コンピュータ・サイエンス1

第4回
コンピュータでの情報の扱い

人間科学科コミュニケーション専攻
白銀 純子

第4回の内容

- コンピュータの構成(続き)

コンピュータでの情報の扱い方

コンピュータの基本構成

- コンピュータは電気回路で構成
  - 電気回路: 電気が通ることで動作する様々な部品(電気素子)を電気を通す線で結んだもの
  - CPUなど、ほとんどの部品は電気回路で構成
- コンピュータは、電気回路に電気が通ることで様々な命令を処理
  - ある瞬間に、電気回路中のどの線に電気が通ったか・通らなかったかで全ての物事を処理
    - 回路中にたくさんスイッチがあり、ある瞬間でどのスイッチがONでどのスイッチがOFFになっていたか、のようなイメージ
  - 人間がコンピュータの動作を考えるとき、電気が通った線を1、通らなかった線を0のように数で表現

コンピュータでの情報の扱い方[1](p. 2)

- コンピュータが扱える情報は「0」と「1」のみ
  - ある瞬間で電気が通らなかった線と通った線を0と1として扱って考える
- 大量の「0」と「1」を組み合わせて情報を表現
  - Ex. 1文字1文字は、0と1の並びで表現
  - それぞれの物事は、決まった個数の0と1で表現
    - 半角英数文字: 8個
    - 全角文字: 16個
    - etc.

コンピュータでの情報の扱い方[2](p. 2)

- 数値は0と1の並びで表現
  - 数値を表す0と1の個数は、扱い方によっていくつか種類が存在

例えば…

「50」: 110010

「100」: 1100100

コンピュータでの情報の扱い方[3](p. 2)

- 1文字1文字は0と1の並びで表現

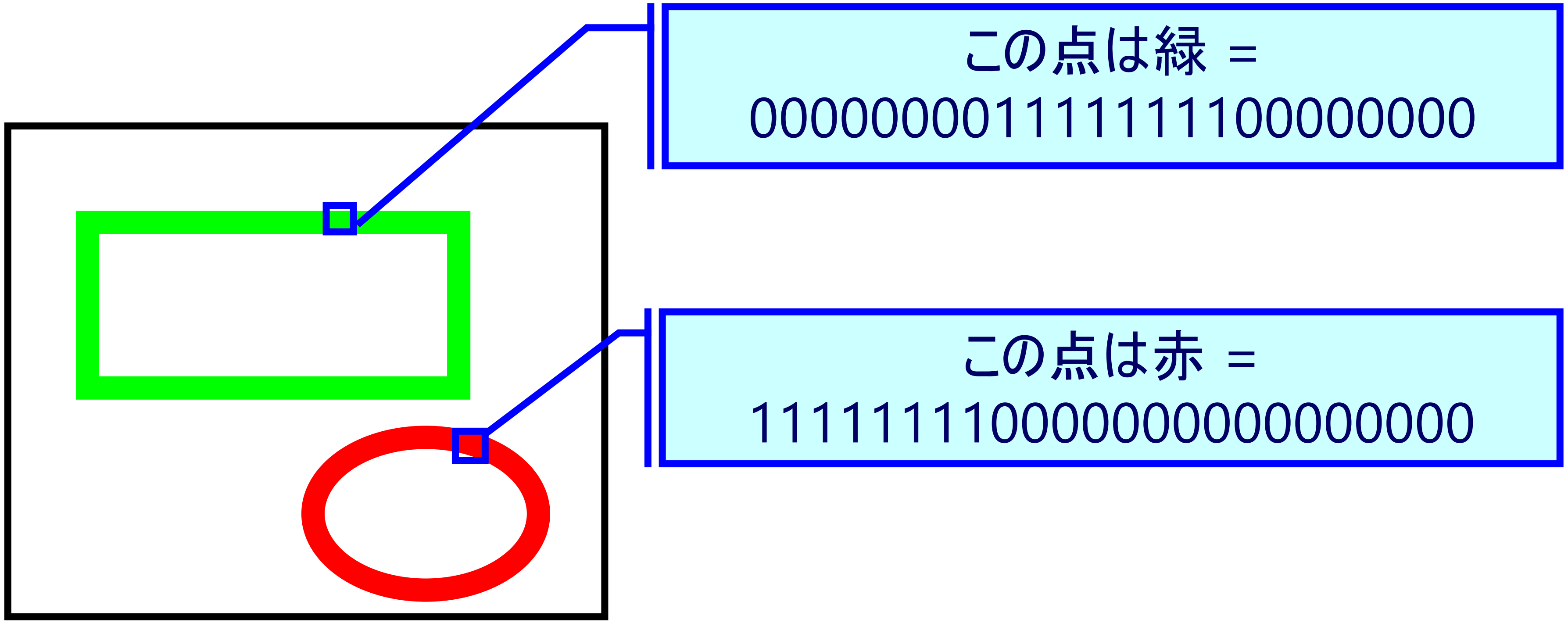
例えば…

アルファベットの「N」: $\underbrace{01001110}_{\text{8個の0と1}}$

日本語の「ん」: $\underbrace{1010010011110011}_{\text{16個の0と1}}$

コンピュータでの情報の扱い方[4](p. 2)

- 画像は、コンピュータにとっては点の集まり
  - 1つ1つの点が何色かで絵を表現



コンピュータでの情報の扱い方[5](p. 2)

- コンピュータの利用以前: フィルムやテープ
  - 情報をそのままの形で記録
  - 情報の種類ごとに別個の機器や記録媒体が必要
    - 動画: 映画フィルムやビデオテープ
    - 音声: レコードや録音テープ
- コンピュータ
  - 様々な情報を「0」と「1」の形(ビット)に加工して記録
    - 数, 文字, 画像, 音声, etc.は、全てそれぞれの方法で0と1の並びに変えてから記録
    - 情報をどのように加工・保存・伝送することも簡単に可能

➢ 同じコンピュータで、様々な情報を扱えるようになった
➢ 個別の機器ごとではできなかった、多様できめ細かな処理が可能になった

ビット[1](p. 4)

- コンピュータで扱う情報は「0」と「1」の2文字
  - 2文字では、2種類の情報しか表現できない

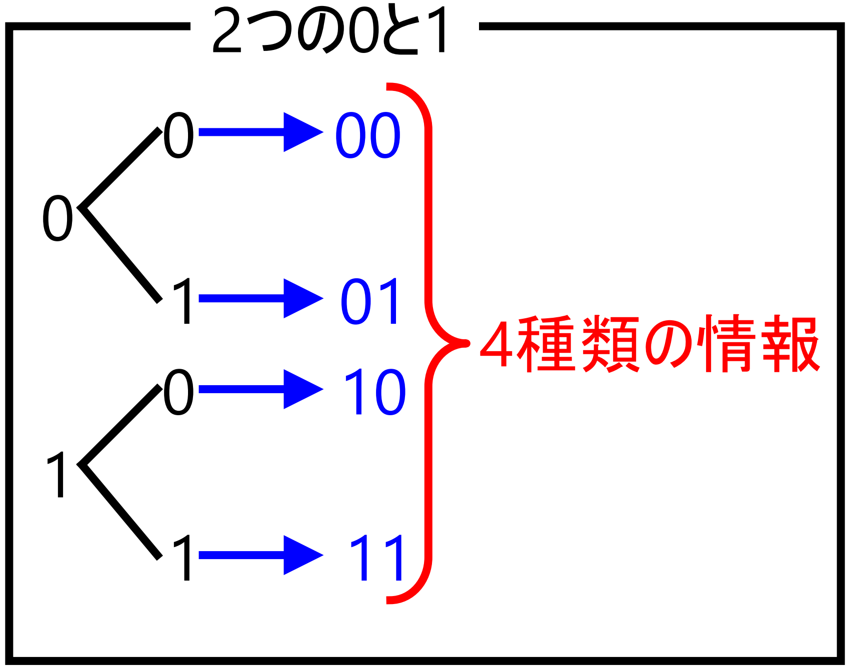
↓

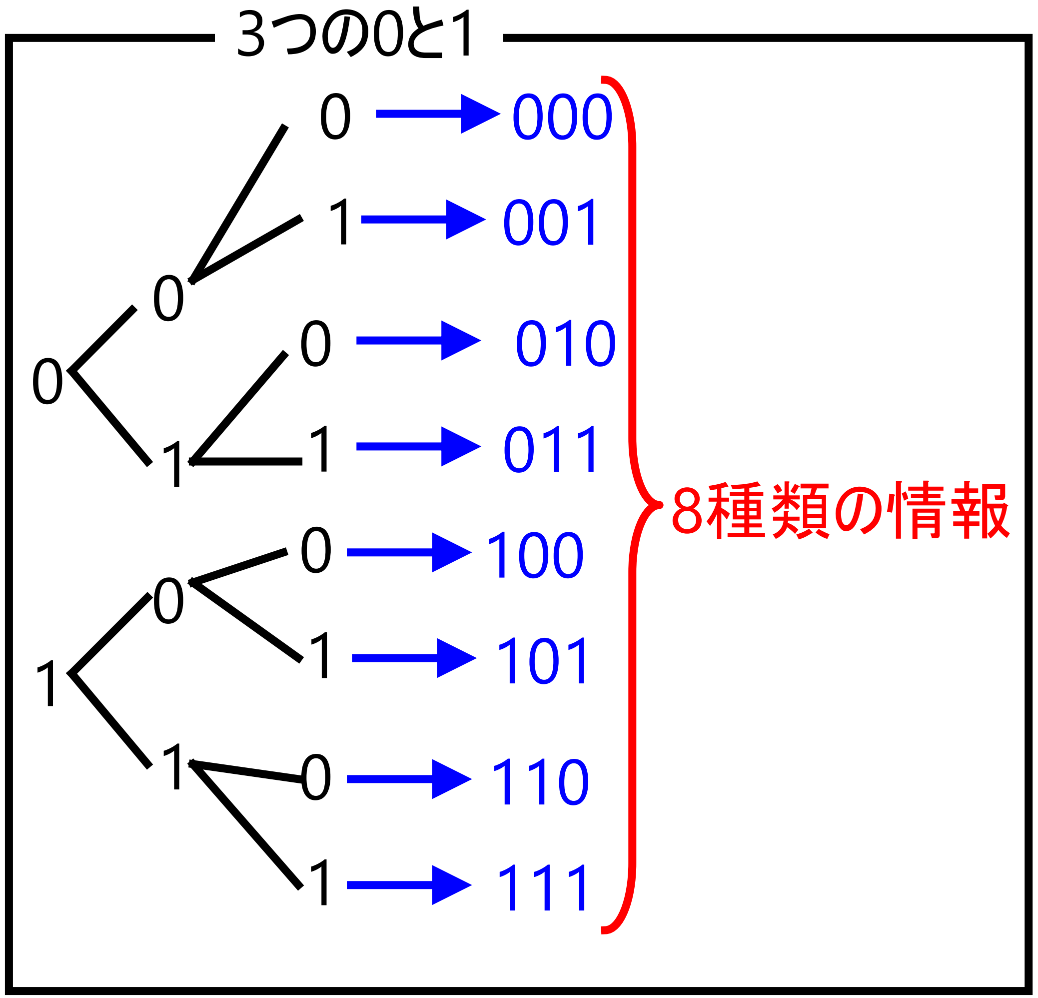
たくさんの種類の情報を扱うには…

大量の0と1を組み合わせる

ビット[2](p. 4)

- 0と1の組み合わせ





ビット[3](p. 4)

- 2個の「1」と「0」 → 4種類の情報
- 3個の「1」と「0」 → 8種類の情報
- n個の「1」と「0」→ 2^n種類の情報

↓

組み合わせる「0」と「1」の数が多くなれば、
表現できる情報の種類も増える

ビット[4](p. 4)

- ビット: 情報を表現する1つ1つの「0」と「1」
  - コンピュータでの情報量の基本単位
  - 情報を表現する「0」と「1」の個数
- ビット列: 情報を表現する1つ1つの「0」と「1」の並び

例えば…

「50」: 110010 → 6 ビット

「100」: 1100100 → 7 ビット

アルファベットの「N」: 01001110 → 8 ビット

日本語の「ん」: 1010010011110011 → 16 ビット

2進数[1](p. 4)

- n進数: 数をn個の文字で表す方法
  - 10進数: 数を10個の文字で表す方法(普段使っている数の表現方法)
    - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9の10個の文字
  - 2進数: 数を2個の文字で表す方法
    - 0, 1の2個の文字

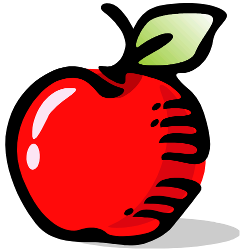
コンピュータ: 「0」と「1」で全ての情報を表現

「2進数で情報を表現している」、と言える

2進数[2](p. 4)

- 2進数
  - 「0」と「1」だけで全ての数を表現

10進数の「50」 = 2進数で「110010」



を

「りんご」と表現する

「apple」と表現する

表現方法が違うだけ

2進数は、10進数での表現を違う表現にしただけ
(数の量などが変わるわけではない)

2進数[3](p. 4)

- 2進数
  - 「0」と「1」だけで全ての数を表現
  - 「2」で繰り上がる、という考え方
    - 10進数: 10で繰り上がる

10進数	2進数
0	00
1	01
2	10
3	11

10進数	2進数
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

繰り上がり

2進数[4](p. 4)

- n進数を区別して数を表記する場合: $(数)_n$と表記
  - 10進数: $(数)_{10}$
  - 2進数: $(数)_2$

$(100)_{10}$: 10進数の百

$(100)_2$: 2進数の100(10進数で4)

10進数を2進数に変換

- 10進数の数は、2進数の表現に直すことができる

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

2) 10進数の数
2) 商1・・・余り1
2) 商2・・・余り2
・
・
・
2) 商n-1・・・余りn-1
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2進数

10進数を2進数に変換(例)

10進数の13を2進数に変換

2) 13
2) 6・・・余り: 1
2) 3・・・余り: 0
2) 1・・・余り: 1
0・・・余り: 1

$(13)_{10} = (1101)_2$

10進数の50を2進数に変換

2) 50
2) 25・・・余り: 0
2) 12・・・余り: 1
2) 6・・・余り: 0
2) 3・・・余り: 0
2) 1・・・余り: 1
0・・・余り: 1

$(50)_{10} = (110010)_2$

※矢印の方向に余りを並べる

2進数の桁数

- 10進数: 普通、「2」や「200」などの数を「02」や「00200」とは表現しない
- 2進数: 「xx桁の2進数」は、2進数の桁数が「xx」に足りなければ、
  2進数の前に「0」をつけて表す
  - Ex. 10進数の「2」を6桁の2進数で表せ → 000010

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, …ができていく

右から左に、0, 1, 2, …と番号をつける

1.

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

2.

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

$$\shortparallel$$

$$2^5 \quad 2^4 \quad 2^3 \quad 0 \quad 2^1 \quad 0$$

4.

$$\boxed{2^5 \quad 2^4 \quad 2^3 \quad 0 \quad 2^1 \quad 0}$$

足し合わせる

$$\shortparallel$$

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

2進数を10進数に変換[4]

- 2^0～2^{10}の数は覚えておくと便利

2のべき乗	10進数	2進数
2^0	1	1
2^1	2	10
2^2	4	100
2^3	8	1000
2^4	16	10000
2^5	32	100000
2^6	64	1000000
2^7	128	10000000
2^8	256	100000000
2^9	512	1000000000
2^{10}	1024	10000000000

やってみよう![1]

- 10進数の「25」を2進数に
- 10進数の「500」を2進数に
- 10進数の「255」を2進数に
- 10進数の「135」を10桁の2進数に
- 10進数の「200」を12桁の2進数に
- 2進数の「001010101010」を10進数に
- 2進数の「01111000010」を10進数に
- 2進数の「0010000111001」を10進数に

ビット数も数えてみよう!

※計算方法は、自分でやりやすい方法があれば、それを使って良い

2進数での足し算

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(10×10=100)のパターンが存在
  - 1+1, 1+2, 1+3, … 2+1, 2+2, 2+3, … … 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

```
   0 1 1 0
+) 0 1 0 1
----------
     2 1 1
     =
   1 0(2進数で表記)
```

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

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

繰り上がり

この桁(3桁目)に残すもの

計算結果: 1011(10進数で11)

桁あふれ(オーバーフロー)[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

$$
\begin{array}{rccccc}
 & & 1 & 1 & 1 & 0 \\
+) & & 0 & 1 & 0 & 1 \\
\hline
 & \color{red}{1} & 0 & 0 & 1 & 1
\end{array}
$$

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た部分は無視される(捨てられてしまう)

$\cancel{1}\ 0\ 0\ 1\ 1$

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超えること:
桁あふれ(オーバーフロー)

桁あふれ(オーバーフロー)[3]

- コンピュータの世界では、数を表現する2進数の桁数は常に固定
  - 計算内容などによる変化はなし
  - 通常は、32桁または64桁で数を表現
    - 授業のスライドは、そんなに長く書けないので、小さい桁数で表現

- 桁あふれ(オーバーフロー)が起こると…
  本来の計算結果とコンピュータでの結果が違ってしまう
  - Ex. 4桁の2進数1110と1010の計算結果: 10011
    - 10011は5桁になってしまったので、0011という4桁で表現
      →本来の計算結果とは違う結果

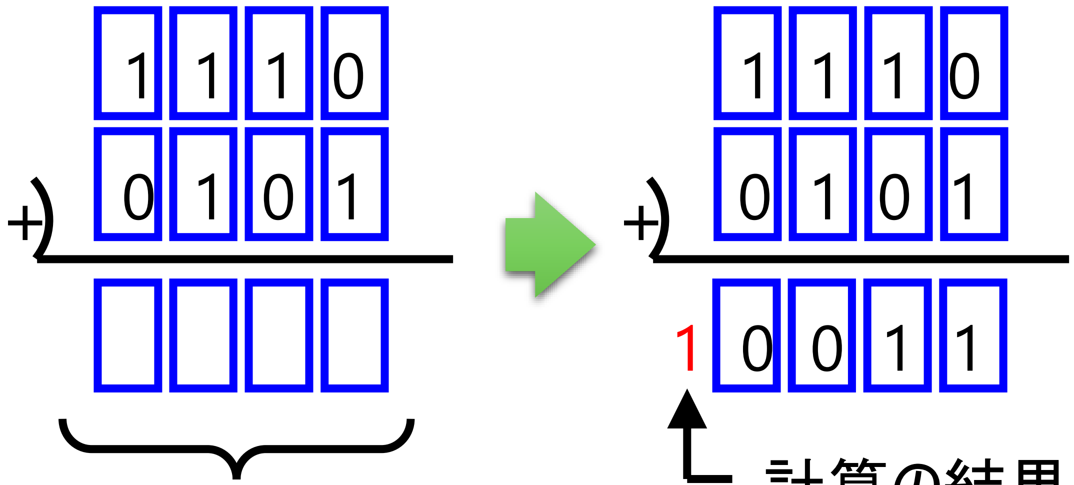
桁あふれ(オーバーフロー)の扱い[1]

- 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
    - 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



計算結果を入れるために
用意されている箱

計算の結果、5桁目に突入してしまった
but…
箱は4つしか用意されていない

5桁目は無視されるので計算結果は$(0011)_2$

やってみよう![2]

- 8ビットの数の足し算をし、結果を10進数で計算すること(桁あふれも考えて結果を計算すること)
  - 10101010 + 01010101
  - 11110000 + 01000000
  - 10010010 + 11001100
- 2進数10110を3倍した数を計算すること(2009年度ITパスポート春期試験問題)

2進数の2^n倍と$1/2^n$

2進数×2^n

- 「2進数×2^n」の計算は簡単
  - 2進数の一番右に、n個分「0」をつけるだけ
    - 2^nは2進数で表現すると、$(10)_2$をn回掛け算した数だから

Ex:

$(101101)_2 \times (8)_{10}$
$=(101101)_2 \times (2^3)_{10}$
$=(101101)_2 \times (1000)_2$
$=101101\underline{000}$

もとの2進数の一番右に3個「0」がついているだけ

2進数×2^n

- かけ算する2進数を小数で表現したとき、小数以下に「0」が並んでいる
  - 2^nを2進数にかけると、小数以下に並んでいた「0」が出てきて、もとの数がn個分左にずれる、というイメージ

「左にnビットシフトする」と呼ぶ

Ex:

$(101101)_2 \times (8)_{10}$

$=(101101)_2 \times (2^3)_{10}$

101101.000000000.....

1011010.00000000.....

10110100.0000000.....

101101000.000000.....

101101を左に3ビットシフトした数(小数点が移動しているだけ)

2進数÷2^n

- 「2進数÷2^n」の計算も簡単
  - 2進数の右からn桁分を小数部分にするだけ
  - 「2進数÷2^n」は2進数で表現すると、2進数を$(10)_2$でn回割り算した数だから

Ex:
$(111101)_2 \div (8)_{10}$
$=(111101)_2 \div (2^3)_{10}$
$=(111101)_2 \div (1000)_2$
$=111.\underline{101}$

もとの2進数右から3桁分を小数部分にしただけ
(小数点が移動しているだけ)

2進数÷2^n

- 2^nで2進数を割ると、$\boxed{\text{その2進数がn個分右にずれる}}$、というイメージ

「右にnビットシフトする」と呼ぶ

Ex:

$(111101)_2 \times (8)_{10}$

$=(111101)_2 \times (2^3)_{10}$

111101

11110 .1

1111 .01

111 .101

111101を右に3ビットシフトした数

シフト算でもオーバーフロー

- オーバーフローが起こるのは…
  - 足し算
  - かけ算(左にシフトする計算)

かけ算の場合… Ex. 4ビットの数: $(1011)_2 \times (8)_{10}$

$$(1011)_2 \times (8)_{10} = (1011)_2 \times (2^3)_{10}$$
$$= (1011)_2 \times (1000)_2$$
$$= (1011000)_2$$

7ビット(7桁)になってしまった

but… 各桁を入れる箱は、小さい桁から4桁分

↓

大きい桁(左の桁)から3桁分が無視されるので、計算結果は$(1000)_2$

やってみよう![3]

- 10010を左に4ビットシフトした数
- 11001を左に7ビットシフトした数
- 1110101を左に2ビットシフトした数
- 10100000を右に3ビットシフトした数
- 11010100000を右に5ビットシフトした数
- 10101000を右に2ビットシフトした数

※すべて2進数のままで良い

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 = $\boxed{\text{ビット}}$

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