

- アナログ計算機 (計算尺など)
- デジタル計算機
  - ★ プログラム機構方式
  - ★ プログラム入力方式
  - ★ プログラム内蔵方式  
(von Neumann 方式)
- 量子計算機

計算機内では  
全てのデータは 0, 1 の組 (列) で表される。

情報の最小単位  
0 か 1 か : **bit (binary digit)**

実際には幾つかの **bit** を組にして一度に扱う  
(通常は 8 **bit** = 1 **byte (octet)**)

1 **byte** で  $2^8 = 256$  通りの値を表せる

数の表し方: 二進法

正の整数の場合、1 byte で

$$0 \leq x \leq 2^8 - 1 = 255$$

の値が表せる

十進	二進
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
⋮	⋮
253	11111101
254	11111110
255	11111111

## 二進の九九 (一一?) の表

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

問題:

次の各式の被演算子 (operand) を  
二進表示で表した上で、  
それを用いて筆算で計算せよ。

(1)  $87 + 26$

(2)  $87 + 50$

(3)  $87 - 26$

(4)  $13 \times 11$

計算機の内部では、

1 つの整数値を 8 bit で表している。

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline \end{array} = 87$$

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline \end{array} = 26$$

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hline \end{array} = 50$$

二進では桁数が多くて煩わしい

→ 十六進 (hexadecimal) 表記

十進	二進	十六進	十進	二進	十六進
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	a
3	0011	3	11	1011	b
4	0100	4	12	1100	c
5	0101	5	13	1101	d
6	0110	6	14	1110	e
7	0111	7	15	1111	f



十進	二進	十六進
0	00000000	0x00
1	00000001	0x01
2	00000010	0x02
3	00000011	0x03
4	00000100	0x04
5	00000101	0x05
⋮	⋮	⋮
253	11111101	0xfd
254	11111110	0xfe
255	11111111	0xff

練習問題:

十六進の九九 (FF?) の表を作ってみよう。

	0	1	2	...	e	f
0						
1						
2						
⋮						
e						
f						

練習問題:

さっきの演習を十六進で筆算せよ。

負の数 (符号付き整数) を表すには?  
(- も 0, 1 で表す)

単純に考えると、

- 符号に 1 bit (正なら 0・負なら 1)
- 絶対値に残り 7 bit ( $0 \leq |x| \leq 2^7 - 1$ )

→ 欠点あり

- 0 が 2 通りに表される (+0, -0)
- 演算で正負の場合分けが面倒

→ 他にもっと良い方法はないか?

負の数 (符号付き整数) を表すには?  
(- も 0, 1 で表す)

→ “2 の補数表示”

$$-128 = -2^7 \leq x \leq 2^7 - 1 = 127$$

の範囲の整数を表せる

$$x' := \begin{cases} x & (0 \leq x \leq 2^7 - 1) \\ x + 2^8 & (-2^7 \leq x \leq -1) \end{cases}$$

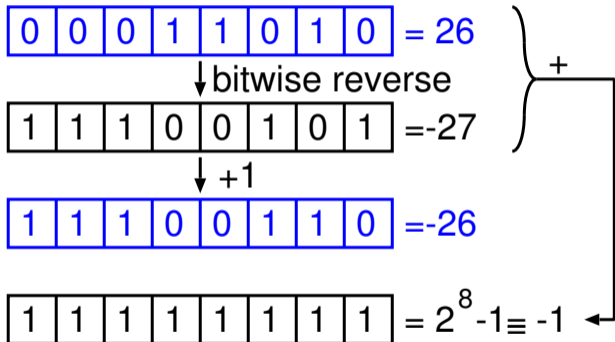
とおくと、

$$0 \leq x' \leq 2^8 - 1$$

→  $x'$  を符号無し整数として表す

$$x \equiv x' \pmod{2^8}$$

$x$  ( $0 \leq x \leq 2^7 - 1$ ) から  $-x$  を求めるには、



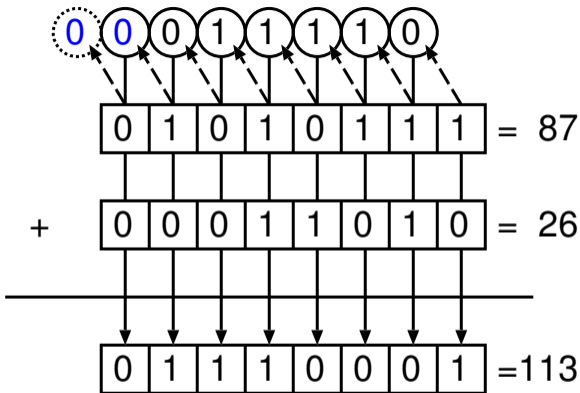
$-2^7 \leq x \leq 2^7 - 1$  の範囲の整数値が表せる。

$$87 + 26$$

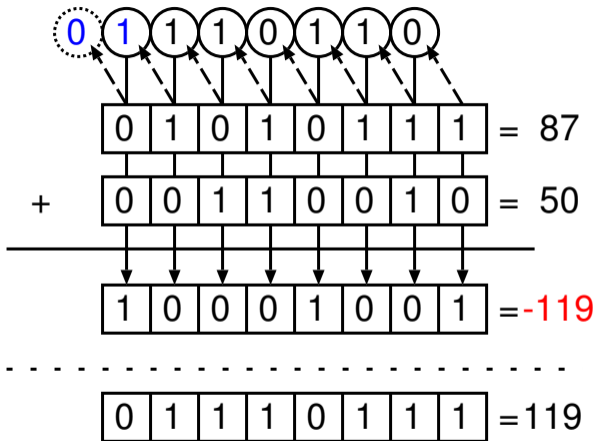
- 結果が上の範囲に収まる
- 正しく求まる。

$$87 + 50$$

- 結果が上の範囲に収まらない。
- 正しく求まらない。「桁溢れ」



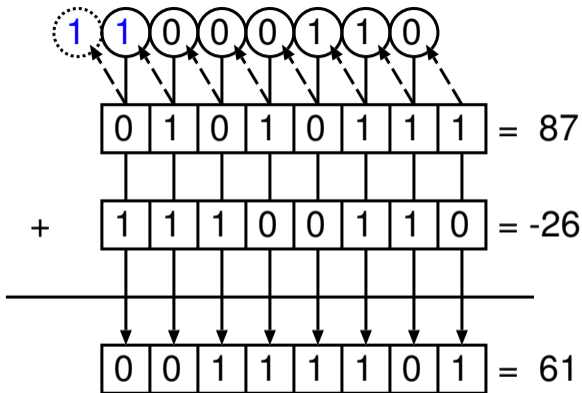




87 - 26

→ 減算を別の演算として扱うのは面倒

→  $87 + (-26)$  として計算してみよう



「桁溢れ」かどうかを判定するには？

(どこの繰上がりを見れば判るか？)

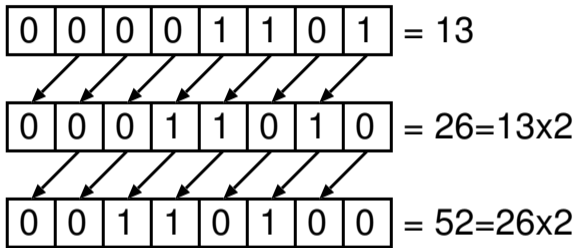
(→ レポート問題の一つ)

$$13 \times 11$$

乗算は桁をずらして加える

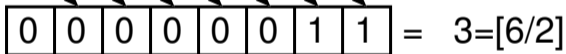
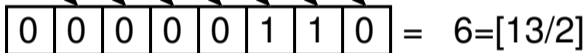
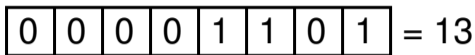
「桁ずらし (**bit shift**)」が基本的な操作

左シフト:  $x \mapsto 2x$



- 桁溢れの判定は？
- 負の数でも大丈夫か？

右シフト:  $x \mapsto \left\lfloor \frac{x}{2} \right\rfloor$



- 桁溢れは起きない。
- 負の数でも大丈夫にするには？  
(符号 bit の扱い)

以上のような、

- データ (bit 情報) の保持
- 基本的な演算

を計算機で実現するには？

→ 「論理回路」