

# 演習ガイド: Windows

1. 講義のウェブページから day2.zip をダウンロードする.
  - Downloads(ダウンロード)フォルダに day2.zip を置かれる
  - ファイルを開く → すべてを展開
  - 展開先として, Documents¥CS1 を指定して, 展開
2. コマンドプロンプトを実行
3. day2 フォルダに移動
4. 内容を確認

OneDriveを使用している場合, zipファイルの展開先が  
OneDrive¥ドキュメント¥CS1  
になるので下の最初のコマンドを変更

```
C:¥Users¥minamide> cd Documents¥CS1  
C:¥Users¥minamide¥Documents¥CS1> cd day2  
C:¥Users¥minamide¥Documents¥CS1¥day2> dir
```

...  
2020/10/19 08:58 445 add-alt.py  
2020/10/19 08:58 472 add.py

...

# 演習ガイド: Mac

1. 講義のウェブページから day2.zip をダウンロードする.
  - Downloads(ダウンロード)フォルダに day2.zip が展開される
2. Terminal を実行
3. Terminal で フォルダ day2 をCS1 に移動
4. day2 フォルダに移動
5. 内容を確認

```
$ cd Documents/CS1
$ mv ~/Downloads/day2 ./
$ cd day2
$ ls
add-alt.py  mult.py    smile.py
add.py      mult2.py   smile2.py
```

# CS第1 テーマ1

## テーマ1の目標

### 計算の基本要素を知る

## 本日の内容

1. データ = 数
2. コンピュータの中では

## 演習課題

### 四則演算でアニメーション

```
100000000000000000000000000000000000000000000000000000000000000  
100000000000000000000000000000000000000000000000000000000000000  
100000000000000000000000000000000000000000000000000000000000000  
100000000000000000000000000000000000000000000000000000000000000  
100000000000000000000000000000000000000000000000000000000000000  
100000001111111111110011000110001111111000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000001100000000000000000000000000000000000000000000000000000  
100000000110001100011111100011000110000000000000000000000000000  
100000000111001110000000111001110000000000000000000000000000000  
100000000011100000000000000000000000000000000000000000000000000
```

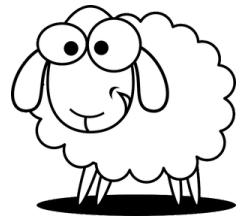


ちょいと苦しい

ひつじさん



ドンマイ



# 1. はじめに

CSのこころ

## すべては計算である

- ・  $(1+4) \times 5 =$
- ・ 12 と 16 の最大公約数
- ・  $x^2 + 2xy + y^2$  の因数分解
- ・ 原子炉の設計図を作成する
- ・ 遺伝子を解析する
- ・ 木の成長
- ・ 脳の形成
- ・ 銀行のATMの制御

ただし、コンピュータに  
載せるには ...

→  
必須

対象をデータとして表すこと  
処理を基本演算の組合せで表すこと

## 2. データは数である

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

・数        21, -5, 3.25, 1/3

0と1の列

・文字

・画像

・音

・映像

....

$$10\text{進数}: 21 = 2^4 + 2^2 + 1$$

II

$$2\text{進数}: 10101$$

・味, におい？？

## 2. データは数である

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

・数      21, -5, 3.25, 1/3

・文字       $a \leftarrow 01100001 (=97)$ ,  $b \leftarrow 01100010 (=98)$

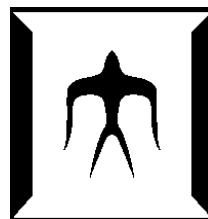
・画像

・音

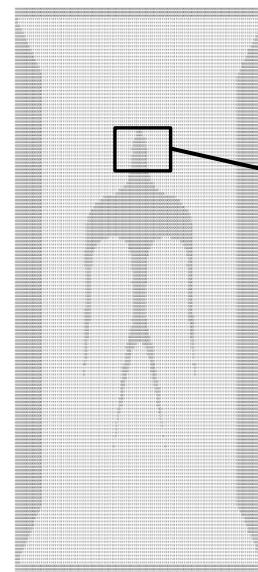
・映像

....

・味, におい？？



→



1111111111111111  
1111111111111111  
1111111111111111  
1111110011111111  
1111100001111111  
1111100001111111  
1111000000011111  
1111000000011111

0と1の列

ASCII という符号法

## 2. データは数である

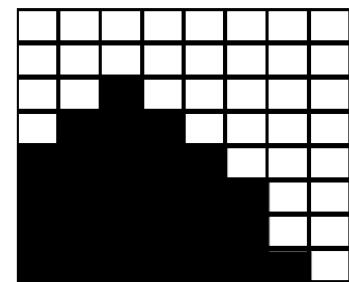
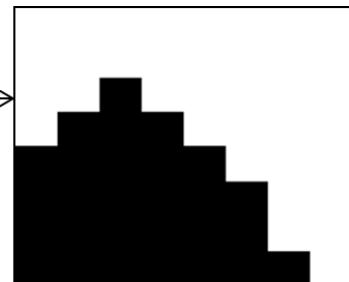
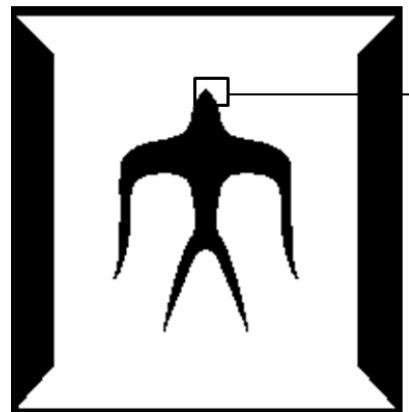
データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

- ・ 数
- ・ 文字
- ・ 画像



0と1の列

0と1の列

ツバメの一部を拡大してみた図（左）と  
さらに方眼メモリを当てはめた図（右）

デジタル化とは  
方眼紙をあてることなり

## 2. データは数である

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

・数      18, -5, 3.25, 1/3

0と1の列

・文字       $a \leftarrow 01100001$  (=97),  $b \leftarrow 01100010$  (=98)

ASCII という符号法

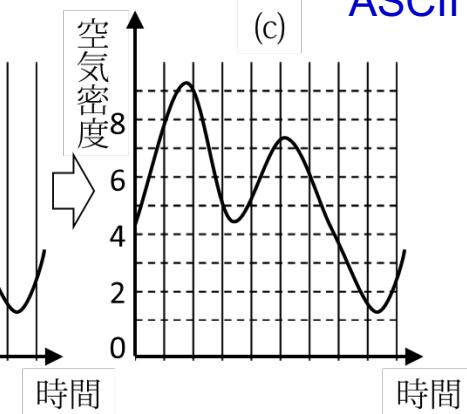
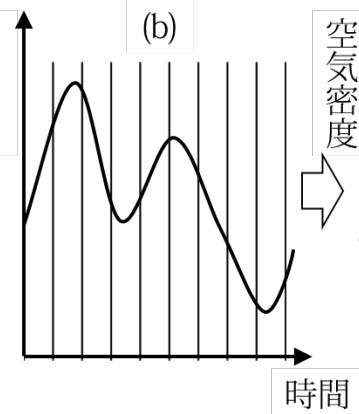
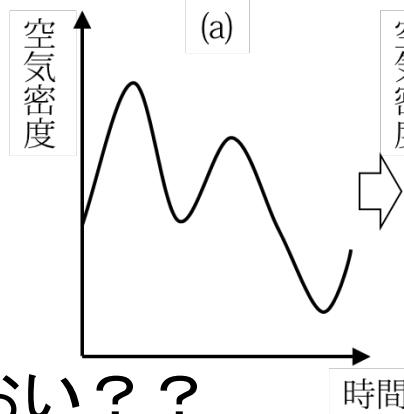
・画像

・音

・映像

....

・味, におい? ?



## 2. データは数である

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが ~~二進列~~

確かめてみよう（例で考える）

- ・ 数      18, -5, 3.25, 1/3
- ・ 文字      a ← 01100001 (=97)

- ・ 画像
- ・ 音
- ・ 映像
- ....

話を簡単に  
するため

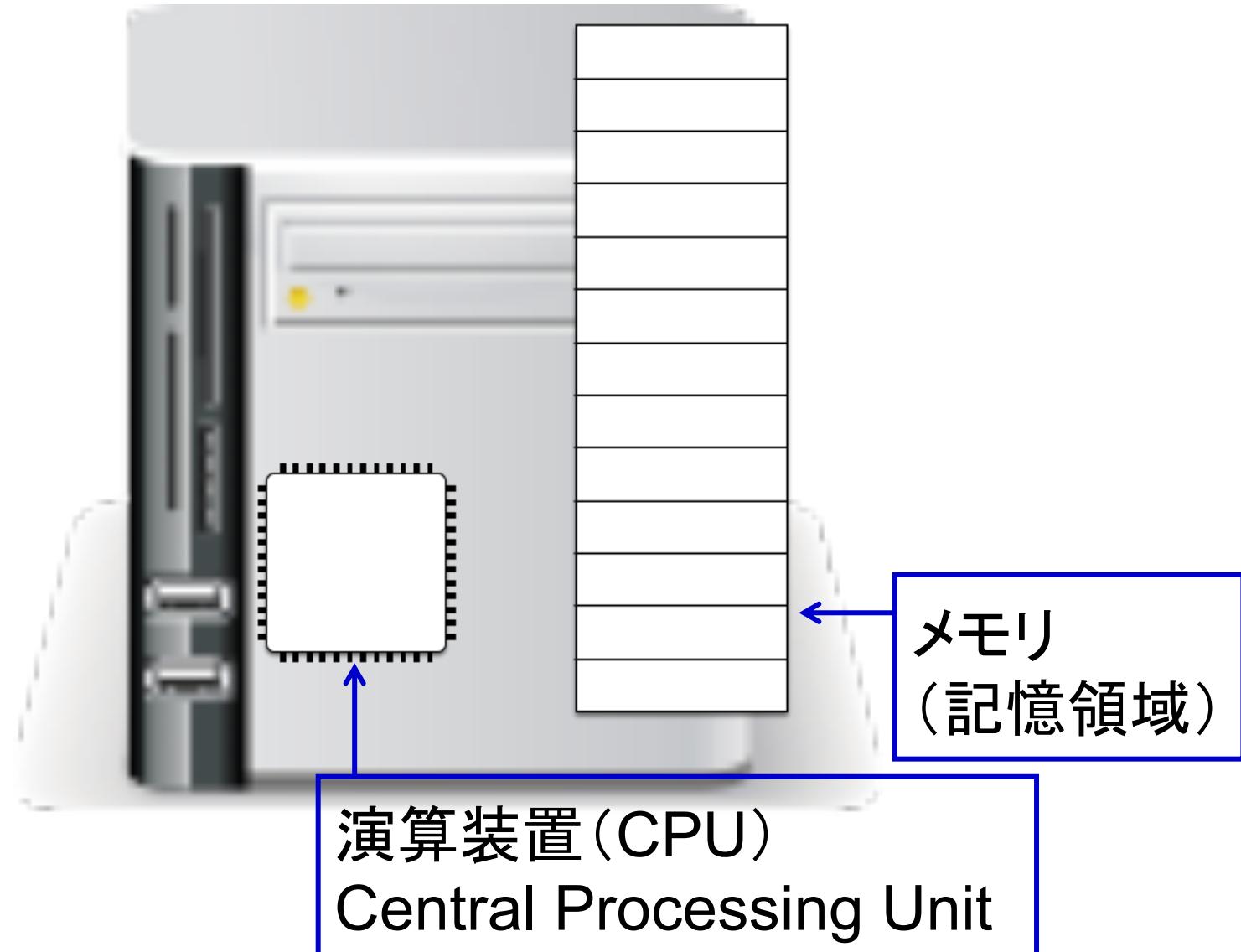
0と1の列

自然数

0, 1, 2, 3, ...

### 3. コンピュータの中では

コンピュータの基本 = 演算装置とメモリ



### 3. コンピュータの中では

メモリ = データ(数)を  
格納する箱



演算装置

|      |        |
|------|--------|
| 1234 | [0]    |
| 78   | [1]    |
| 0    | [2]    |
| 123  | [3]    |
| 9542 | [4]    |
| :    |        |
| 0    | [9998] |
| 0    | [9999] |

メモリ

データ

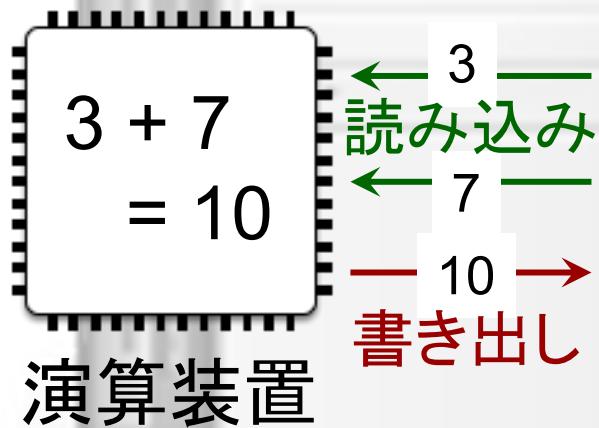
番地

### 3. コンピュータの中では

演算装置 = 基本演算を行った装置

プログラム

```
lw $x, M(100)  
lw $y, M(101)  
add $z, $x, $y  
sw $z, M(102)
```



|                   |        |
|-------------------|--------|
| lw \$x, M(100)    | [20]   |
| lw \$y, M(101)    | [21]   |
| add \$z, \$x, \$y | [22]   |
| sw \$z, M(102)    |        |
| ⋮                 |        |
| 0                 | [9998] |
| 0                 | [9999] |

メモリ

誰が指示するの？

プログラムも  
メモリに格納  
される

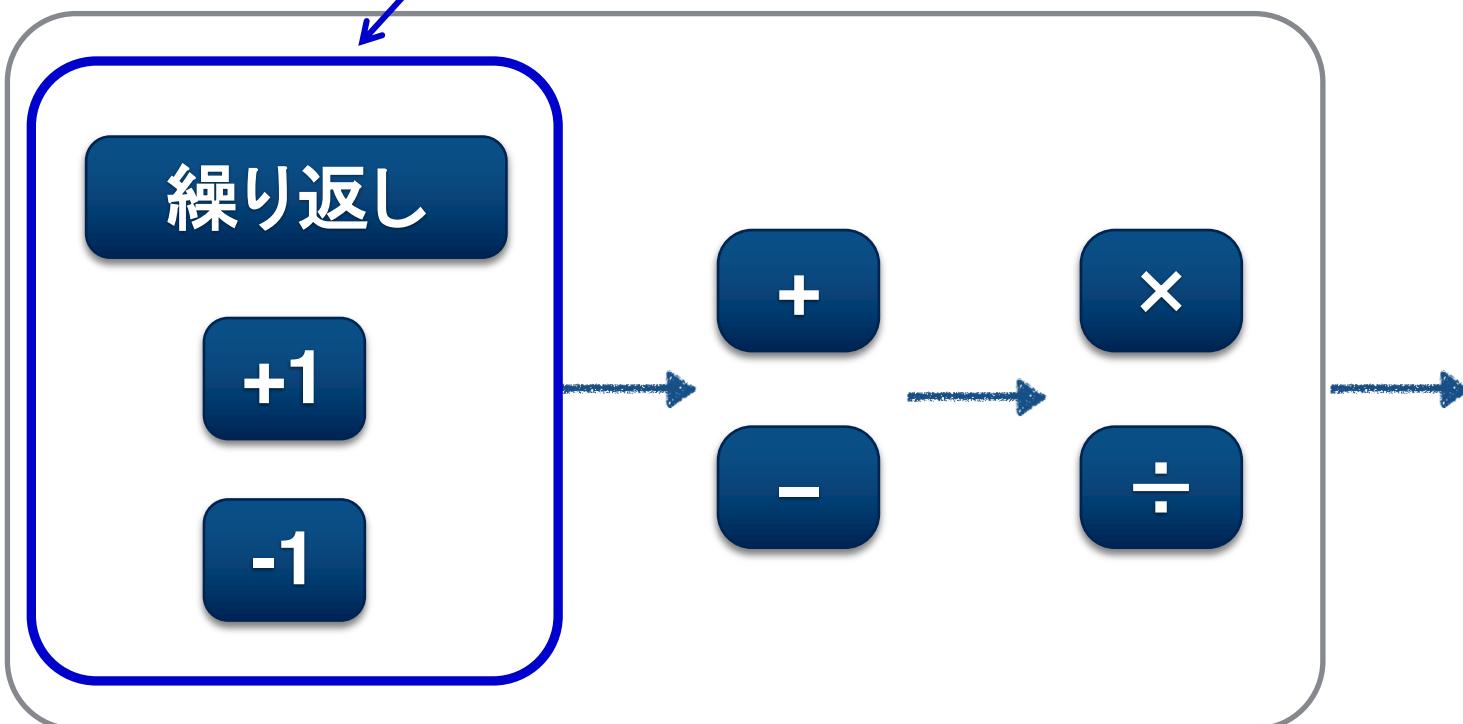
### 3. コンピュータの中では

演算装置 = 基本演算を行う装置  
とは？

計算の原子に  
相当する物は何？



チューリング

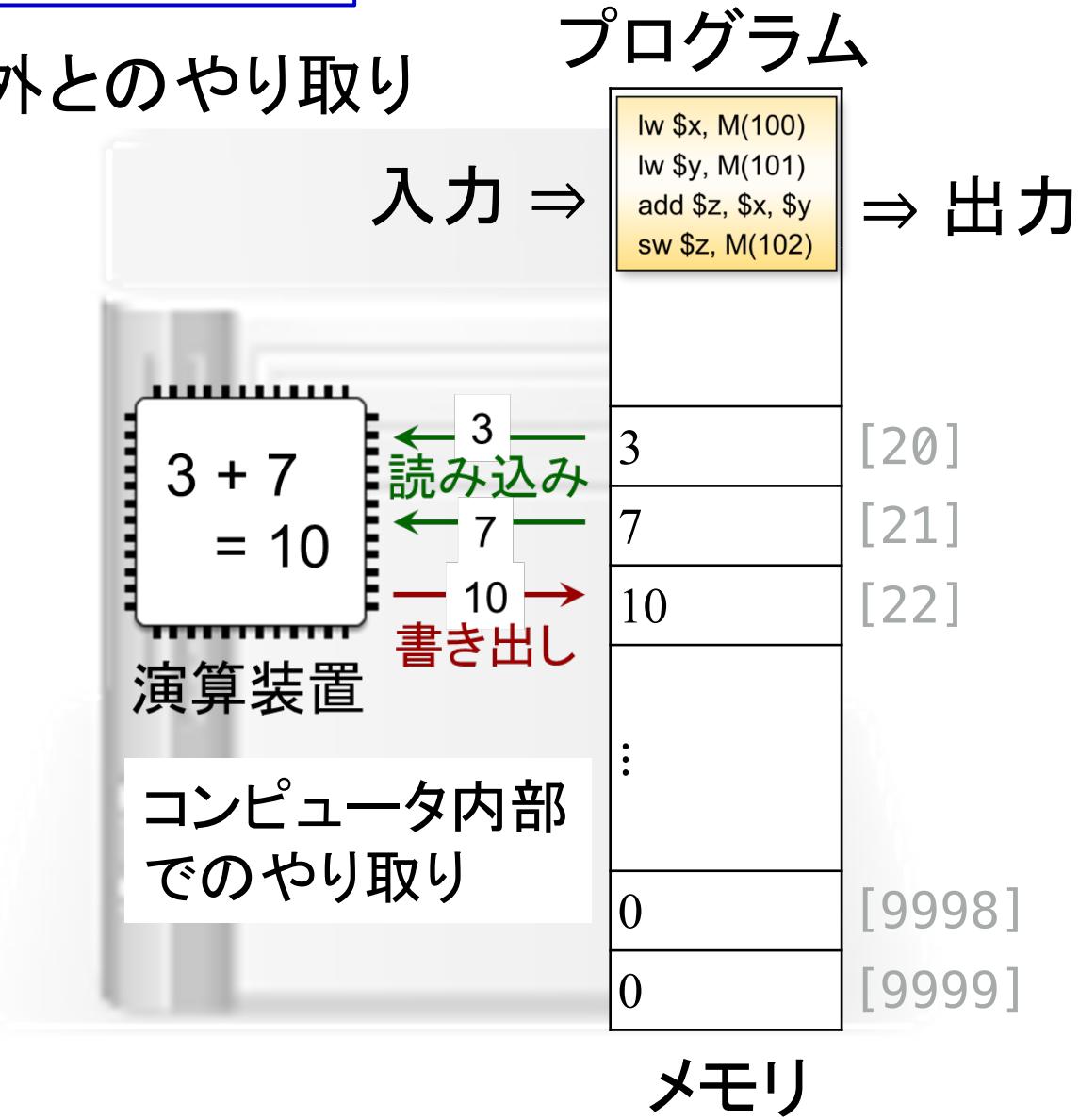


四則演算

囲碁, 将棋

### 3. コンピュータの中では

入出力 = 外とのやり取り

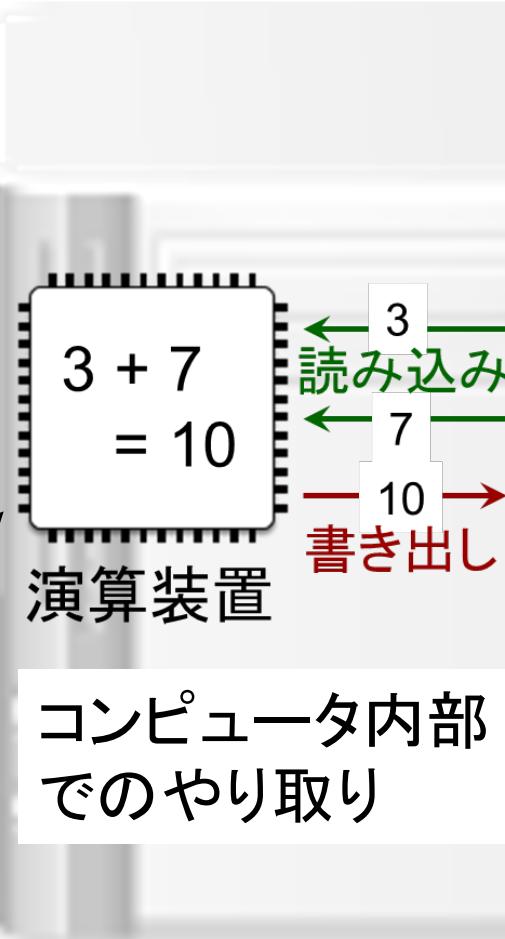


### 3. コンピュータの中では

入出力 = 外とのやり取り



インターネット



プログラム

```
lw $x, M(100)  
lw $y, M(101)  
add $z, $x, $y  
sw $z, M(102)
```

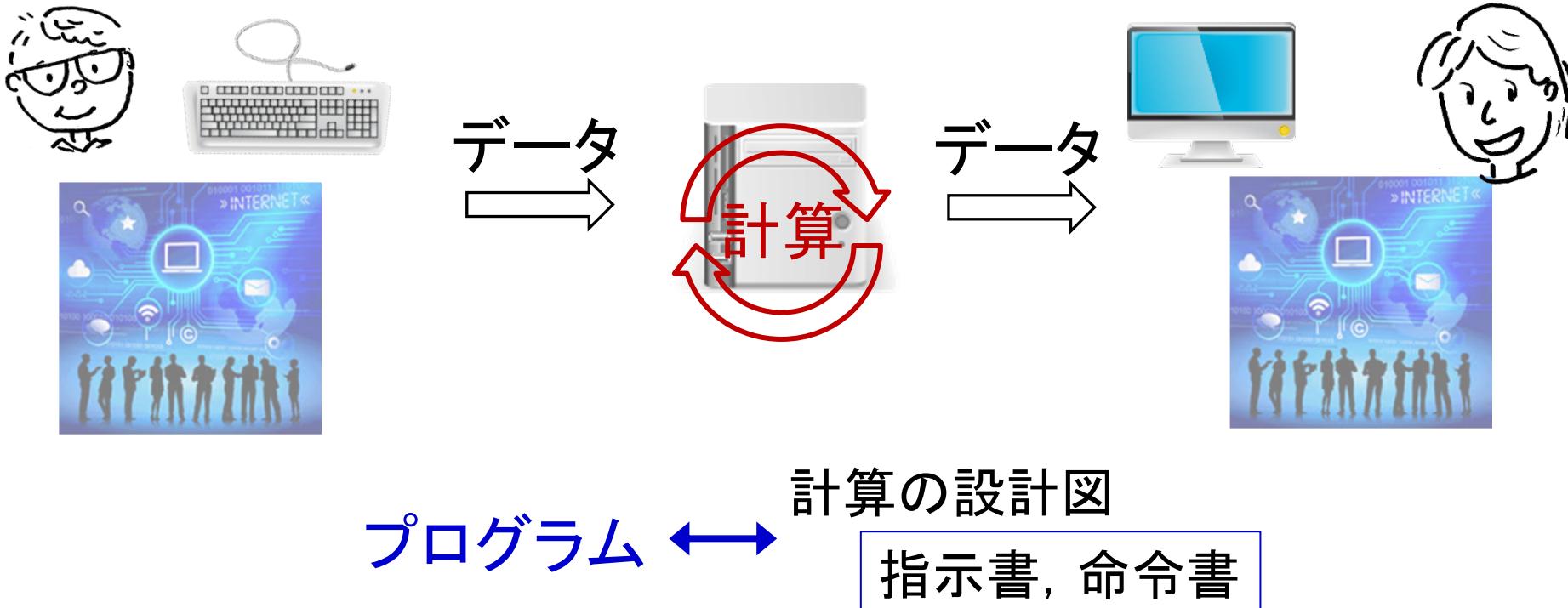
|    |        |
|----|--------|
| 3  | [20]   |
| 7  | [21]   |
| 10 | [22]   |
| :  |        |
| 0  | [9998] |
| 0  | [9999] |

メモリ

# ここまでまとめ

データ = 自然数

計算 = 超基本要素は ±1 と繰り返し



# 基本演算

掛け算を±1と繰り返しのみで実現する

add.py

```
a = 入力データ  
b = 入力データ  
wa = a  
while b > 0:  
    wa = wa + 1  
    b = b - 1  
print(wa)
```

mult.py

```
x = 入力データ  
y = 入力データ  
seki = 0  
while y > 0:  
    seki = seki + x  
    y = y - 1  
print(seki)
```

±1以外も使ってるよ

# 基本演算

掛け算を±1と繰り返しのみで実現する

mult.py

```
x = 入力データ  
y = 入力データ  
seki = 0  
while y > 0:  
    seki = seki + x  
    y = y - 1  
print(seki)
```

```
a = seki  
b = x  
wa = a  
while b > 0:  
    wa = wa + 1  
    b = b - 1  
seki = wa
```

```
x = 入力データ  
y = 入力データ  
seki = 0  
while y > 0:  
    a = seki  
    b = x  
    wa = a  
    while b > 0:  
        wa = wa + 1  
        b = b - 1  
    seki = wa  
    y = y - 1  
print(seki)
```

# プログラムを走らせてみる: Windows

```
C:\Users\minamide\Documents\CS1\day2>python mult.py
```

```
3
```

```
5
```

```
15
```

```
C:\Users\minamide\Documents\CS1\day2>chcp 65001
```

```
C:\Users\minamide\Documents\CS1\day2>type mult.py
```

```
# mult.py
```

```
# 入力: 自然数 x, y
```

```
# 出力: x × y
```

```
...
```

mult2.py を同様に実行してみる。

## プログラムを走らせてみる: Mac

```
$ python3 mult.py
```

```
3
```

```
5
```

```
15
```

```
$ cat mult.py
```

```
# mult.py
```

```
# 入力: 自然数 x, y
```

```
# 出力: x × y
```

```
...
```

mult2.py を同様に実行してみる。

# プログラムの実行が止まらないことがある

```
a = int(input())
b = int(input())
wa = a
while b > 0:
    wa = wa + 1
    b = b - 1
print(wa)
```

```
$ python add.py
3
0
3
$ python add.py
3
-1
3
```

止まる  
正しい加算の答えではない

```
a = int(input())
b = int(input())
wa = a
while b != 0:
    wa = wa + 1
    b = b - 1
print(wa)
```

```
$ python add-alt.py
3
0
3
$ python add-alt.py
3
-1
```

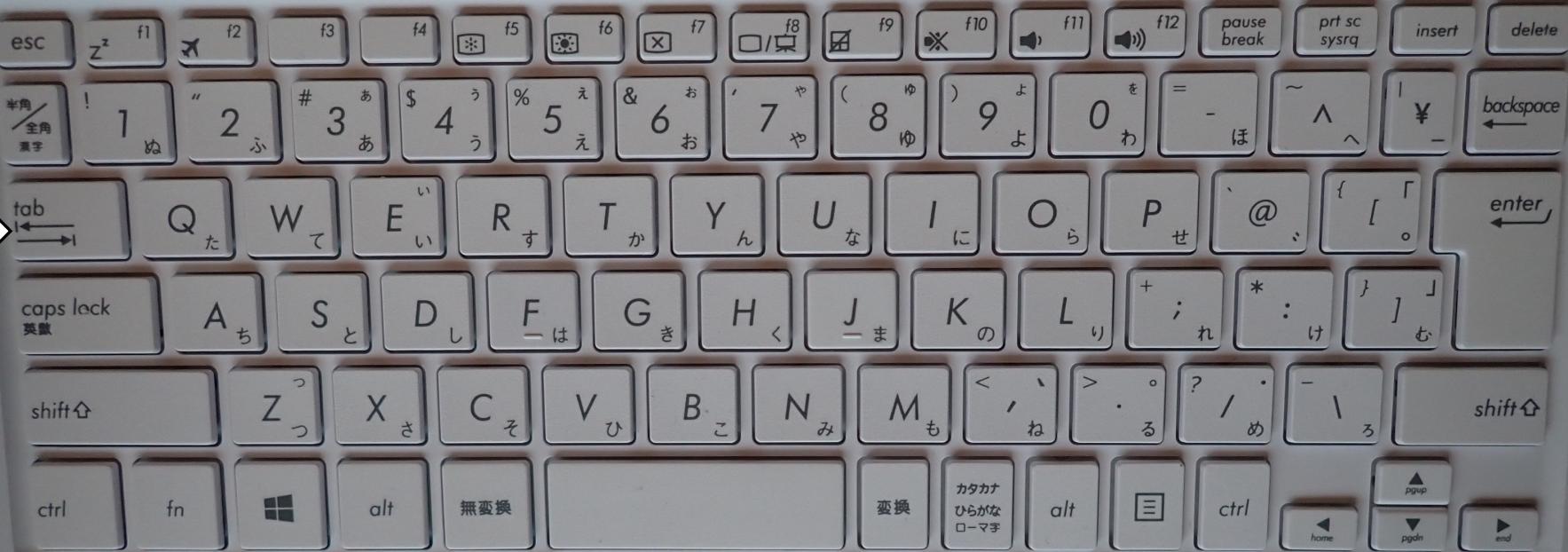
止まらない  
Windows: control +  
break(pause)  
Mac: control + c で止める

ASUS

pause/break



タブ



HDMI

お問い合わせ窓口  
ASUS コールセンター<sup>TEL 0800-123-2787</sup>  
<sup>www.asus.com/TE 0570-783-884</sup>  
<sup>http://www.asus.com/japan</sup>

# Terminal コマンド

| 命令<br>(Mac) | 使用例                           | windows       | 意味                 |
|-------------|-------------------------------|---------------|--------------------|
| pwd         | pwd                           | cd            | 現在いるフォルダを表示        |
| mkdir       | mkdir day2                    |               | day2 というフォルダを作る    |
| cd          | cd day2                       |               | day2 というフォルダに入る    |
|             | cd ..                         |               | 上のフォルダに戻る          |
|             | cd ../../                     | cd ..¥..      | 上の上のフォルダに戻る        |
| ls          | ls                            | dir           | そのフォルダにあるファイルを表示する |
| cat         | cat test.txt                  | type          | テキストファイルの内容を表示     |
|             |                               | chcp<br>65001 | 文字コードを Unicode に変更 |
| mv          | mv a.txt b/                   | move          | ファイルの移動            |
| cp          | cp a.txt b¥<br>cp a.txt b.txt | copy          | ファイルのコピー           |