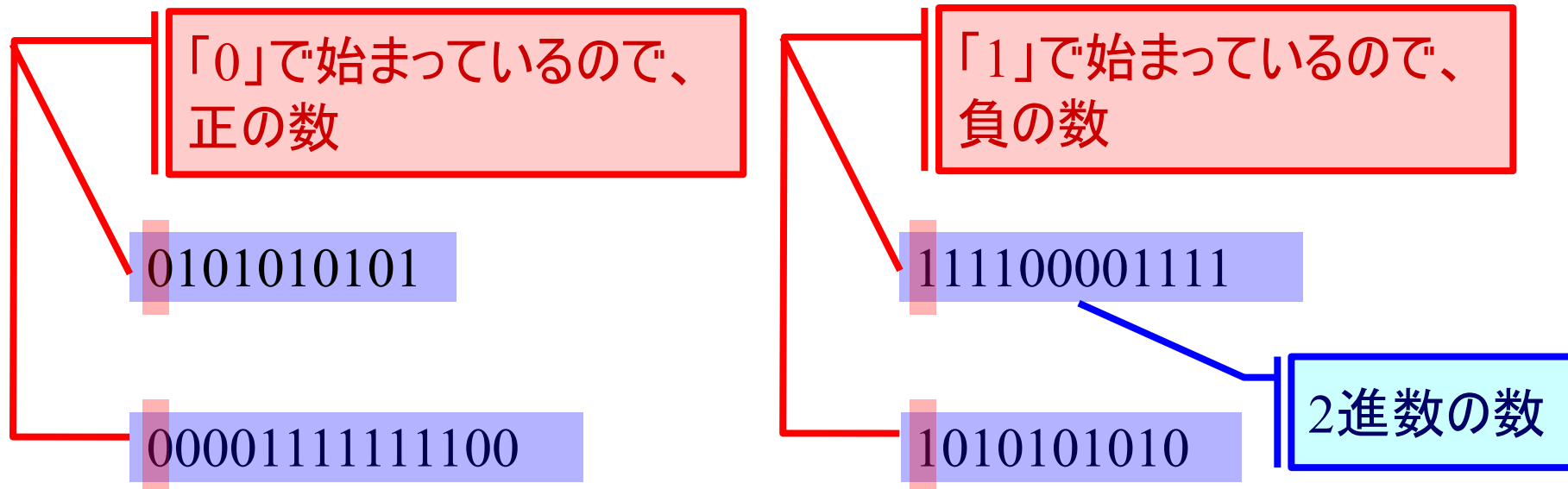
$$\begin{aligned} (-10)_{10} &\rightarrow (10)_{10} \\ &= (0000\dots00001010)_2 \\ &\rightarrow (1111\dots11110101 + 1)_2 = (1111\dots11110110)_2 \end{aligned}$$

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)

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

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)

4桁の2進数の計算結果: $(100110)_2$  桁数を超えた部分 = 削除  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

4桁の2進数の計算結果: $(0101)_2$  正の数と判断  計算結果: $(5)_{10}$

4桁の2進数の計算結果: $(1010)_2$  負の数と判断  計算結果: $(-6)_{10}$