**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が決まる**

**許される内部状態:** $Q \subset \{0,1\}^B$

$n$ **入力** $m$ **出力の順序回路は Boole 関数**

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

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

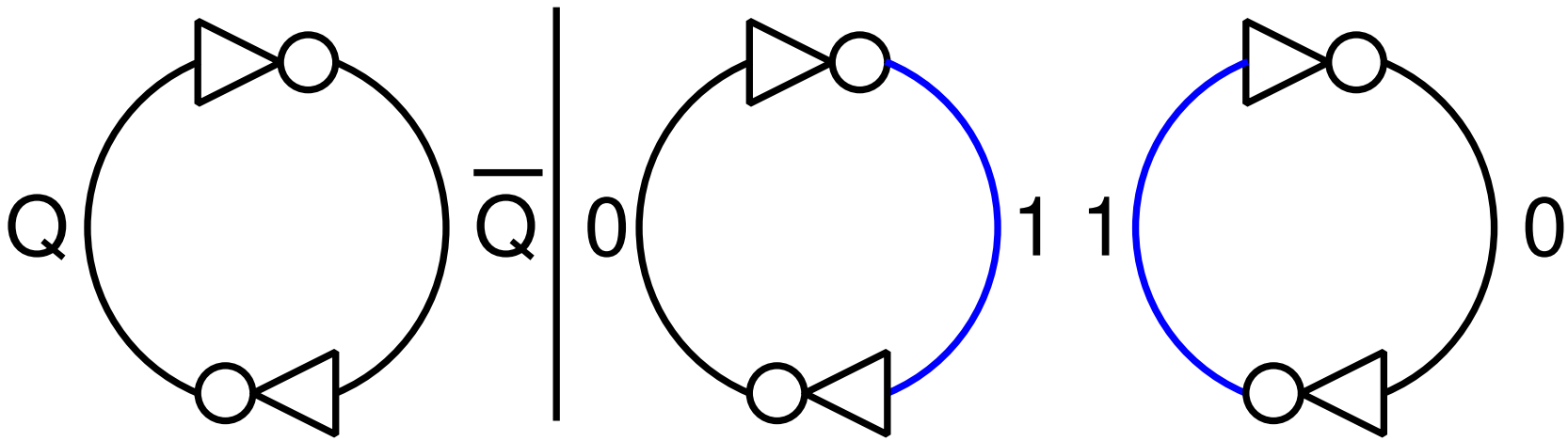
# 内部状態を計算機内に如何に保持するか

- コンデンサに電荷を蓄える

- 複数の安定状態を持つ論理回路で実現
  例: フリップフロップ

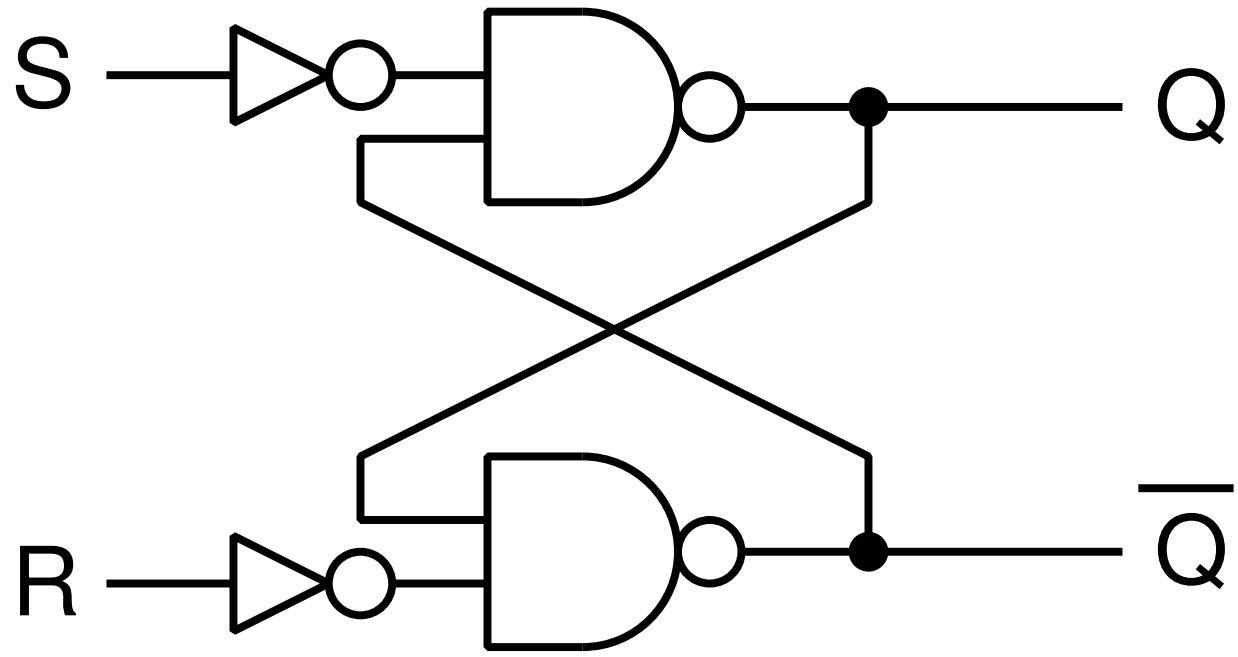
# フリップフロップ (flip-flop)

## 原理図:

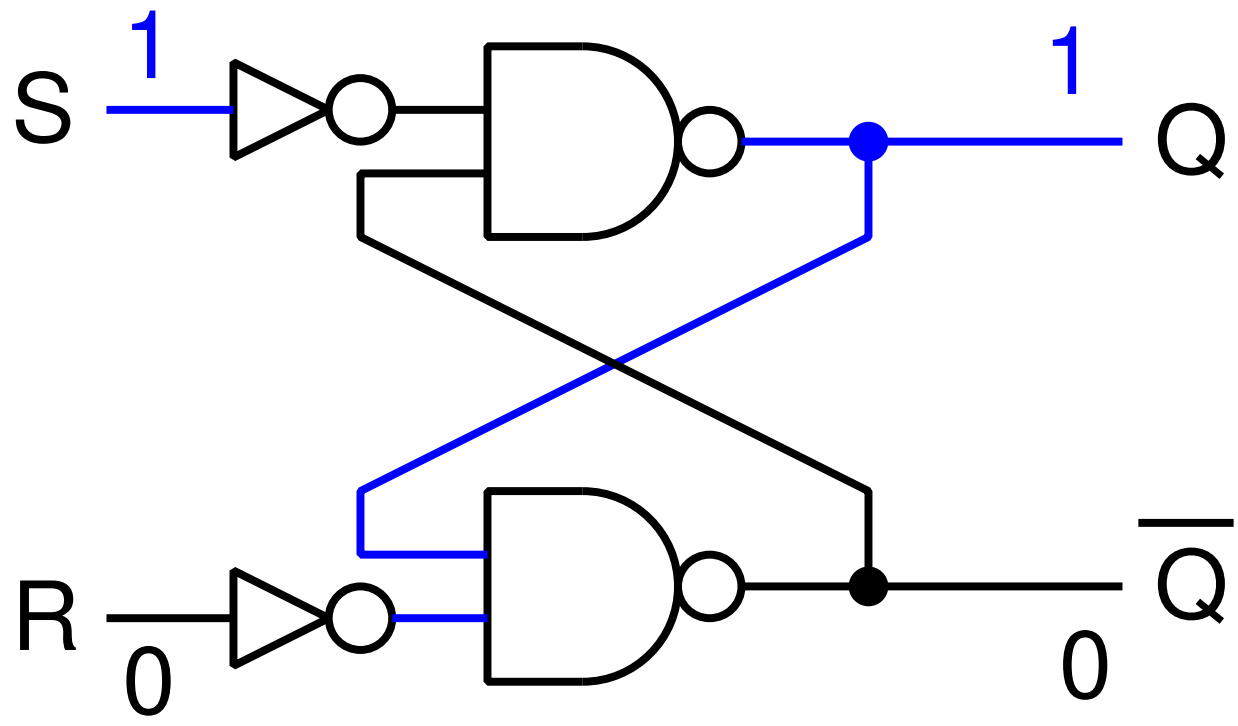



**これに入出力端子を付ける**

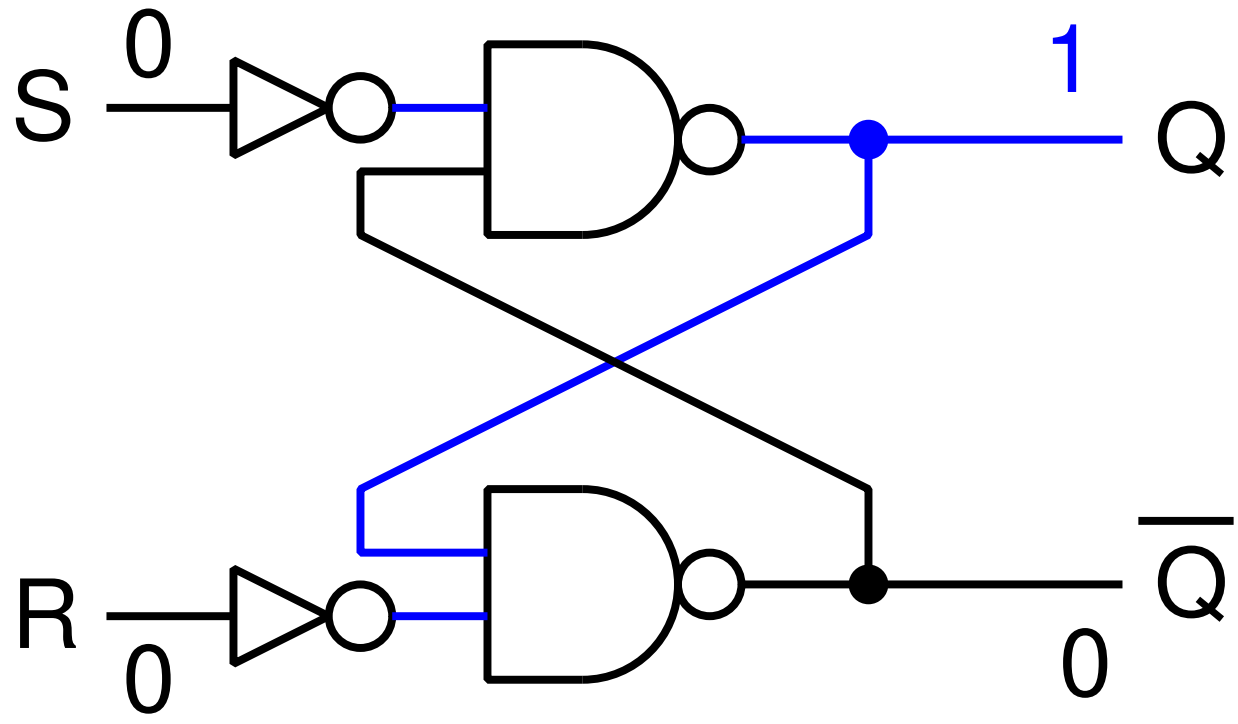
# SR-フリップフロップ (Set-Reset)



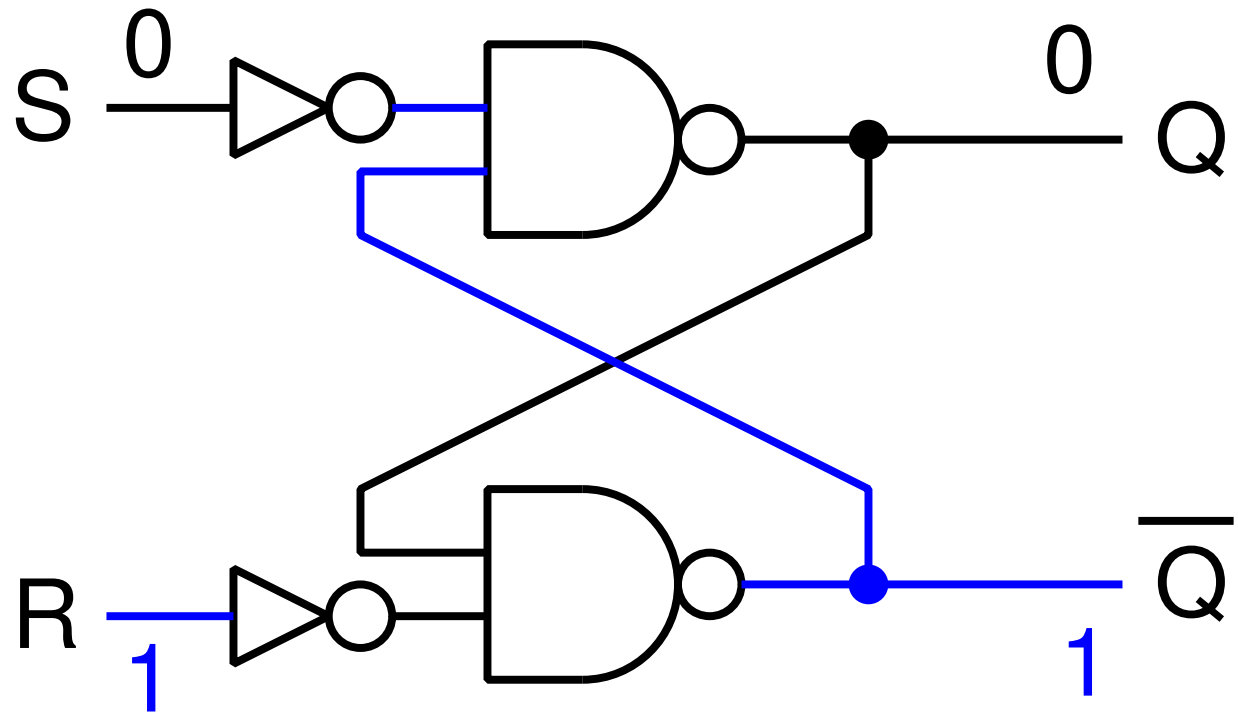
$$(S, R) = (1, 0) \longrightarrow (0, 0) \longrightarrow (0, 1) \longrightarrow (0, 0)$$



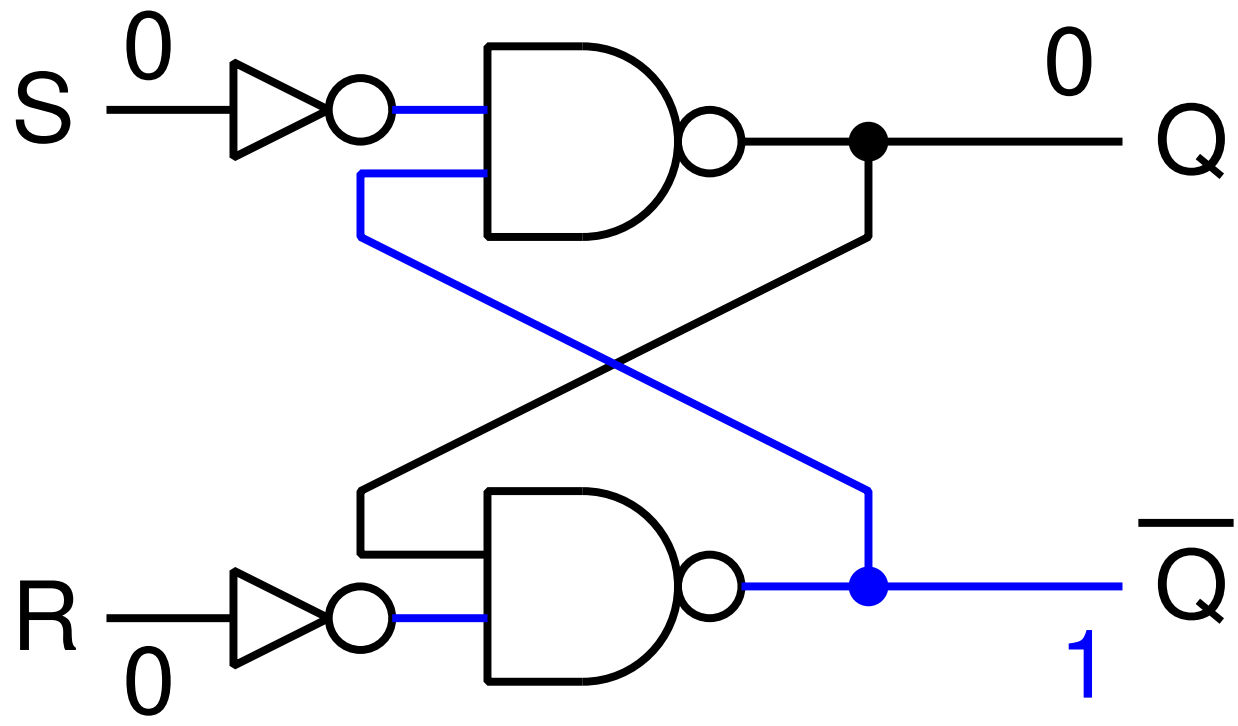
$$(S, R) = (1, 0) \longrightarrow (0, 0) \longrightarrow (0, 1) \longrightarrow (0, 0)$$



$$(S, R) = (1, 0) \longrightarrow (0, 0) \longrightarrow (0, 1) \longrightarrow (0, 0)$$



$$(S,R) = (1,0) \longrightarrow (0,0) \longrightarrow (0,1) \longrightarrow (0,0)$$



$(S, R) = (1, 1)$ **は禁止入力**

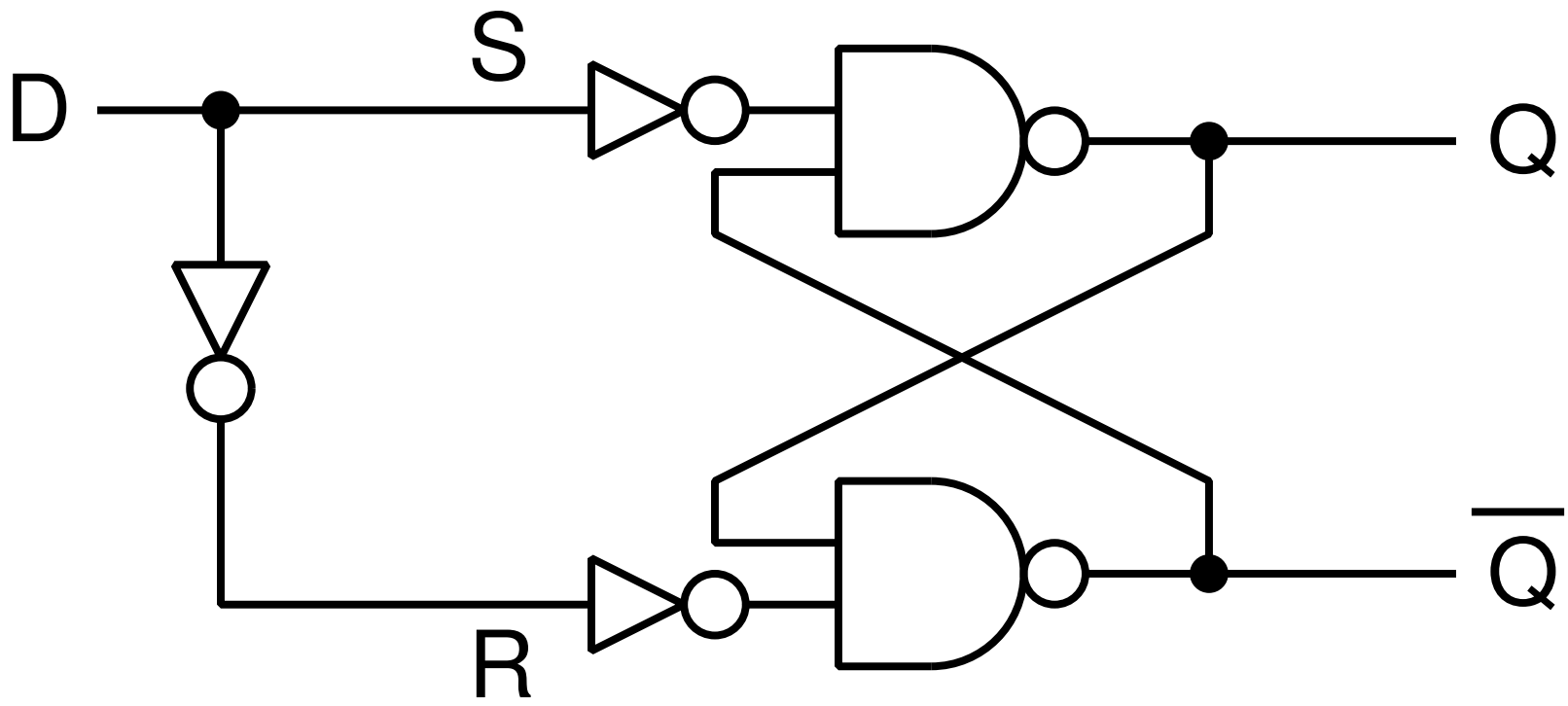
**(**$X_f = \{(0, 0), (0, 1), (1, 0)\}$**)**

| S | R | Q | Q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | x |
| 1 | 1 | 1 | x |

| S | R | Q |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | x |

**入力が $(S, R) = (1, 1)$ とならない回路設計**

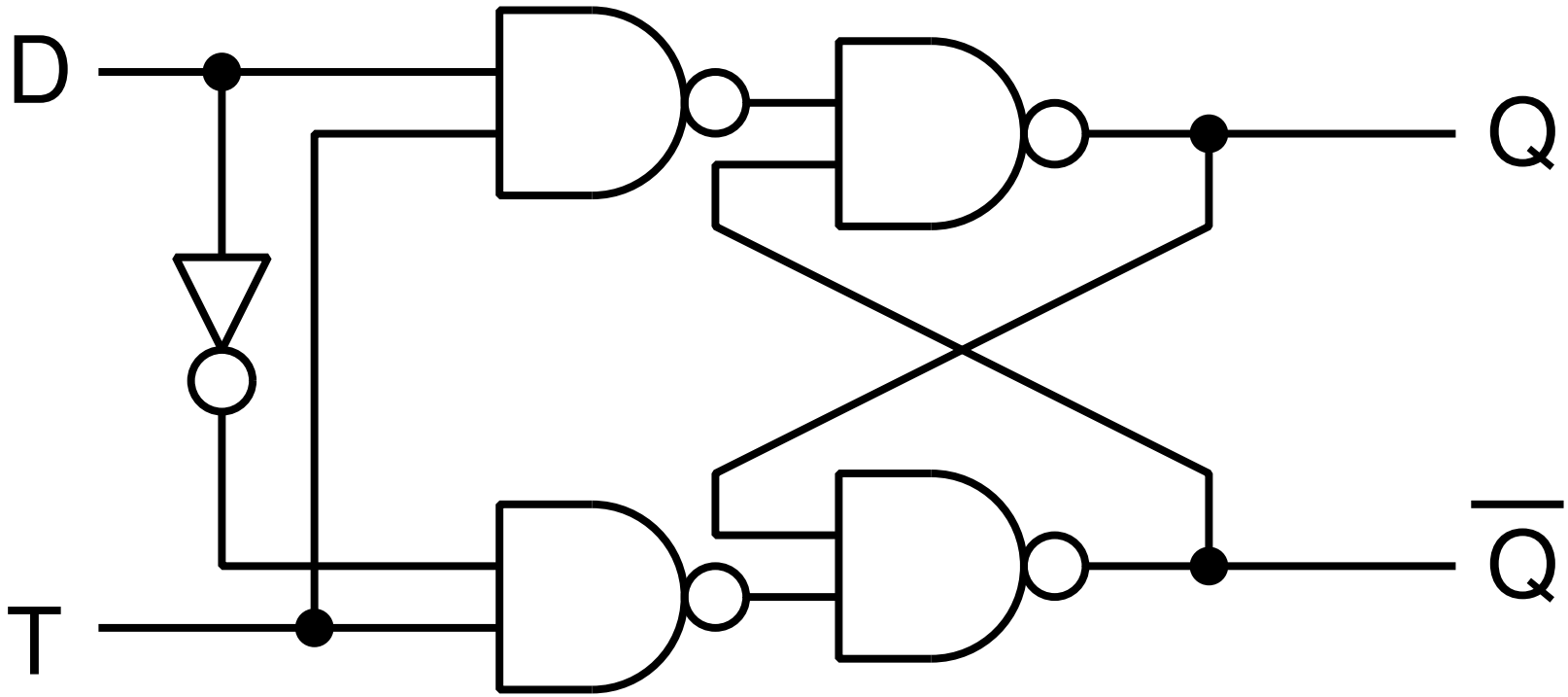
$\longrightarrow$ **$S = \overline{R}$ となるようにしてみる**

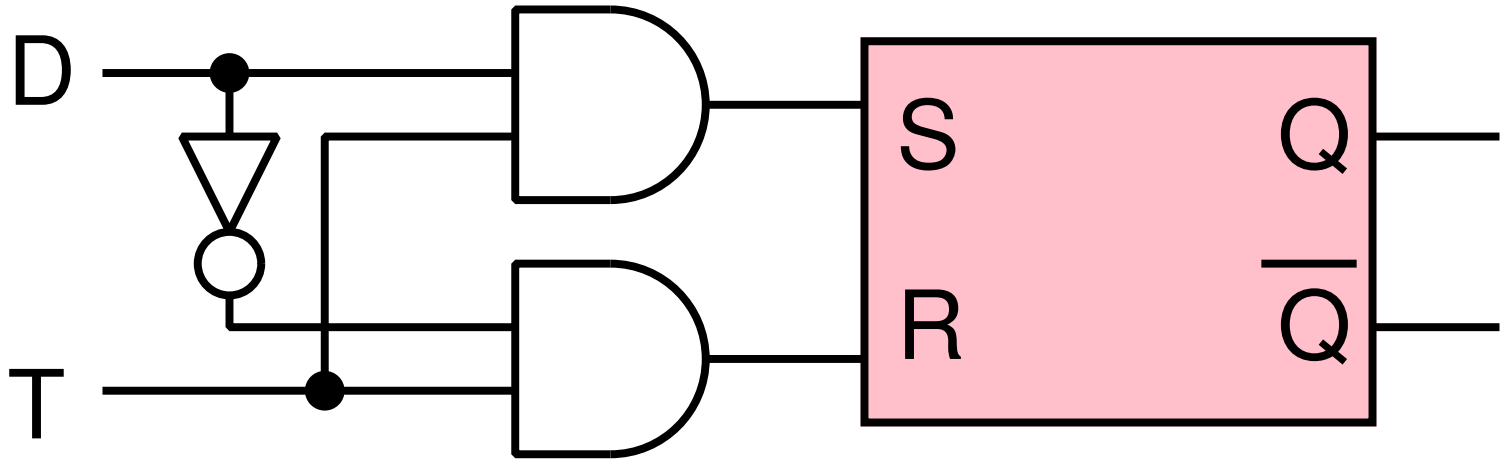



$\longrightarrow$ **入力 $D$ を保持・出力**

$T$：**スイッチ入力**

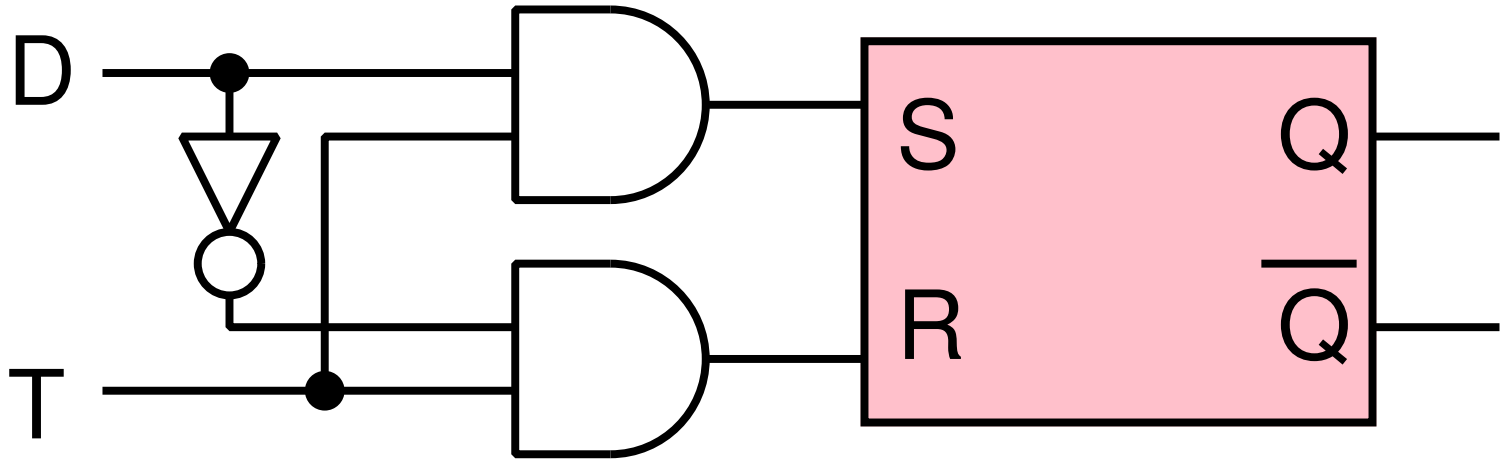
- $T = 0$：**そのまま**
- $T = 1$：$D$ **を取り込む**






| T | D | S | R | Q |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Q |
| 0 | 1 | 0 | 0 | Q |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |

| T | D | S | R | Q |
|---|---|---|---|---|
| 0 | * | 0 | 0 | Q |
| 1 | * | D | $\overline{D}$ | D |




- $Q$ からの出力により他の値を計算
- 他からの入力により $D$ を計算

$\longrightarrow$ $Q$ が $D$ に影響を与えることにより、
状態が変わってしまう

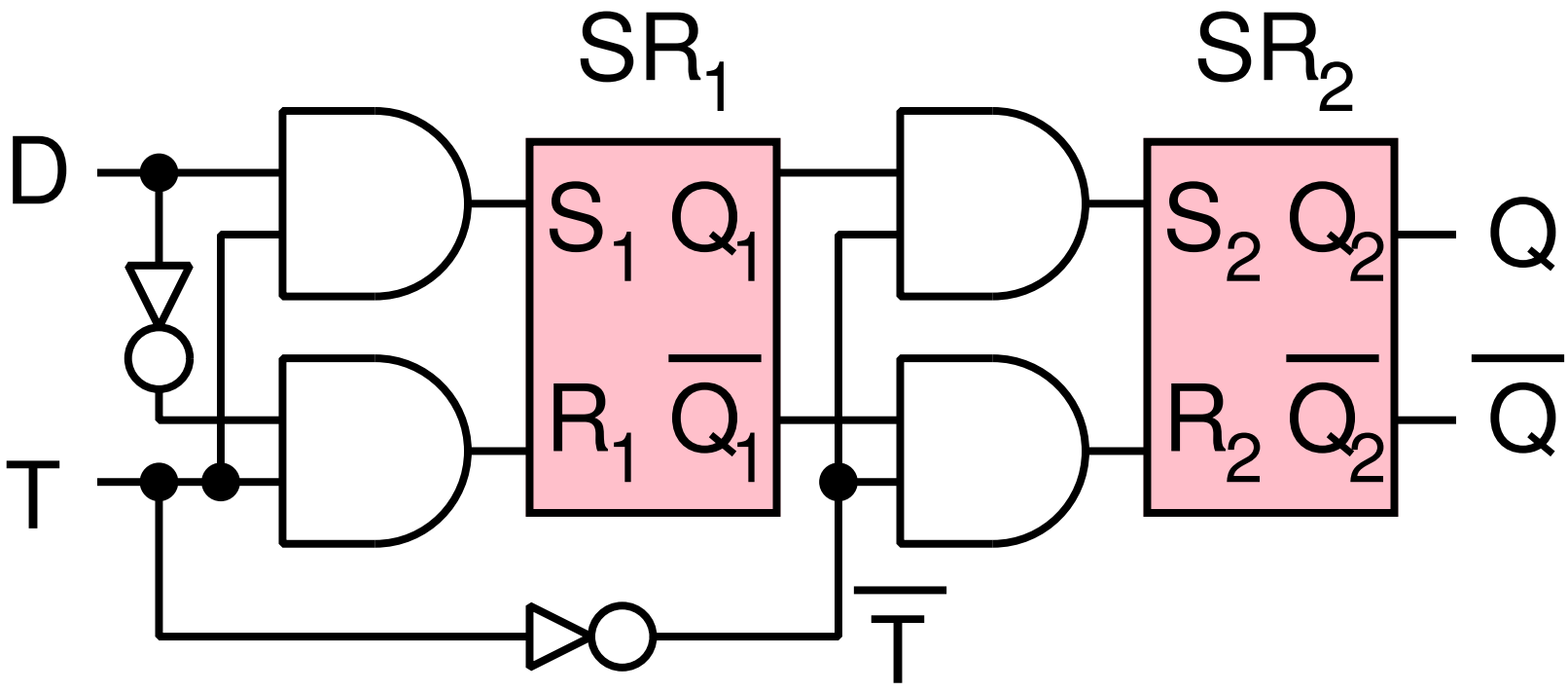
**回路の他の部分と**
**タイミングを合わせる必要あり**

$\longrightarrow$ **出力を一旦せき止めて、**
**一段階づつ計算を進める**

$\longrightarrow$ **D-フリップフロップ・クロックパルス**
**の利用**

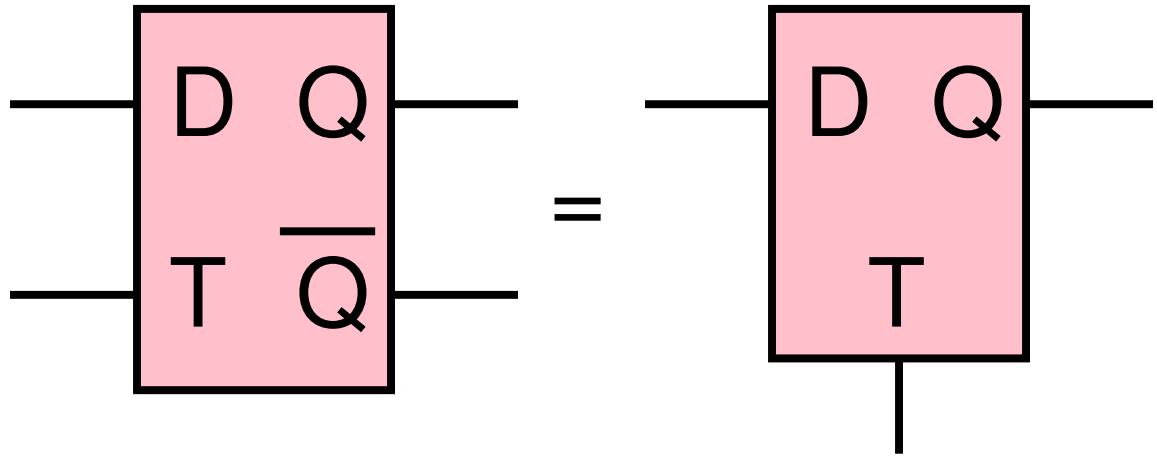
**(ここまで復習)**

# D-フリップフロップ (Double, Delay)




- $T = 1, \overline{T} = 0 \Longrightarrow$ **SR$_1$: 取込、SR$_2$: 保持**
- $T = 0, \overline{T} = 1 \Longrightarrow$ **SR$_1$: 保持、SR$_2$: 取込**

# D-フリップフロップ (Double, Delay)

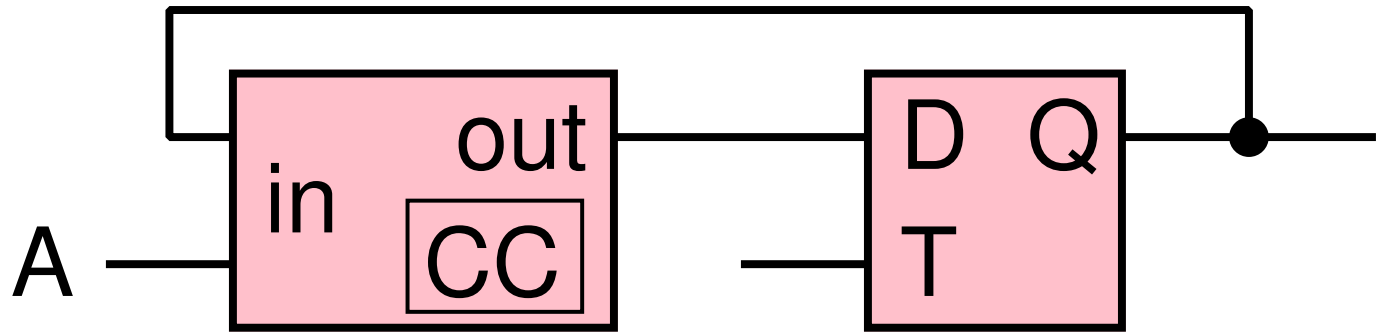



- $T = 1$ : **入力 $D$ を内部に取り込む (出力 $Q$ は変わらない)**
- $T = 0$ : **内部状態を $Q$ に送り出す (入力 $D$ は受付けない)**

**例: ($CC$ は適当な組合せ回路)**

- $T = 1$ : **入力 $D$ を内部に取り込む**
- $T = 0$ : **$Q$ に出力 $\longrightarrow$ $CC$ の入力へ**
  $\longrightarrow$ **$CC$ での計算結果が $D$ に到達**
- $T = 1$ : **新たな $D$ を内部に取り込む**

$\longrightarrow$ **$T = 1 \to 0 \to 1$ と変化する度に、**
**計算が一段階進む**



**一定の周期で $1 \rightarrow 0 \rightarrow 1$ を繰り返す信号**

**(クロックパルス, clock pulse) を**

**水晶発振器などで発生させることにより、**

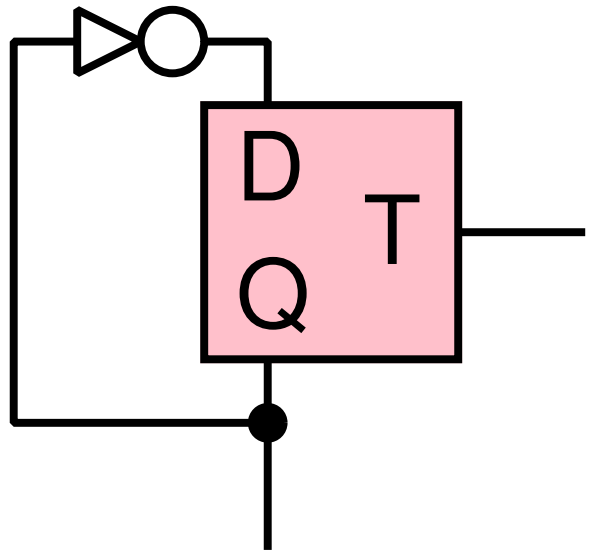
**回路全体での同期を取って、計算を進める。**

**クロックパルスの周波数**

**$\longrightarrow$ 計算機の動作の速さ**
**(動作 clock ○○ Hz)**

# (二進) カウンタ

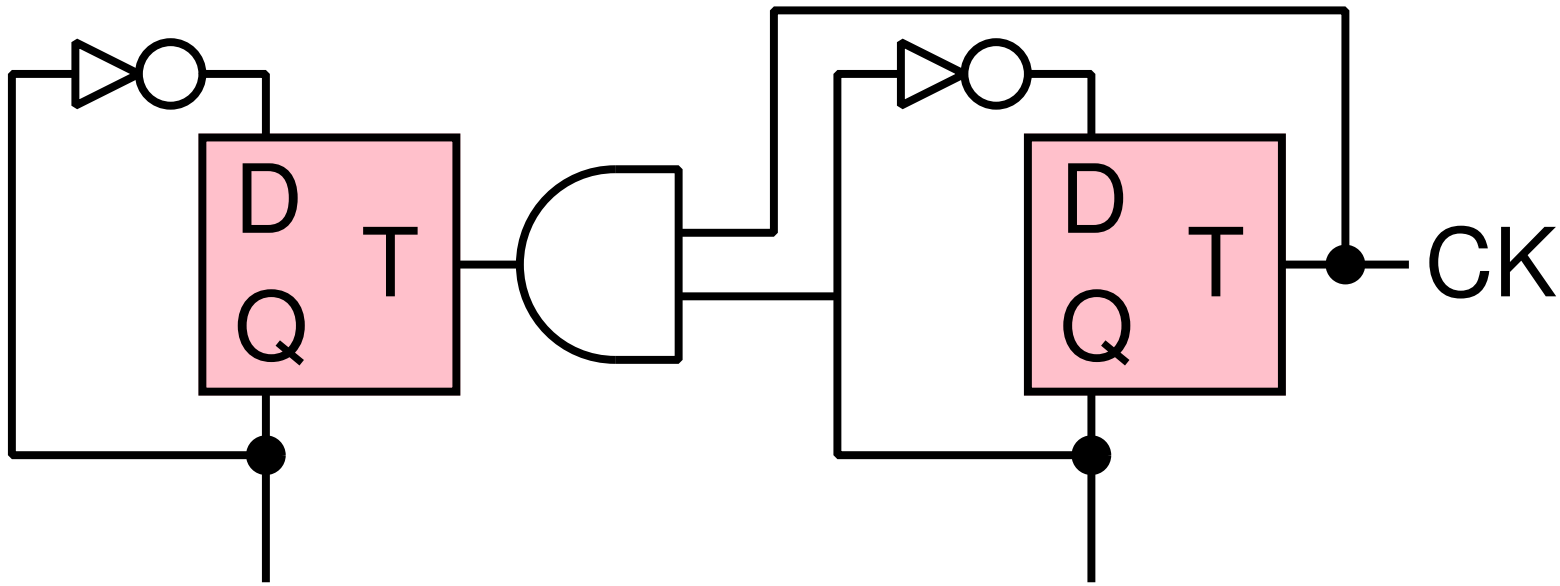
**$T$ が一周期変化する度に**
**内部状態が $1 \to 0 \to 1$ と変化する。**



# 四進カウンタ

$T$ **が一周期変化する度に、内部状態が**

$00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00$ **と変化する。**



**ここまでで、**

**計算機を構成する基本的な論理回路の**

**部品については紹介した。**

⇓

**どのような部品をどう組み合わせれば**

**計算機の機能を実現できるか**

# 計算機が実現すべきこと

- データの保持・書込・読出 $\longrightarrow$ 順序回路
- データの処理 (演算) $\longrightarrow$ 組合せ回路
- (データの入出力)
- 実行の制御
  - $\star$ プログラムの読出
  - $\star$ 順次実行
  - $\star$ 条件分岐

## データの保持: 記憶装置 (memory)

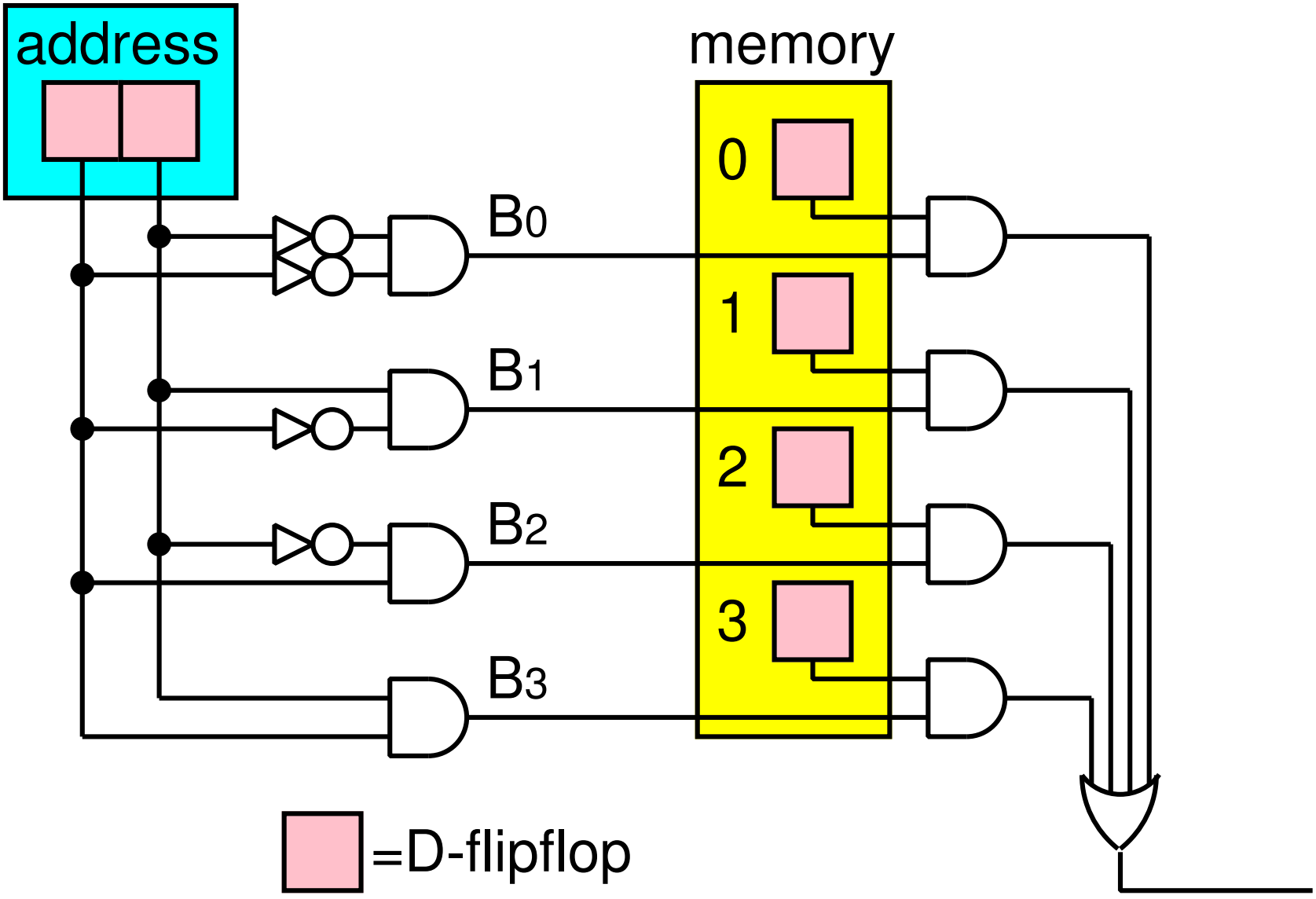
## D-フリップフロップを必要なだけ並べる

## その中の

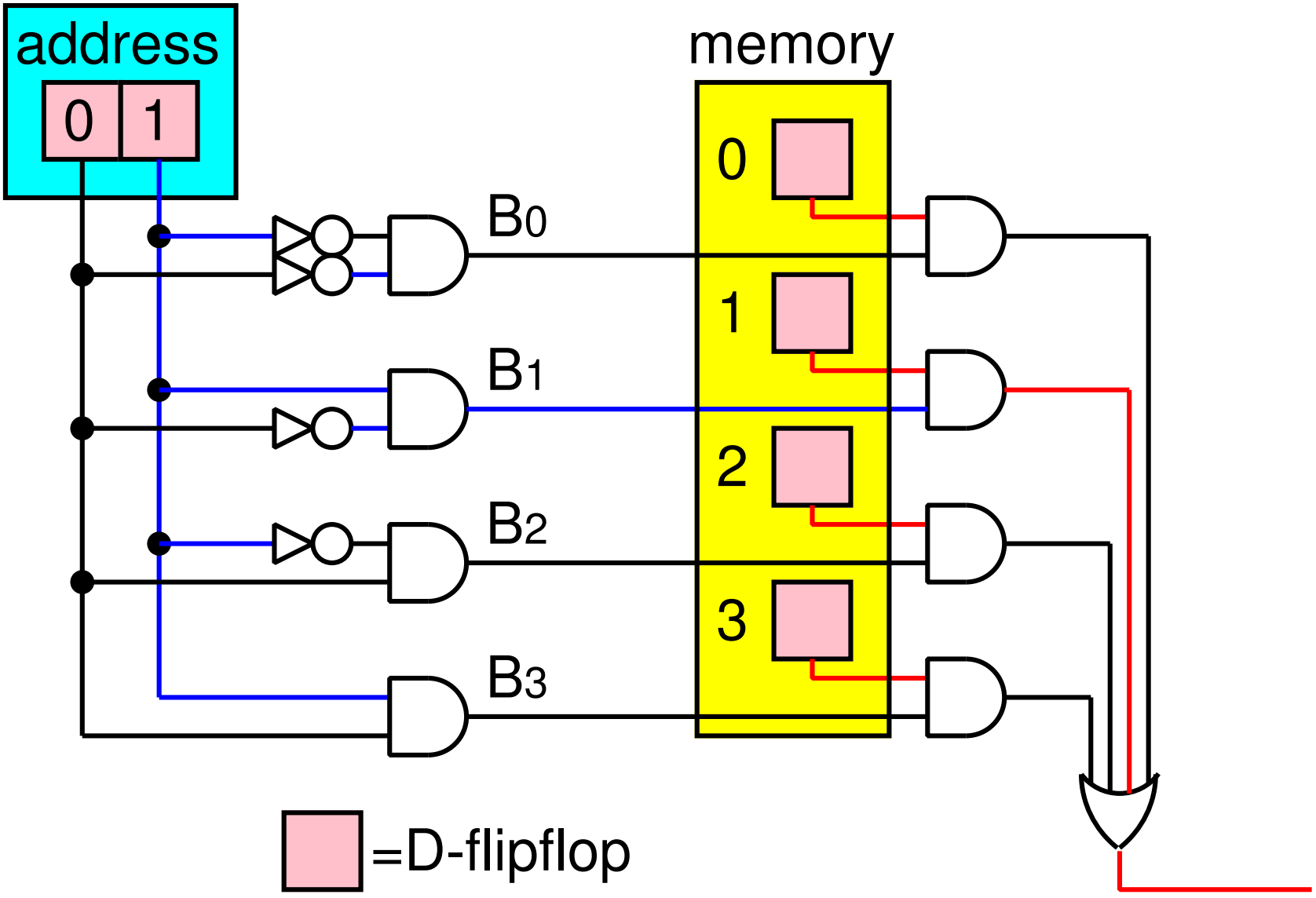
- どれを読み出すか
- どれに書き込むか

## $\longrightarrow$ 番地 (address) で管理

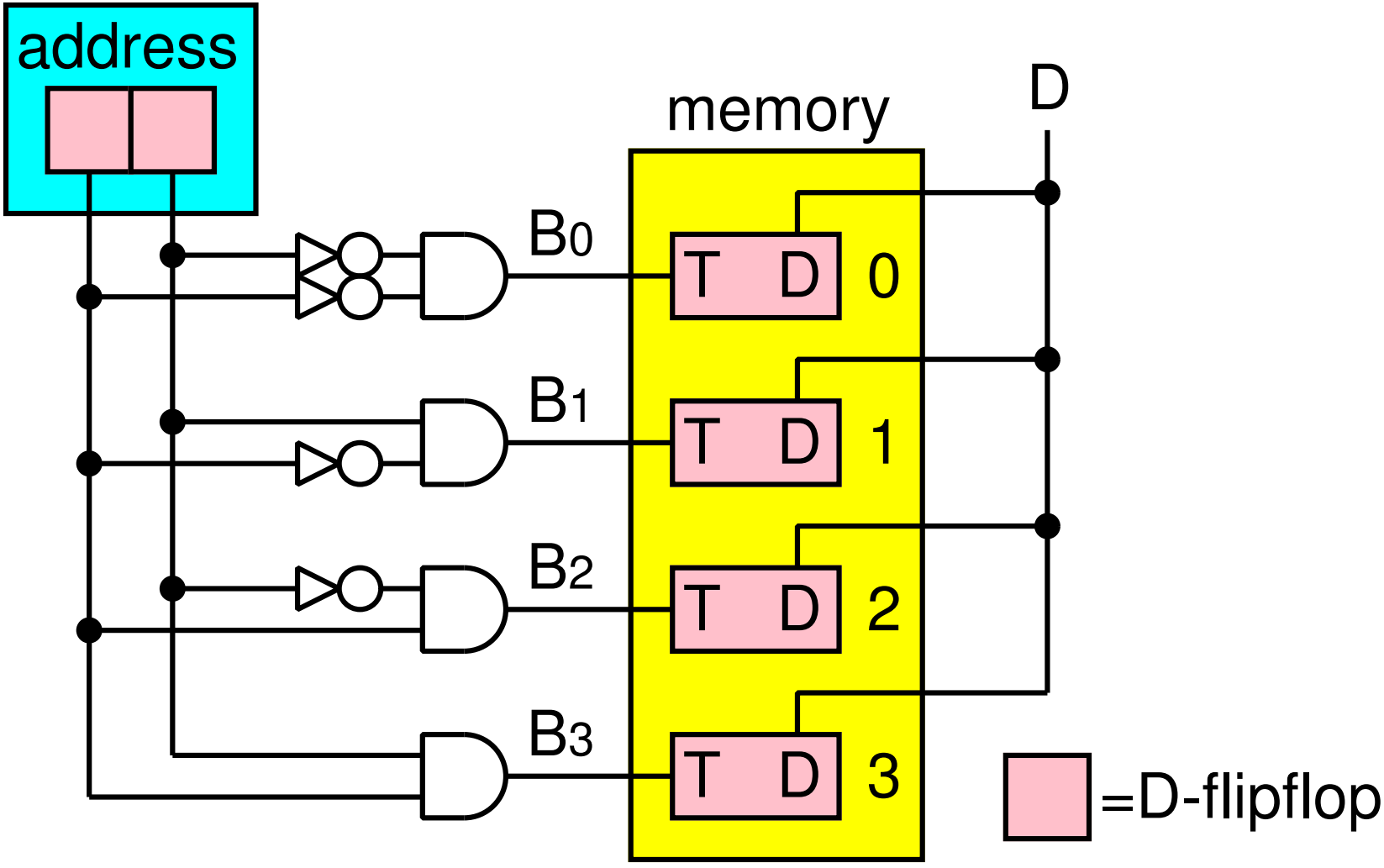
# データの読出し:



# データの読出し: 1 番地のデータを読出し



# データの書込み:



実際には、何 bit かで一まとまりとして、

データの読み書き・演算等の処理を行なう

$\longrightarrow$ 1 **語 (word)**

**(例:** 1 **word** $=$ 8 **bit**

$\Longrightarrow$ 8 **つを並列に並べておく)**

$\longrightarrow$ ○○ **bit CPU**

**(一度に処理できるデータの大きさを表す)**

# プログラム内蔵方式 (von Neumann 型)

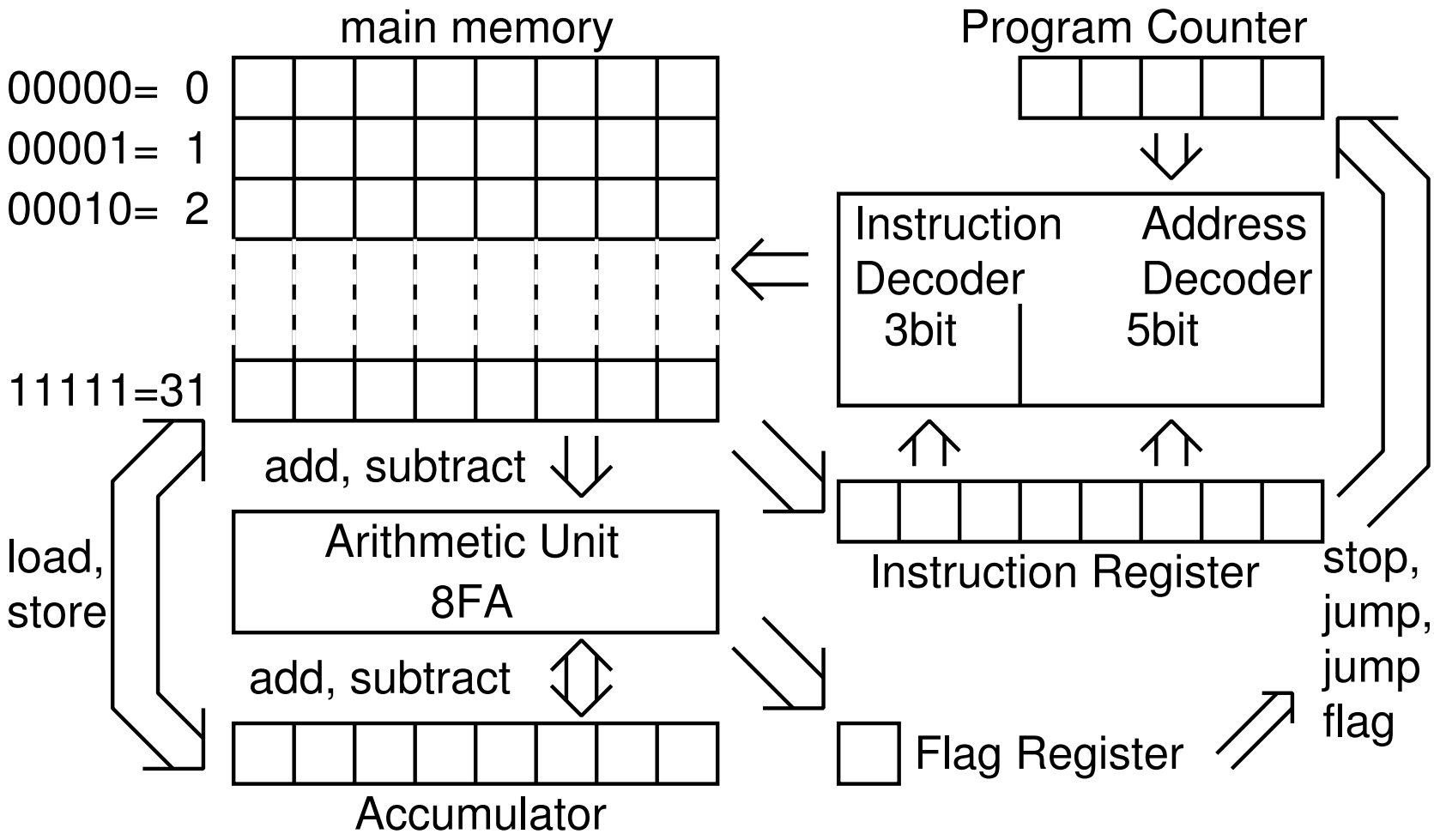
プログラム・データを共にメモリ上に置く

$\longrightarrow$ **主記憶装置**

実行の流れ: 以下を繰り返す

- プログラムを一命令づつ読み出す
- 命令の実行

# 説明用の簡易モデル:


main memory
Program Counter
00000= 0
00001= 1
00010= 2
11111=31
Instruction Decoder 3bit
Address Decoder 5bit
add, subtract
load, store
Arithmetic Unit 8FA
Instruction Register
stop, jump, jump flag
add, subtract
Flag Register
Accumulator