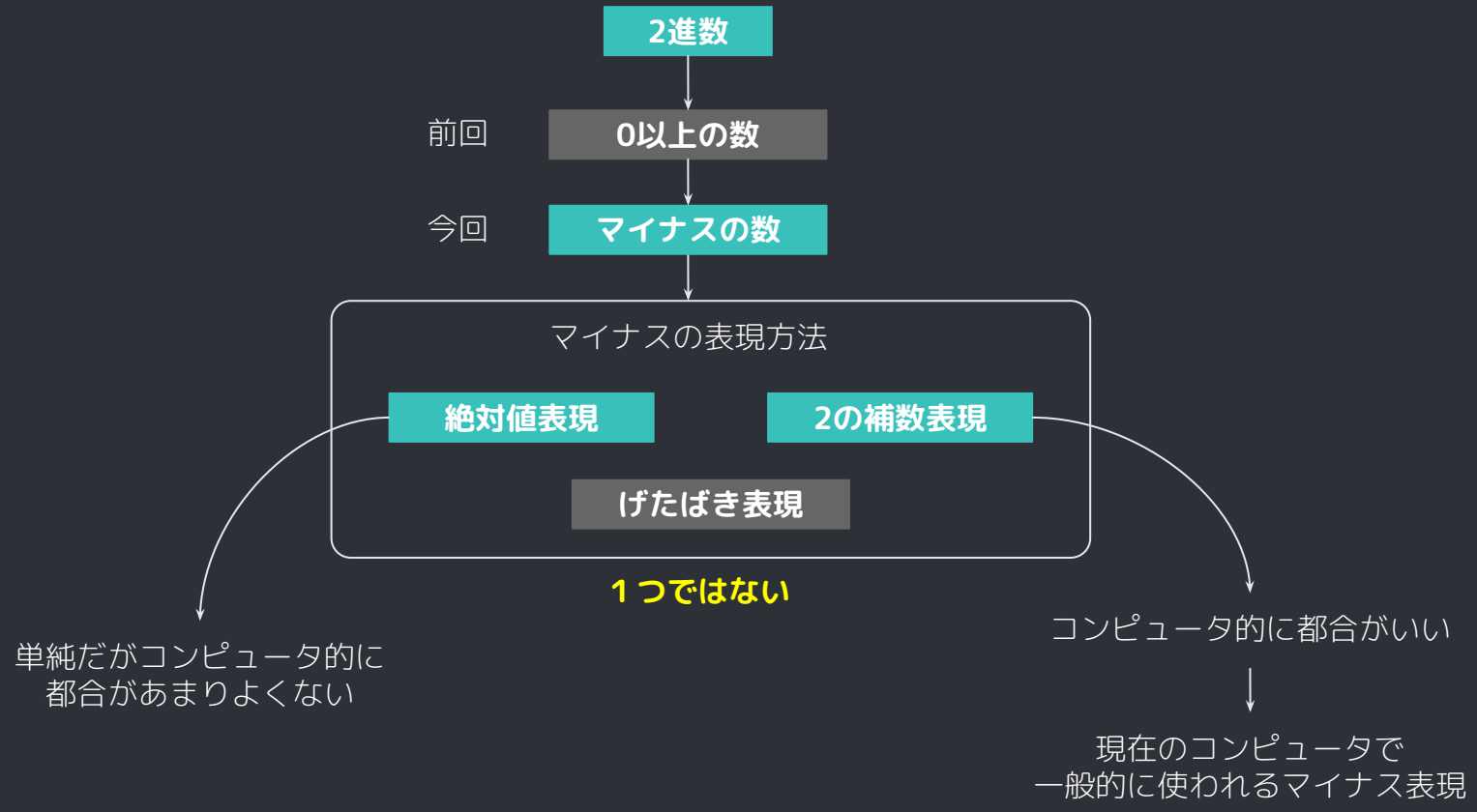
# 2進数でマイナスを表現

プログラマのためのC言語 第6回


2進数
前回
0以上の数
今回
マイナスの数
マイナスの表現方法
絶対値表現
2の補数表現
げたばき表現
１つではない
単純だがコンピュータ的に
都合があまりよくない
コンピュータ的に都合がいい
現在のコンピュータで
一般的に使われるマイナス表現

# 絶対値表現

**10進数では正負をどう表現してる？**

符号 + 絶対値

**+1**　　**-1**

**10進数では正負をどう表現してる？**

符号 + 絶対値

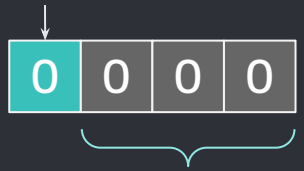
**+1**　　**-1**

**2進数でも同じように考えてみる**

※コンピュータは0と1しか扱えない

符号± も **0と1** で表現

符号ビット(0なら+、1なら-)




絶対値 3bitなら 0〜7 まで扱える

**10進数では正負をどう表現してる？**

符号 + 絶対値

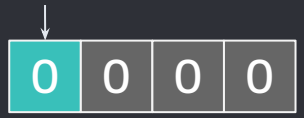
**+1**　　**-1**

**2進数でも同じように考えてみる**

※コンピュータは0と1しか扱えない

符号± も **0と1** で表現

符号ビット(0なら+、1なら-)




絶対値 3bitなら 0〜7 まで扱える

| 10進数 | 符号ビット | 絶対値 | 2進数 |
|---|---|---|---|
| +7 | 0 | 111 | 0111 |
| +6 | 0 | 110 | 0110 |
| +5 | 0 | 101 | 0101 |
| +4 | 0 | 100 | 0100 |
| +3 | 0 | 011 | 0011 |
| +2 | 0 | 010 | 0010 |
| +1 | 0 | 001 | 0001 |
| +0 | 0 | 000 | 0000 |
| -0 | 1 | 000 | 1000 |
| -1 | 1 | 001 | 1001 |
| -2 | 1 | 010 | 1010 |
| -3 | 1 | 011 | 1011 |
| -4 | 1 | 100 | 1100 |
| -5 | 1 | 101 | 1101 |
| -6 | 1 | 110 | 1110 |
| -7 | 1 | 111 | 1111 |

**10進数では正負をどう表現してる？**

符号 + 絶対値

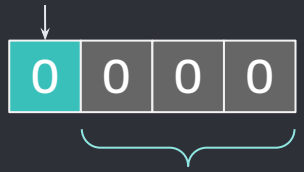
**+1**　　**-1**

**2進数でも同じように考えてみる**

※コンピュータは0と1しか扱えない

符号± も **0と1** で表現

符号ビット(0なら+、1なら-)




絶対値 3bitなら 0〜7 まで扱える

| 10進数 | 符号ビット | 絶対値 | 2進数 |
|---|---|---|---|
| +7 | 0 | 111 | 0111 |
| +6 | 0 | 110 | 0110 |
| +5 | 0 | 101 | 0101 |
| +4 | 0 | 100 | 0100 |
| +3 | 0 | 011 | 0011 |
| +2 | 0 | 010 | 0010 |
| +1 | 0 | 001 | 0001 |
| +0 | 0 | 000 | 0000 |
| -0 | 1 | 000 | 1000 |
| -1 | 1 | 001 | 1001 |
| -2 | 1 | 010 | 1010 |
| -3 | 1 | 011 | 1011 |
| -4 | 1 | 100 | 1100 |
| -5 | 1 | 101 | 1101 |
| -6 | 1 | 110 | 1110 |
| -7 | 1 | 111 | 1111 |

**±0**

0が2つできる

**10進数では正負をどう表現してる？**

符号 + 絶対値

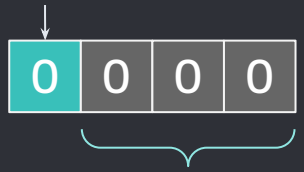
**+1**　　**-1**

**2進数でも同じように考えてみる**

※コンピュータは0と1しか扱えない

符号± も **0と1** で表現

符号ビット(0なら+、1なら-)




絶対値 3bitなら 0〜7 まで扱える

| 10進数 | 符号ビット | 絶対値 | 2進数 |
|---|---|---|---|
| +7 | 0 | 111 | 0111 |
| +6 | 0 | 110 | 0110 |
| +5 | 0 | 101 | 0101 |
| +4 | 0 | 100 | 0100 |
| +3 | 0 | 011 | 0011 |
| +2 | 0 | 010 | 0010 |
| +1 | 0 | 001 | 0001 |
| +0 | 0 | 000 | 0000 |
| -0 | 1 | 000 | 1000 |
| -1 | 1 | 001 | 1001 |
| -2 | 1 | 010 | 1010 |
| -3 | 1 | 011 | 1011 |
| -4 | 1 | 100 | 1100 |
| -5 | 1 | 101 | 1101 |
| -6 | 1 | 110 | 1110 |
| -7 | 1 | 111 | 1111 |

**±0**

0が2つできる

1 + (-1) = 0 になって欲しいけど、単純に足すと

0001 + 1001 = 1010 → -2

# 都合がわるい

# 2の補数表現

# 2の補数表現

# 2の補数表現の考え方

# 2の補数表現の考え方

補数

オーバーフロー

マイナス(逆元)

# 2の補数表現の考え方

**補数**

**足すと桁が上がる数**

(例)
4なら6とか
8なら2とか

**オーバーフロー**

**マイナス(逆元)**

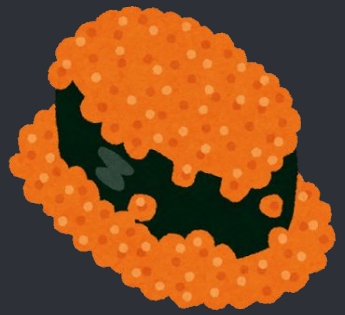
# 2の補数表現の考え方

## 補数

**足すと桁が上がる数**

(例)
4なら6とか
8なら2とか

## オーバーフロー



コンピュータで数値の計算結果が大きくなったときに**桁が溢れること**

溢れた部分は捨てられる

## マイナス(逆元)

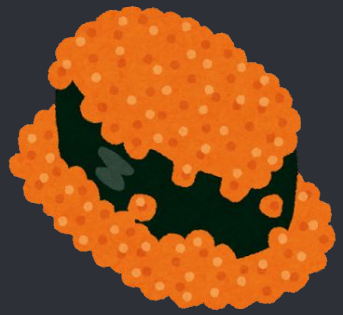
# 2の補数表現の考え方

## 補数

**足すと桁が上がる数**

(例)
4なら6とか
8なら2とか

## オーバーフロー



コンピュータで数値の計算結果が大きくなったときに**桁が溢れること**

溢れた部分は捨てられる

## マイナス(逆元)


**ある数に足すと0になる数**
元の数を打ち消す数

1 + **(-1)** = 0

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

x と置く

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| + | x | x | x | x |
| | 0 | 0 | 0 | 0 |

2進数で書くとこういうこと

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| + | x | x | x | x |
| | 0 | 0 | 0 | 0 |

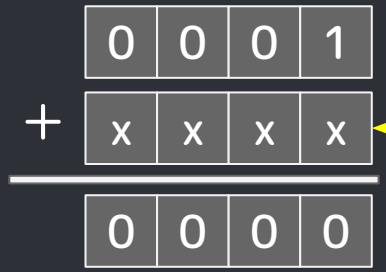
# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1




つまり、これが -1 ということ

xxxx を求める

2進数の -1 がわかる

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| + | x | x | x | x |
| | 0 | 0 | 0 | 0 |

# 2の補数表現の考え方

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| + | x | x | x | x |
| 1 | 0 | 0 | 0 | 0 |

## xxxx を求める

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| + | 1 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 |

足すと桁が上がる数を入れる

**2の補数** という

## xxxx を求める

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1


足すと桁が上がる数を入れる

2の補数 という

計算すると 桁上がり する

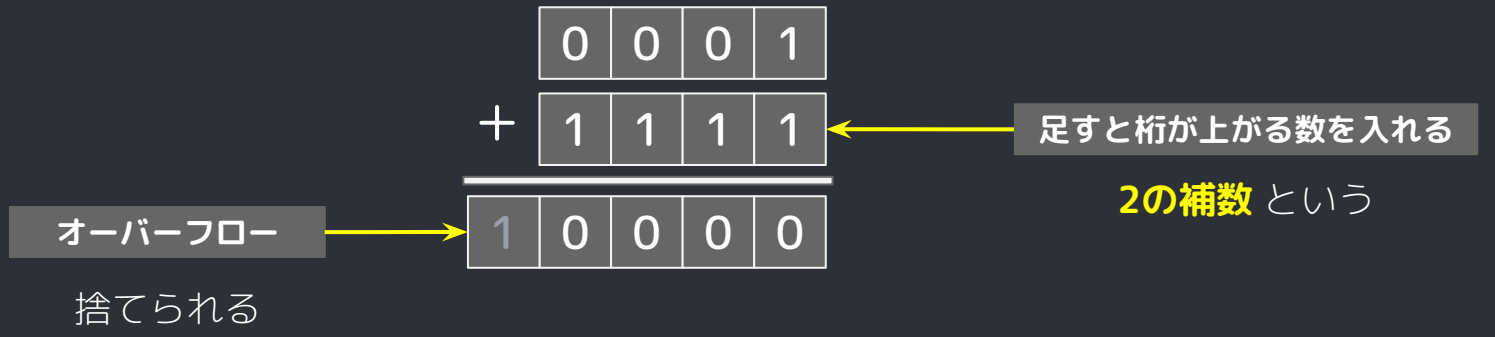
## xxxx を求める

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1



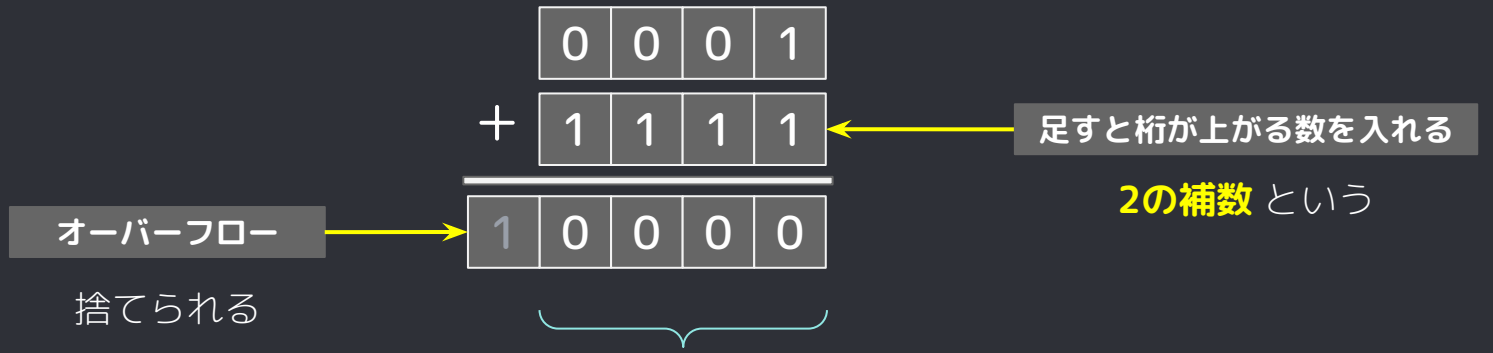
## xxxx を求める

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1




結果的に 足した結果が0

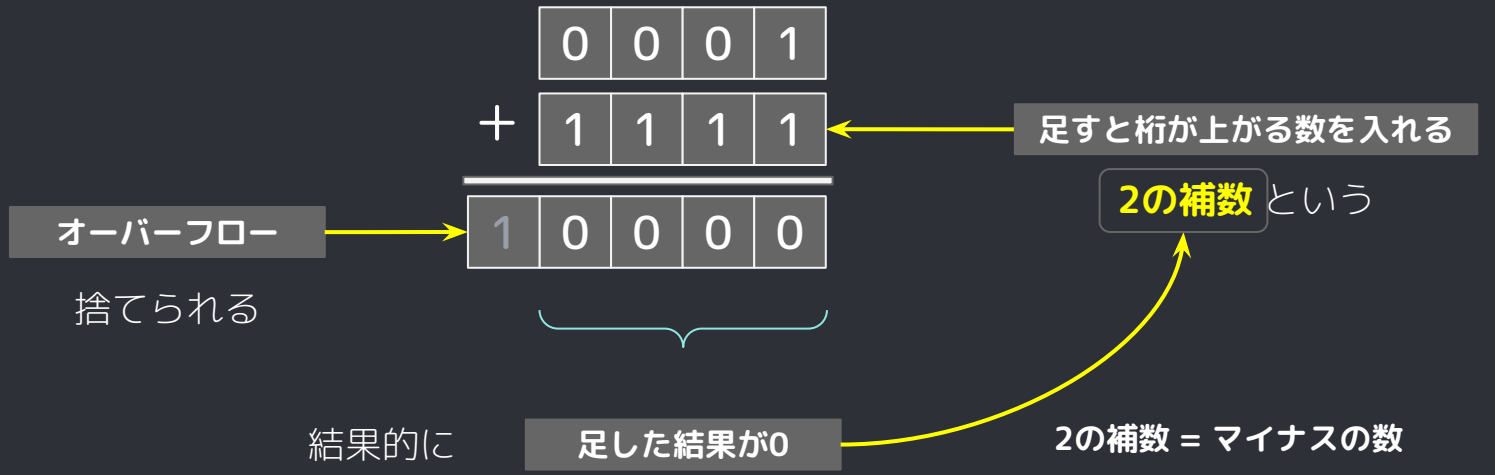
## xxxx を求める

仮に数を 4桁(4bit) で表すとする

1に 足したら0になる数 を求める

1 + x = 0

x = -1



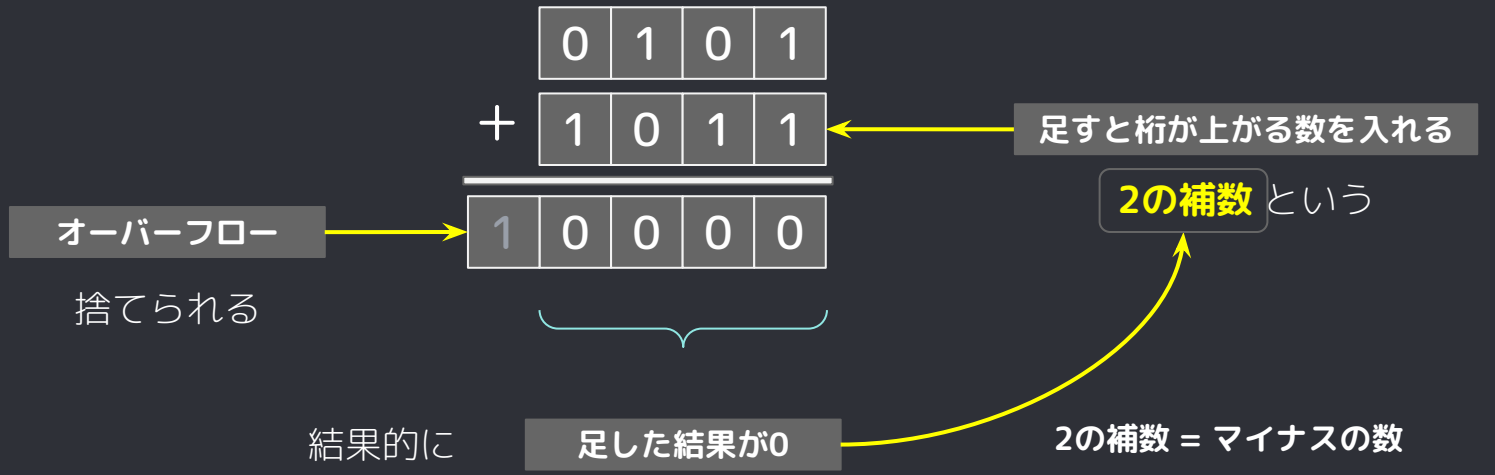
## xxxx を求める

仮に数を 4桁(4bit) で表すとする

5に 足したら0になる数 を求める

5 + x = 0

x = -5




他の数でも同じ、例えば -5 は 1011 になる

# 2の補数？

## 補数とは？

元の数に足して桁が1つ上がる最も小さい数のこと

## 補数とは？

元の数に足して桁が1つ上がる最も小さい数のこと

4+6 = 10

12 + 88 = 100

0001 + 1111 = 10000

0010 + 1110 = 10000

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数
減基数の補数

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

**基数の補数**　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

**基数の補数**　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数

4+6 = 10

12 + 88 = 100

0001 + 1111 = 10000

0010 + 1110 = 10000

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
**減基数の補数**：元の数に足して桁上がりしない最大の数のこと

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
**減基数の補数**：元の数に足して桁上がりしない最大の数のこと

4+5 = 9

12 + 87 = 99

0001 + 1110 = 1111

0010 + 1101 = 1111

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数：元の数に足して桁上がりしない最大の数のこと

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数：元の数に足して桁上がりしない最大の数のこと

| | 基数の補数 | 減基数の補数 |
|---|---|---|
| 2 進数 | 2の補数 | 1の補数 |
| 10 進数 | 10の補数 | 9の補数 |

基数

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数：元の数に足して桁上がりしない最大の数のこと

**10進数なら「10の補数」「9の補数」がある**

| 10進数 | 9の補数 | 10の補数 |
|---|---|---|
| 6 | 3 | 4 |
| 12 | 87 | 88 |
| 127 | 872 | 873 |
| | 減基数の補数 | 基数の補数 |

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数：元の数に足して桁上がりしない最大の数のこと

**10進数なら「10の補数」「9の補数」がある**

| 10進数 | 9の補数 | 10の補数 |
|---|---|---|
| 6 | 3 | 4 |
| 12 | 87 | 88 |
| 127 | 872 | 873 |
| | 減基数の補数 | 基数の補数 |

**2進数なら「2の補数」「1の補数」がある**

| 2進数 | 1の補数 | 2の補数 |
|---|---|---|
| 01 | 10 | 11 |
| 0100 | 0011 | 0100 |
| 01111111 | 10000000 | 10000001 |
| | 減基数の補数 | 基数の補数 |

## 補数とは？

**元の数に足して桁が1つ上がる最も小さい数のこと**

基数の補数　：元の数に足して桁上がりする最も小さい数のこと
減基数の補数：元の数に足して桁上がりしない最大の数のこと

**10進数なら「10の補数」「9の補数」がある**

| 10進数 | 9の補数 | 10の補数 |
|---|---|---|
| 6 | 3 | 4 |
| 12 | 87 | 88 |
| 127 | 872 | 873 |
| | 減基数の補数 | 基数の補数 |

+1

**2進数なら「2の補数」「1の補数」がある**

| 2進数 | 1の補数 | 2の補数 |
|---|---|---|
| 01 | 10 | 11 |
| 0100 | 0011 | 0100 |
| 01111111 | 10000000 | 10000001 |
| | 減基数の補数 | 基数の補数 |

+1

## 2の補数の求め方

# 1の補数 + 1 = 2の補数

**1の補数を求める**

2進数の場合は元の数の0,1を反転するだけ

**2の補数を求める**

+1して終わり

# 2の補数表現の特徴(4bitの例)

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| | 0000 |
| | 0001 |
| | 0010 |
| | 0011 |
| | 0100 |
| | 0101 |
| | 0110 |
| | 0111 |
| | 1000 |
| | 1001 |
| | 1010 |
| | 1011 |
| | 1100 |
| | 1101 |
| | 1110 |
| | 1111 |

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| | 0000 |
| | 0001 |
| | 0010 |
| | 0011 |
| | 0100 |
| | 0101 |
| | 0110 |
| | 0111 |
| | 1000 |
| | 1001 |
| | 1010 |
| | 1011 |
| | 1100 |
| | 1101 |
| | 1110 |
| | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

- ✓ 4bit = 2の4乗 = 16パターン
- ✓ 全体を半分にわけて使う

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| | 0001 |
| | 0010 |
| | 0011 |
| | 0100 |
| | 0101 |
| | 0110 |
| | 0111 |
| | 1000 |
| | 1001 |
| | 1010 |
| | 1011 |
| | 1100 |
| | 1101 |
| | 1110 |
| | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| | 1000 |
| | 1001 |
| | 1010 |
| | 1011 |
| | 1100 |
| | 1101 |
| | 1110 |
| | 1111 |

+1

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

- ✓ 4bit = 2の4乗 = 16パターン
- ✓ 全体を半分にわけて使う

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| | 1000 |
| | 1001 |
| | 1010 |
| | 1011 |
| | 1100 |
| | 1101 |
| | 1110 |
| -1 | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| -8 | 1000 |
| -7 | 1001 |
| -6 | 1010 |
| -5 | 1011 |
| -4 | 1100 |
| -3 | 1101 |
| -2 | 1110 |
| -1 | 1111 |

0以上 8パターン

0未満 8パターン

-1

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| -8 | 1000 |
| -7 | 1001 |
| -6 | 1010 |
| -5 | 1011 |
| -4 | 1100 |
| -3 | 1101 |
| -2 | 1110 |
| -1 | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

## 特徴

✓ 4bitでは **-8 ～ 7 を表せる**

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| -8 | 1000 |
| -7 | 1001 |
| -6 | 1010 |
| -5 | 1011 |
| -4 | 1100 |
| -3 | 1101 |
| -2 | 1110 |
| -1 | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

## 特徴

✓ 4bitでは **-8 ～ 7 を表せる**

✓ 0は1つになる

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| -8 | 1000 |
| -7 | 1001 |
| -6 | 1010 |
| -5 | 1011 |
| -4 | 1100 |
| -3 | 1101 |
| -2 | 1110 |
| -1 | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

## 特徴

✓ 4bitでは **-8 ～ 7 を表せる**

✓ 0は1つになる

✓ マイナスを含む計算も普通に足すだけ

7 + (-8) = -1

0111 + 1000 = 1111

# 2の補数表現の特徴(4bitの例)

| 10進数 | 2進数 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| -8 | 1000 |
| -7 | 1001 |
| -6 | 1010 |
| -5 | 1011 |
| -4 | 1100 |
| -3 | 1101 |
| -2 | 1110 |
| -1 | 1111 |

0以上 8パターン

0未満 8パターン

## 4bitで表せる数のパターン

✓ 4bit = 2の4乗 = 16パターン

✓ 全体を半分にわけて使う

## 特徴

✓ 4bitでは **-8 ～ 7 を表せる**

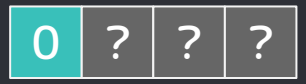
✓ 0は1つになる

✓ マイナスを含む計算も普通に足すだけ

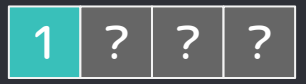
7 + (-8) = -1

0111 + 1000 = 1111

✓ 先頭のビットが符号を表す符号ビットになる




**0 か +**



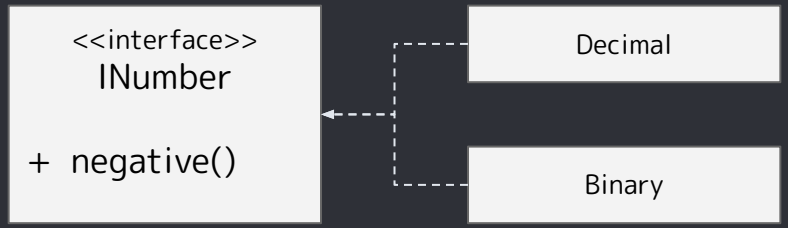

**-**

## まとめ

- ✔ コンピューターは0と1で表現するしかない
- ✔ 故に0と1でなんとかするというのが基本にある
- ✔ マイナスの数は 2の補数 で表現
  - ・オーバーフローを利用して実現(桁数が決まっている)
  - ・マイナスを含む計算も普通に足すだけでよい

## まとめ

- ✔ コンピューターは0と1で表現するしかない
- ✔ 故に0と1でなんとかするというのが基本にある
- ✔ マイナスの数は 2の補数 で表現
  - ・オーバーフローを利用して実現(桁数が決まっている)
  - ・マイナスを含む計算も普通に足すだけでよい



# おしまい