

計算機構成論 第2回

—計算機における数値表現(1)—

大連理工大学・立命館大学 国際情報ソフトウェア学部

大森 隆行

講義内容

■ プロセッサ・メモリを構成するもの

➡ ■ トランジスタ、論理素子、IC

■ 2進数

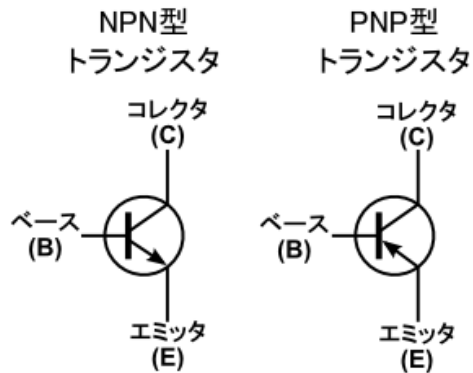
■ 2進数、16進数とは

■ 2進数と10進数の変換

■ 負数の表現、2の補数

プロセッサ・メモリを構成するもの

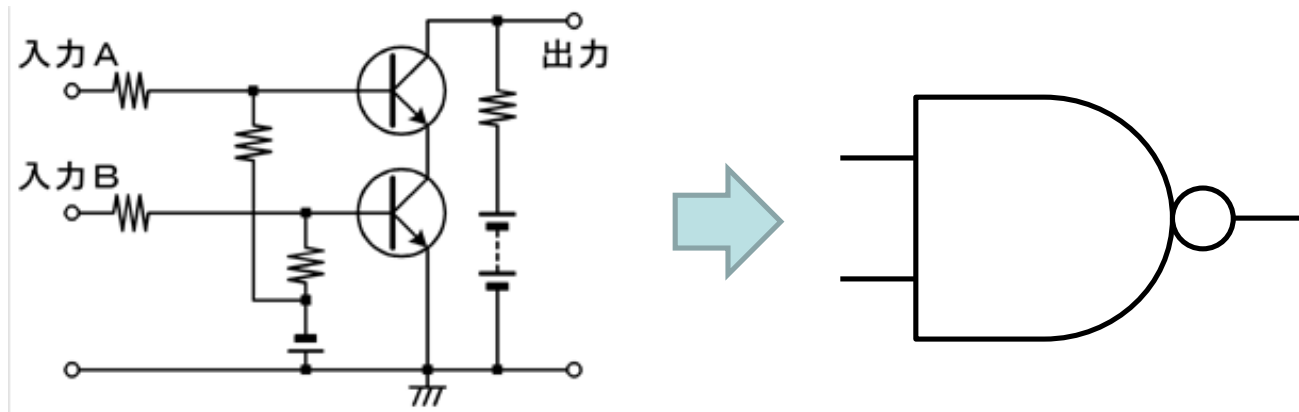
■ トランジスタ



- 電氣的なON/OFF切り替えができる
- N型半導体とP型半導体を組み合わせて作られる

プロセッサ・メモリを構成するもの

■ 論理ゲート(論理素子)

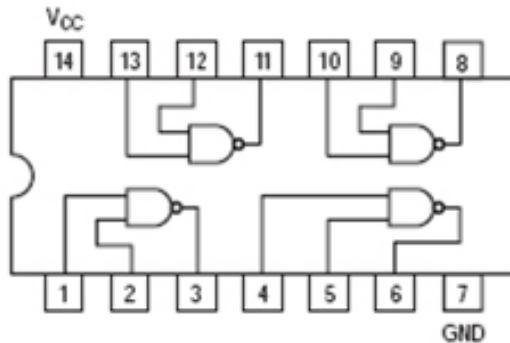


- 基本的な論理演算に対応する電子回路
- AND, OR, NOT, NAND, NOR, XORなど

プロセッサ・メモリを構成するもの

■ 集積回路 (integrated circuit)

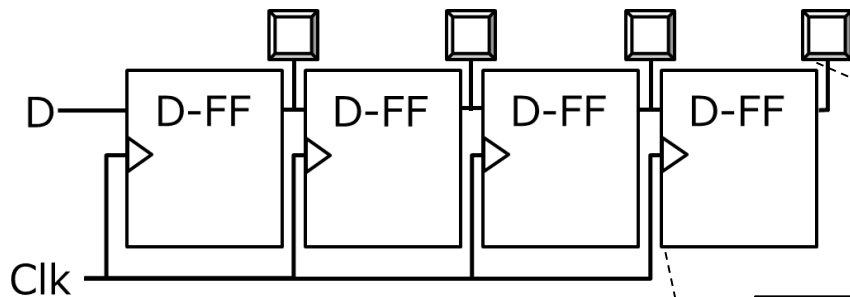
- 数十個から数百万個のトランジスタを集積した電子素子
- チップ (chip) とも呼ばれる



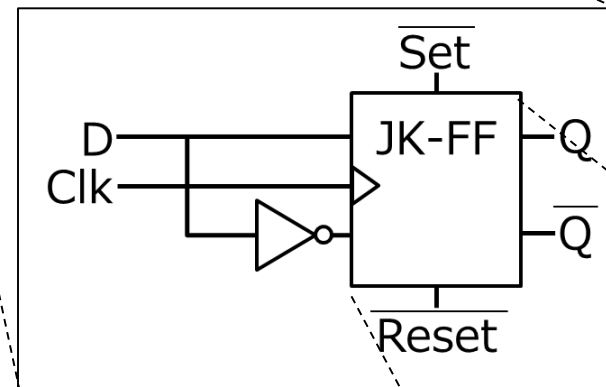
(例) NANDゲート4つで構成されたIC

プロセッサ・メモリを構成するもの

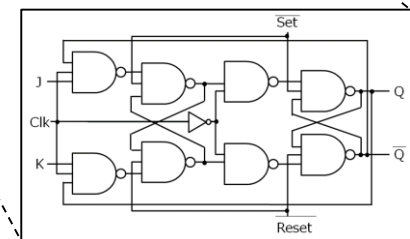
■ 記憶回路で情報を記憶 (例：レジスタ)



4bitシフトレジスタ



D-FF



JK-FF

プロセッサ・メモリを構成するもの

■ LSI (large scale integration)

■ 大規模集積回路

特に大規模なICのこと

■ 千～数十万個のトランジスタを集積

■ VLSI (very -)

■ 数十万～数百万個のトランジスタを集積

■ ULSI (ultra -)

■ 1000万個以上のトランジスタを集積

Pentium4チップの製造

- 1. シリコンインゴットを薄切り(~2.5mm)にしてウェーハを作成
- 2. ウェーハに一連の加工を施してトランジスタ、導体、絶縁体からなる回路を作成
- 3. ウェーハのテストを行う
(欠陥が見つかったものは製品として使わない)
- 4. ウェーハからダイ(チップ)を切り出す
- 5. ダイのテストを行う
(欠陥が見つかったものは製品として使わない)
- 6. ダイをパッケージ化
- 7. 部品テスト
- 8. 問題のなかったものを出荷

ソースコードから機械語へ

C言語

```
int a,b,c,d;  
a = b + c;  
d = a * b;  
...
```

そのままでは
実行できない

アセンブリ言語

```
lw    $9, 4($1)  
lw    $10, 8($1)  
add   $8, $9, $10  
...
```

コンパイラ
により変換

機械語

```
01001101  
00100100  
01001000  
00100000
```

そのまま回路上
で実行可能

- 高級言語のプログラムは、コンパイラ、アセンブラ等により機械語のコードに変換される
- アセンブリ言語と機械語の命令は1対1対応

確認問題

- AND、OR、NOTなどの論理を実現するための素子を (1) と呼ぶ。
- 次の用語を英語に直せ。
 - (2) 集積回路
 - (3) 大規模集積回路
- 高級プログラミング言語からアセンブリ言語のコードへの変換は、(4)が行う。
- アセンブリ言語から機械語への変換は、(5)が行う。



講義内容

■ プロセッサ・メモリを構成するもの

- トランジスタ、論理素子、IC

➡ 2進数

- 2進数、16進数とは
- 2進数と10進数の変換
- 負数の表現、2の補数

2進数、10進数、16進数

10進数	2進数	16進数
0	00000	00
1	00001	01
2	00010	02
3	00011	03
4	00100	04
5	00101	05
6	00110	06
7	00111	07
8	01000	08
9	01001	09
10	01010	0a
11	01011	0b
12	01100	0c
13	01101	0d
14	01110	0e
15	01111	0f
16	10000	10

2進数：

0～1で表現した数値

10進数：

0～9で表現した数値

16進数：

0～9,a～fで表現した数値

※大文字(A～F)でも良い

2進数とコンピュータ

- コンピュータ内部では、
電位(電圧: voltage)が高いか低いかを
1、0に割り当て



- コンピュータ内部のデータや命令は
すべて2進数で表現

123 (整数)	01111011
A (文字)	01000001
load word命令	00100011
35 (整数)	00100011

同じ？

同じ2進数でも、
いろいろな解釈がありうる
→解釈のルールが重要

2進数

■ 2進数：0と1のみで表現された数値

■ 基数は右下に小さく書く

■ 10進数の1111 → 1111_{10}

■ 2進数の1111 → 1111_2

最上位ビット

(MSB: most significant bit)

最下位ビット

(LSB: least significant bit)

2進数と10進数の変換

■ 2進数 → 10進数


$$\begin{aligned} \blacksquare 1111_{10} &= 1*1000 + 1*100 + 1*10 + 1*1 \\ &= 1*10^3 + 1*10^2 + 1*10^1 + 1*10^0 \end{aligned}$$

$$\begin{aligned} \blacksquare 1111_2 &= 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 \\ &= 1*8 + 1*4 + 1*2 + 1*1 \\ &= 15_{10} \end{aligned}$$

■ 10進数 → 2進数

$$11_{10} \rightarrow 1011_2$$

$11/2 = 5$	余り	1
$5/2 = 2$	余り	1
$2/2 = 1$	余り	0
$1/2 = 0$	余り	1


$$1011$$

2進数と16進数の変換

■ 2進数 → 16進数

- 4ビットずつ
区切って
右の表に
従って変換

例) 01011110_2
→ $5E_{16}$

■ 16進数 → 2進数

- 右の表に従って
4ビットずつ変換

16進数	2進数
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

16進数	2進数
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

2進数の負数表現 -絶対値表現-

10進数	2進数
-8	-
-7	1111
-6	1110
-5	1101
-4	1100
-3	1011
-2	1010
-1	1001
-0	1000

10進数	2進数
8	-
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

4ビットでは
表現不可能

符号(1:負数, 0:正数) + 絶対値

2進数の負数表現 -1の補数-

10進数	2進数
-8	-
-7	1000
-6	1001
-5	1010
-4	1011
-3	1100
-2	1101
-1	1110
-0	1111

10進数	2進数
8	-
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

4ビットでは
表現不可能

0と1を反転

2進数の負数表現 -2の補数-

10進数	2進数
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
-0	0000

10進数	2進数
8	-
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

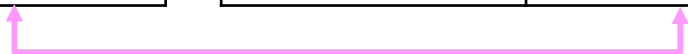
4ビットでは
表現不可能

1の補数+1

負数の2の補数を取ると？

10進数	2進数
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
-0	0000

10進数	2進数
8	-
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000



1の補数+1

2の補数の利点

- 最上位ビットを見ると、符号がわかる
 - 1→マイナス、0→プラス
- -0 がない
- 加算が簡単にできる
- 符号拡張も簡単にできる
 - 符号拡張：より多くのビットを使った値に変換すること(8bit→16bit)

2進数と10進数の変換

■ 2進数 → 10進数 (符号なし) 再掲

$$\begin{aligned} \blacksquare 1111_2 &= 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 \\ &= 1*8 + 1*4 + 1*2 + 1*1 \\ &= 15_{10} \end{aligned}$$

■ 2進数 → 10進数 (符号あり)

$$\begin{aligned} \blacksquare 1111_2 &= -1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 \\ &= -1*8 + 1*4 + 1*2 + 1*1 \\ &= -1_{10} \end{aligned}$$

2の補数変換の正当性

■ なぜビット反転して1足せば良いのか？

$$a_3a_2a_1a_0 + \sim a_3\sim a_2\sim a_1\sim a_0 = 1111_2$$

$$a_3a_2a_1a_0 + (\sim a_3\sim a_2\sim a_1\sim a_0 + 1) = \cancel{1}0000_2 = 0$$

$$(a_3a_2a_1a_0 + 1) + \sim a_3\sim a_2\sim a_1\sim a_0 = \cancel{1}0000_2 = 0$$

$$a_3a_2a_1a_0 = -2^3a_3 + 2^2a_2 + 2^1a_1 + 2^0a_0$$

$$\sim a_3\sim a_2\sim a_1\sim a_0 = -2^3\sim a_3 + 2^2\sim a_2 + 2^1\sim a_1 + 2^0\sim a_0$$

$$a_3a_2a_1a_0 + \sim a_3\sim a_2\sim a_1\sim a_0$$

$$= -2^3 + 2^2 + 2^1 + 2^0 = -1$$

確認問題

- (1) 符号なし整数 10101010_2 を10進数に変換せよ。
- (2) 符号なし整数 11110010_2 を16進数に変換せよ。
- (3) 23_{10} を8ビットの2進数に変換せよ。
- (4) 23_{10} を16進数に変換せよ。
- (5) -21_{10} を絶対値表現で8ビットの2進数に変換せよ。
- (6) -21_{10} を1の補数表現で8ビットの2進数に変換せよ。
- (7) -21_{10} を2の補数表現で8ビットの2進数に変換せよ。
- (8) 2の補数表現で、8ビットで表現できる数値の範囲を答えよ。





参考文献

- コンピュータの構成と設計 上 第5版
David A.Patterson, John L. Hennessy 著、
成田光彰 訳、日経BP社
- 山下茂 「計算機構成論 1」 講義資料
- 画像は著作権で保護されている可能性がありますので、
公開・頒布を禁止します。