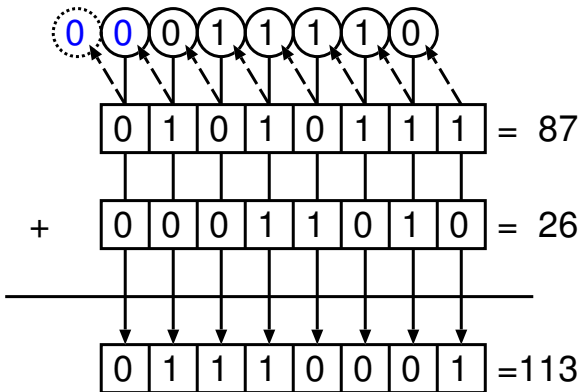
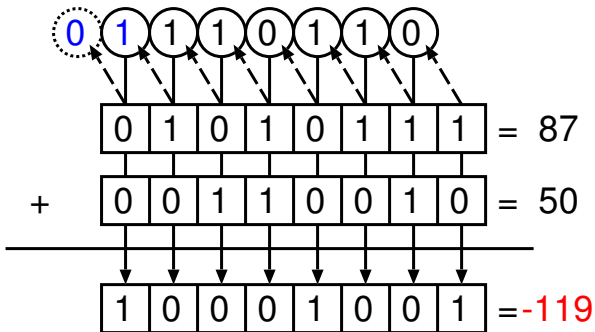


87 + 26



87 + 50



このような、

- 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」

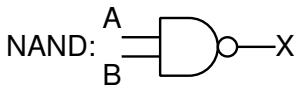
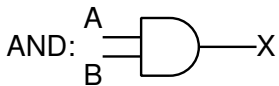
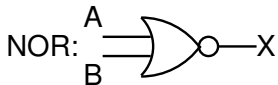
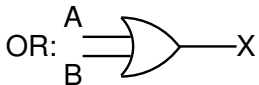
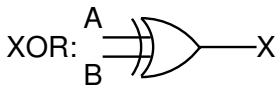
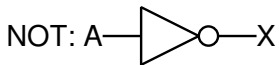
## 論理回路

- 組合せ回路:  
入力 (の組) によって出力が決まる
  
- 順序回路:  
内部状態を保持し、  
入力と入力前の状態とによって  
出力と出力後の状態とが決まる

## 論理回路の基本部品:

### 論理素子 (論理ゲート)

- NOT: 否定 :  $\neg A$
- OR: 論理和 :  $A \vee B$
- AND: 論理積 :  $A \wedge B$
- XOR: 排他的論理和 :  
 $A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$
- NOR: 論理和の否定 :  
 $\neg(A \vee B) = \neg A \wedge \neg B$
- NAND: 論理積の否定 :  
 $\neg(A \wedge B) = \neg A \vee \neg B$



## 真理值表

NOT		X
A	0	1
	1	0

OR	B	
	0	1
A	0	0 1
	1	1 1

AND	B	
	0	1
A	0	0 0
	1	0 1

XOR	B	
	0	1
A	0	0 1
	1	1 0

NOR	B	
	0	1
A	0	1 0
	1	0 0

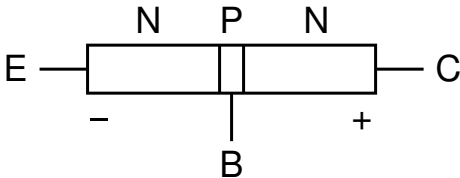
NAND	B	
	0	1
A	0	1 1
	1	1 0

## 論理素子の物理的な実現:

- 電磁石 (リレー)
- 真空管
- 半導体素子 (トランジスタ)



トランジスタ (NPN 接合型) の仕組み:  
ベース (B) の電位を制御することで、  
エミッタ (E)-コレクタ (C) 間の電流が  
大きく変化する



- 情報の増幅
- 情報の伝達 (スイッチング)

- トランジスタ 1 つに適切に配線  
→ NOT ゲート
  
- トランジスタ 2 つを並列に接続  
→ OR ゲート
  
- トランジスタ 2 つを直列に接続  
→ AND ゲート

これらの論理素子を組み合わせ、

基本的な演算を実現しよう。

組合せ回路:

入力 (の組) によって出力が決まる

$n$  入力  $m$  出力の回路は **Boole** 関数

$$f : X_f \longrightarrow \{0, 1\}^m$$

(ここに  $X_f \subset \{0, 1\}^n$  は許される入力全体)  
を定める

組合せ回路 :

**Boole** 関数の論理素子による実現

NOT, OR, AND のみで、

全ての **Boole** 関数を実現できる

$$f(A_1, \dots, A_n) = (X_{11} \wedge \dots \wedge X_{1t_1}) \\ \vee \dots \\ \vee (X_{s1} \wedge \dots \wedge X_{st_s})$$

(各  $X_{ij}$  は  $A_k$  または  $\neg A_k$ )

… 論理和標準形・選言標準形  
(disjunctive normal form, DNF)

双対的に、次の形でも書ける。

$$\begin{aligned} f(A_1, \dots, A_n) = & (X_{11} \vee \dots \vee X_{1t_1}) \\ & \wedge \dots \\ & \wedge (X_{s1} \vee \dots \vee X_{st_s}) \end{aligned}$$

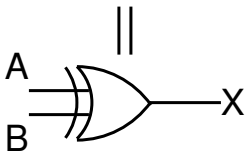
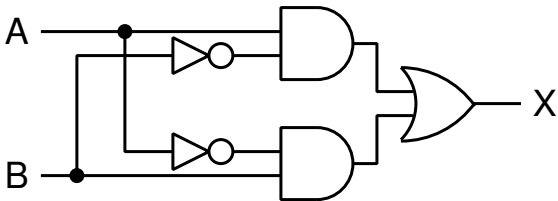
(各  $X_{ij}$  は  $A_k$  または  $\neg A_k$ )

... 論理積標準形・連言標準形  
(**conjunctive normal form, CNF**)

- NOT, OR, AND のみで、  
全ての **Boole** 関数が実現できる
- NOT があれば OR, AND は片方で良い
- OR, AND だけでは  
全ての **Boole** 関数は実現できない
- 一種類の論理素子 NOR または NAND だけで、全ての **Boole** 関数が実現できる

以下では、  
NOT, OR, AND を用いた実現を考える。

例: XOR





問題:

全加算器 (full adder) を、  
NOT, OR, AND を用いて構成せよ。

- 入力: **A**, **B**: 各桁の値, **C**: 下からの繰上がり
- 出力: **X**: 上への繰上がり, **Y**: 当桁の値

