

# 医用データ科学| 現代のコンピュータ概要

芳賀 昭弘

# 医用データ科学とコンピュータ

AI・機械学習を含む、医用データ科学による医療支援と課題解決、次世代システム開発



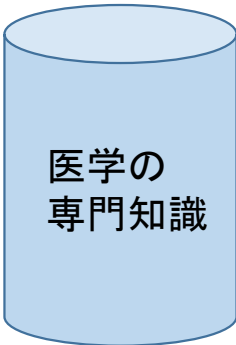
統計学

高等数学  
物理学等の基礎自然科学  
...



コンピュータ・  
プログラミング

- 情報理論
- 現代コンピュータの構造
  - ハードウェア
  - ソフトウェア
  - ネットワークとセキュリティ
- AI・機械学習手法
  - 回帰と分類、クラスタリング



医学の  
専門知識

解剖学・生理学  
放射線診断・治療  
...

# デジタル情報（復習）

- ビット(bit)とバイト(byte)
- 単位の接頭辞
- 二進数とn進数

# デジタル情報(復習)

## 数値データの表現法

### (1) 整数型

#### 1) 符号なし整数と符号付整数

##### ① 符号なし整数

8ビットの符号なし整数では、0～255を表現できる。

32ビットの符号なし整数では、0～4,294,967,295を表現できる。

##### ② 符号付整数

最上位ビットで正負を表す。

8ビットの符号付整数では、-127～127を表現できる。

32ビットの符号付整数では、-2,147,483,647～2,147,483,647を表現できる。

##### ③ 負の数の補数表示

### (2) 実数型

#### 1) 固定小数点表示

有効桁

アンダーフロー

オーバーフロー

#### 2) 浮動小数点表示

$a \times R^b$

a: 仮数

b: 指数

IEEE754:浮動小数演算規格

単精度:32ビット

倍精度:64ビット

# デジタル情報(復習)

## 論理演算

### NOT否定

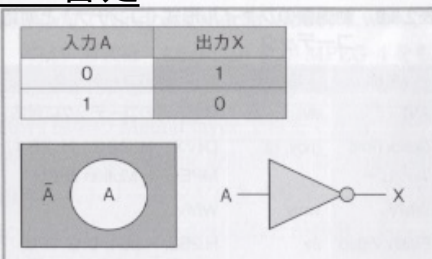


図2.3.9 NOT演算のベン図とMIL記号

### 論理積(AND)

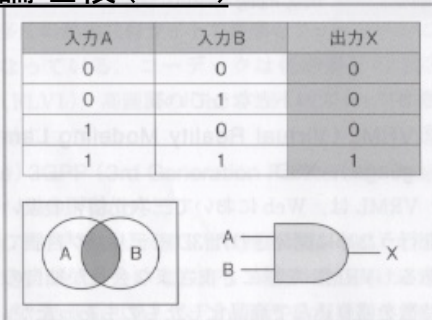


図2.3.10 AND演算のベン図とMIL記号

### 論理和(OR)

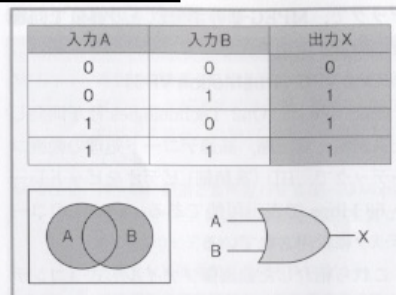


図2.3.11 OR演算のベン図とMIL記号

### 否定論理積(NAND)

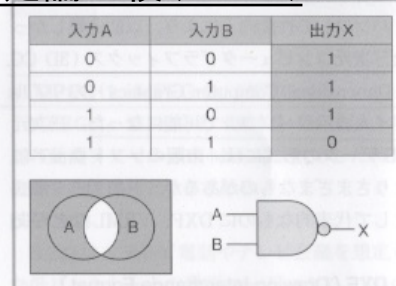


図2.3.12 NAND演算のベン図とMIL記号

### 否定論理和(NOR)

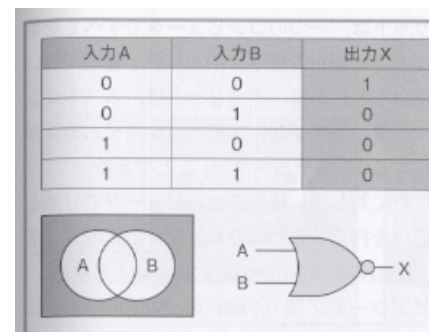


図2.3.13 NOR演算のベン図とMIL記号

### 排他的論理和(XOR)

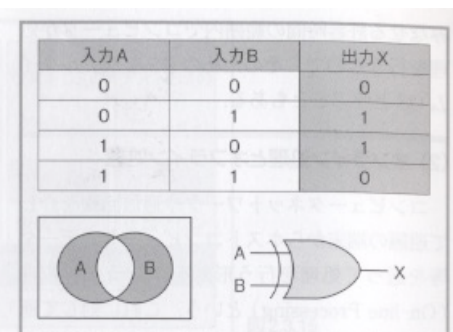


図2.3.14 XOR演算のベン図とMIL記号

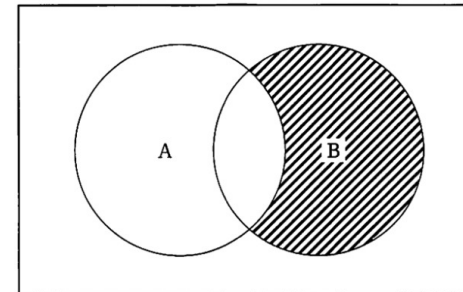
## 演習：昨年度の診療放射線技師国家試験問題

情報の数の表現で正しいのはどれか

1. 1Gbitは1,024 Mbyteである
2. 1 byteは10進数の8桁分である
3. 16進数は0から15までの整数で表す
4. アスキーコードは数値データの表現形式である
5. 2 byteの符号なし整数型は0から65,535の値を表現できる

集合A, Bからなるベン図で射線部分を表すのはどれか

1. NOT A
2. A XOR B
3. A NAND B
4. NOT(A XOR B)
5. (NOT A) AND B

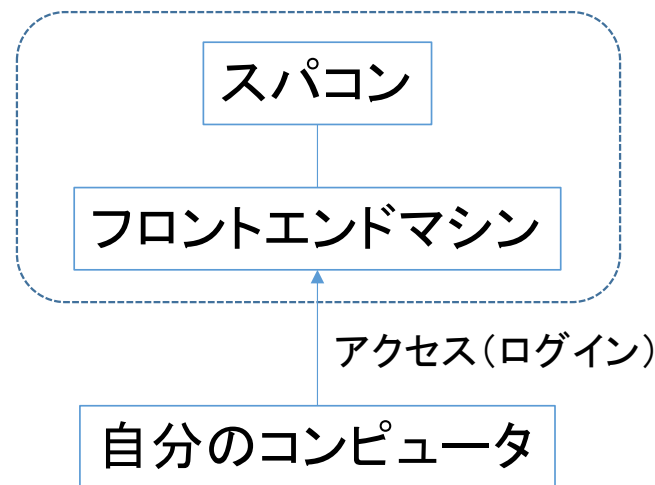


コンピュータのハードウェア（復習）

# コンピュータの種類：スーパーコンピューター



大規模な大学・民間の研究機関、  
共同研究施設に設置



スパコンを利用するには、設置機関に利用申請を行い承認を受ける必要がある(基本有償)。

## ログイン画面の例

```
Cybermedia Center, Osaka University
Welcome to a large-scale computing system !!

Select the system number of the application

[0] The information from the center.
[1] Frontend Terminal (fronta) >> online user(s) : 32
[2] Frontend Terminal (frontb) >> online user(s) : 21
[3] Frontend Terminal (frontc) >> online user(s) : 12
[4] Frontend Terminal (frontd) >> online user(s) : 12
[8] Super Computer (sx8)
[G] Gftp Client (gftp2)
[99] Logout (Session Close)

Enter number ? 3
Last login: Sat May 3 12:36:16 2014 from login1

=====
# COMMAND      FUNCTION                                     #DATE#
about-sx9       新スーパーコンピュータSX-9について             090203
about-sx8       新スーパーコンピュータSX-8について             070105
about-sx9-class SX-9のジョブクラスについて         100405
about-pcc       PCクラスターのサービスについて                 090422
about-futankin  新食料金表について                             120912
about-nkf       ファイルの文字コード変換について               070115
about-gaus      Gaussian03/09の利用について                     110428
about-job       ジョブの利用について                           110428
about-riyo      利用状況の表示コマンドについて                 110428
about-sxf90     sx-f90コマンドについて                         080704
=====
Topics
上記コマンドは、/usr/local/bin の配下にあります。

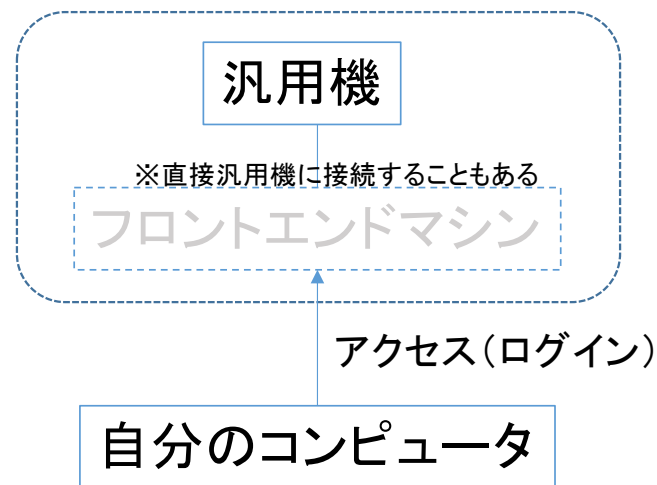
=====
フロントエンド端末 (fronta ~ frontd) は、クロスコンパイルなど
スパコン関連の前後処理および、AVS等のアプリケーションにご利用ください。
※演算機としての用途がある場合は、NQSを用いた PCクラスターサービス
をご利用ください。
```



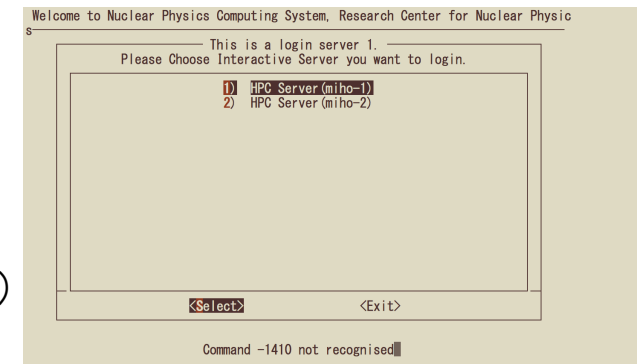
# コンピュータの種類：汎用機・メインフレーム



大学・病院・民間企業施設に設置

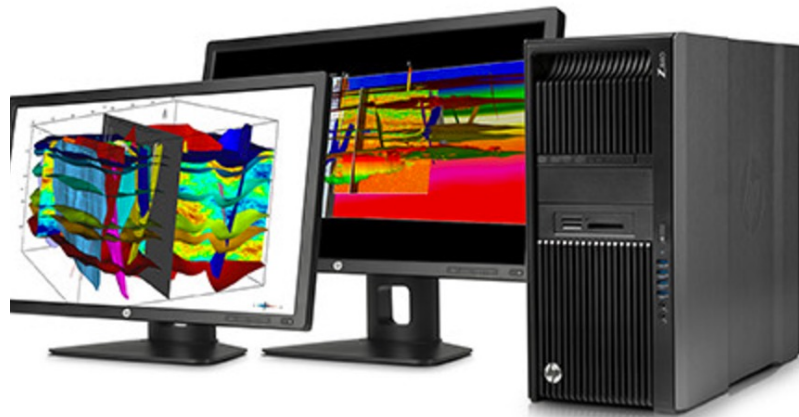


## ログイン画面の例

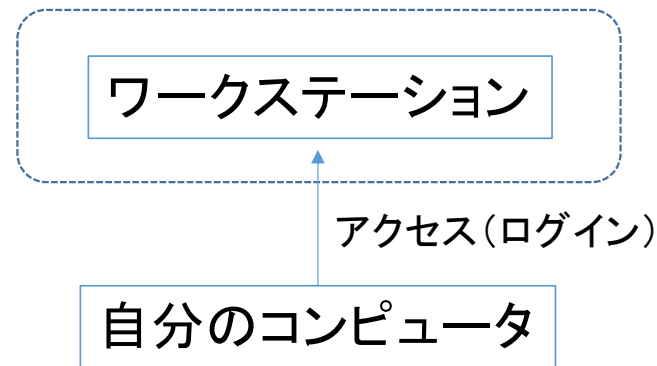


設置施設に所属しているものが申請・承認を得て利用する。

# コンピュータの種類：ワークステーション



研究室・部門毎に設置



ワークステーションの管理者に申請・承認を得て利用する。

# コンピュータの種類：パーソナルコンピュータ



個人／部門で所有・利用  
デスクトップ型とノート型

文書作成、表計算、プレゼン資料作成  
メール、WEB閲覧など

スパコン、汎用機、ワークステーション等  
に接続する”端末機”として利用

# コンピュータの種類：PCサーバ



PCタワー

研究室・部門毎に設置

サーバとは、ある特定のサービスを提供するコンピュータのこと。  
PCサーバは、サーバ機能をパーソナルコンピュータで実現する。

サーバ機能の例；

1. ファイルサーバ・・・データ保存を提供
2. WEBサーバ・・・WEB作成・閲覧許可機能を提供
3. メールサーバ・・・メール授受機能を提供
4. データベースサーバ・・・データベース機能を提供
5. 計算機サーバ・・・規模の大きい計算機能を提供

# コンピュータの種類：その他



問診、現場における記録などにおいて、  
病院内での利用が増加

個人／部門で所有・利用

# コンピュータの処理形態

## リアルタイム処理とバッチ処理

- リアルタイム処理（普通のPCでのジョブの処理）
  - インタラクティブ処理ともいう
  - PCで処理を行うようにコマンドを入力して実行・処理
  - サーバでは、リアルタイム処理を行う時間を制限していることが多い
- バッチ処理（サーバでのジョブの処理）
  - サーバに処理を依頼して実行する方法
  - 大規模な（本番の）実行を行うときに利用
  - リアルタイム処理より長い実行時間を許容されている

バッチ処理がなぜ必要か？

大勢の利用者と共同で利用するコンピュータでは、利用者間の公平性や利用率向上の目的のため、“ジョブの順番待ち制度”を設けている

例：スパコンのバッチ処理キュー

バッチキューの設定について

• バッチシステム：PBS Professional (Atlas社)

• 主要コマンド

- ジョブの投入： `qsub [ジョブスクリプトファイル名]`
- 投入したジョブの状況確認： `rbstat`
- 投入ジョブの削除： `qdel [ジョブID]`
- バッチキューの詳細構成を見る： `rbstat -rsc -x`
- 投入されているジョブ数を見る： `rbstat -b`
- 過去のジョブ投入履歴を見る： `rbstat -H`

# コンピュータの処理形態

## オンライン処理とオフライン処理

- オンライン処理
    - ネットワークに繋がりながら処理を行う形態
  - オフライン処理
    - ネットワークを介在せずに処理を行う形態
    - 例として、USBなどでデータを別のコンピュータに移動して処理を行うこと、など
- ※IT化が遅れている医療でも...  
紙媒体によるカルテを利用していた時代はオフライン処理が一般的だったが、  
今では電子カルテの普及も進み、オンライン処理が一般的になりつつある。

# コンピュータの処理形態

## 集中処理と分散処理

- 集中処理
  - 1つのコンピュータで全ての処理を行うこと
- 分散処理
  - 複数のコンピュータで分散して処理を行うこと
  - 水平分散と垂直分散



# コンピュータの処理形態

## シンクライアント (thin client)

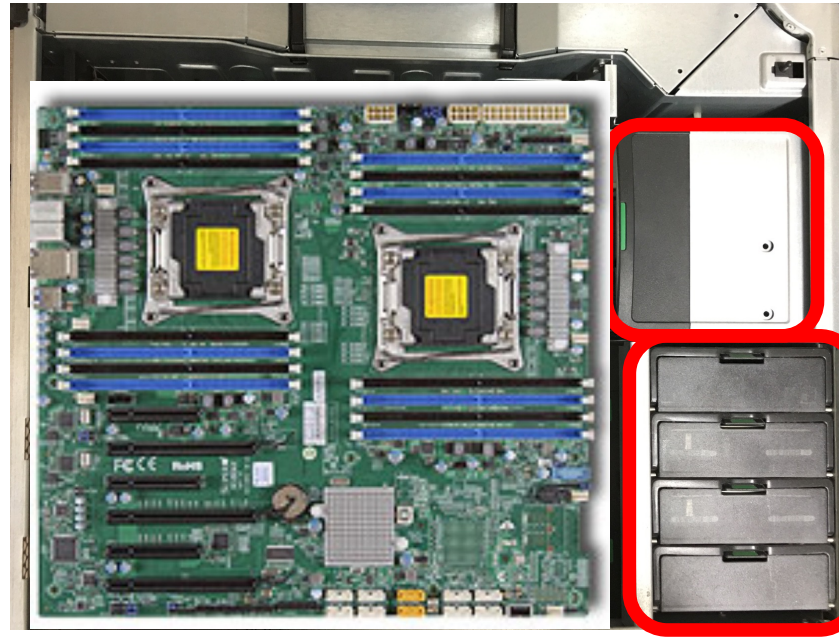
- 画像転送型サーバベース方式
- ブレードPC方式
- 仮想PC方式
- ネットワークブート方式

# ハードウェア

メモリ



CPU



DVD/CD ドライブ



ハードディスク・ドライブ:HDD



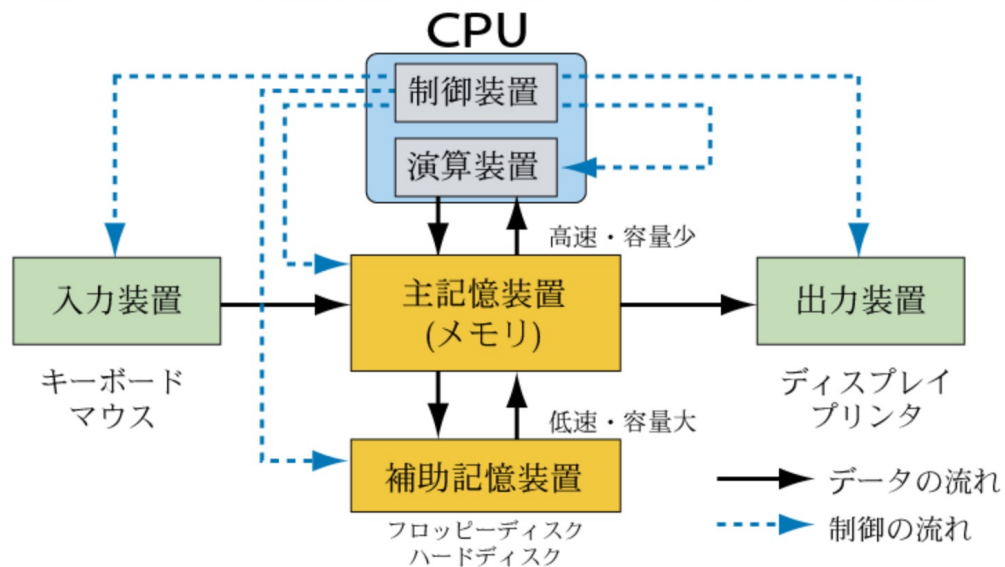
GPU





## パソコンのハードウェア

### ● パソコンの動作原理と，基本ハードウェアの構成



## CPU

### (1) 制御装置・演算装置

#### 1) CPUの動作原理

##### ① CPUの基本動作

##### 命令サイクル

- ・ 命令取り出し段階
- ・ 命令実行段階

##### 4つのステージ

1. F: メインメモリからの読み込み(フェッチ)
2. D: 命令の解読(デコード)
3. E: 有効アドレスの計算と命令の実行(エグゼキューション)
4. WB: 演算結果のメインメモリへの書き戻し(ライトバック)

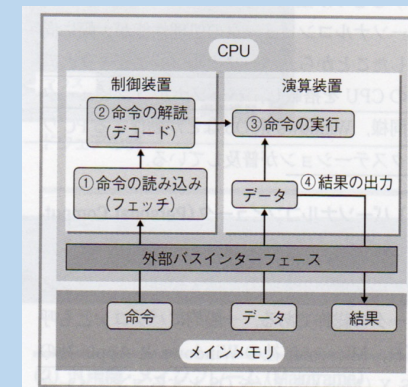


図 2.1.3 CPUの動作原理

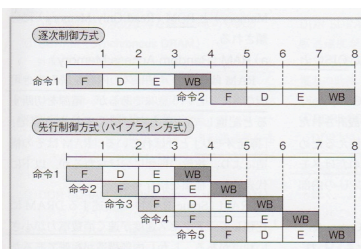
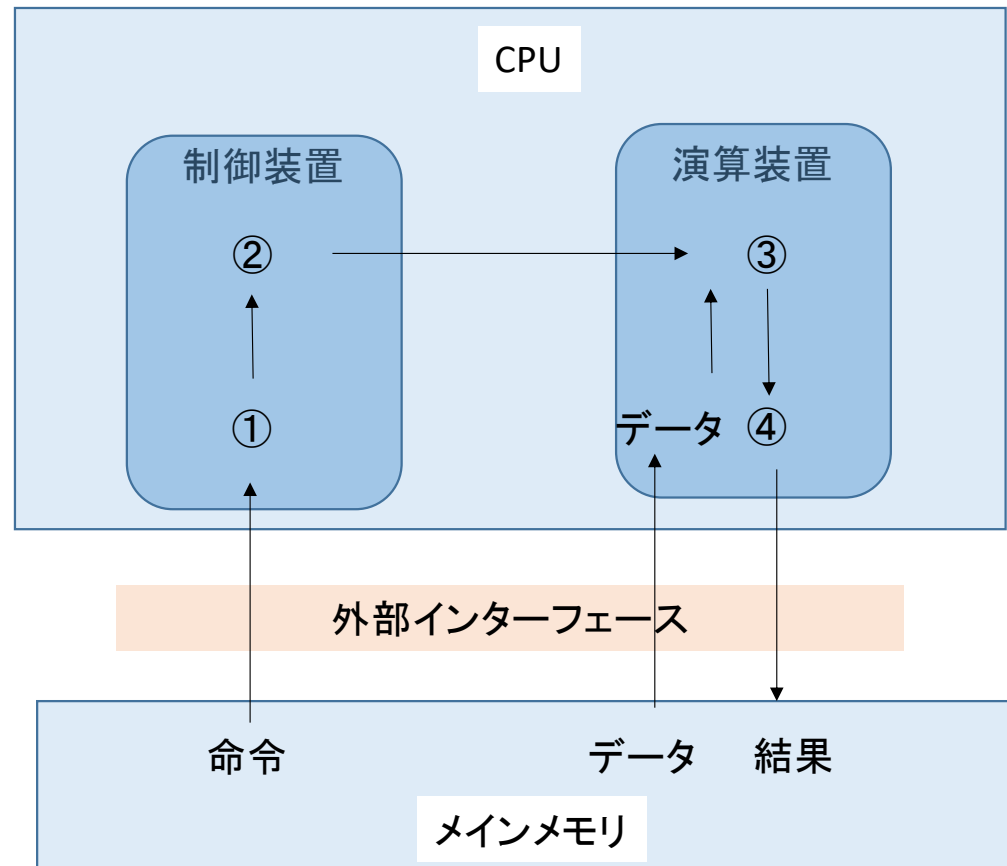


図 2.1.4 CPUの命令実行の方法  
(アイティメディア株式会社Webサイト「@IT」より引用)

## CPU制御方式

演習: 次の図で①～④に当てはまるものはどれか

エグゼキュート  
フェッチ  
デコード  
ライドバック



# CPUとGPU

## ➤ CPU (Central Processing Unit)

- パソコンの処理で中心的な役割を担っている演算処理を行なう半導体チップ



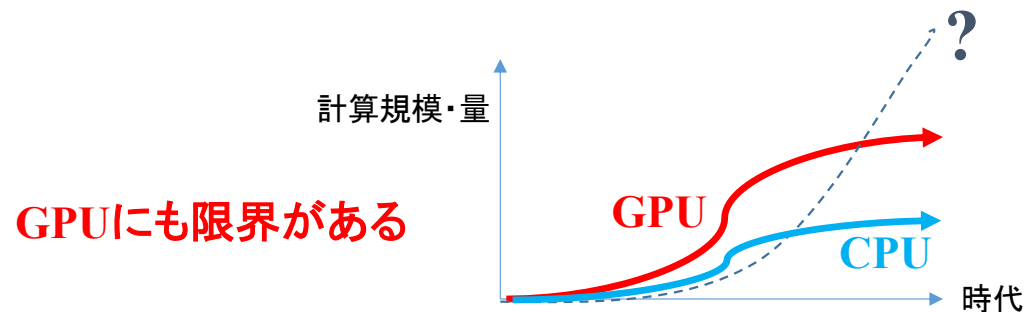
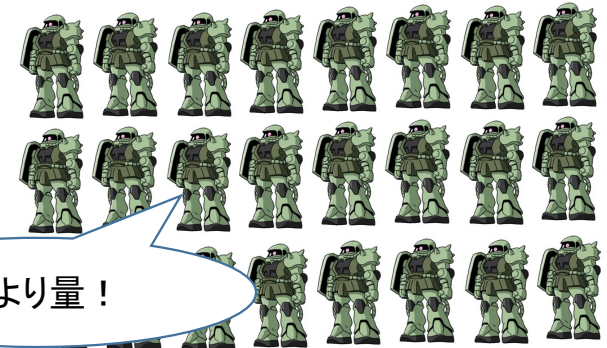
## ➤ GPU (Graphics Processing Unit)

- もともとピクセル毎の処理のために搭載した膨大な数のグラフィカルプロセッサをグラフィックス以外の目的に転用

### 賢い人少し (CPU)



### 普通の人沢山! (GPU)



# 量子コンピューティング Quantum Computing

「自然は古典力学では動かない」  
「自然をシミュレーションしたいならば量子論に基づく計算機を作るべき」— R.P. Feynman



## 2022年のノーベル物理学賞に 「量子もつれ」の研究者3人

2022年10月5日

アラン・アスペ、ジョン・クラウザー、アントン・ツァイリンガー



量子の世界の情報の最小単位

古典ビット

$\uparrow 0$  or  $\downarrow 1$   $x \in \{0, 1\}$

量子ビット

$q$   $\uparrow \downarrow$   $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

0と1の重ね合わせ状態

複素ベクトル空間

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

測定 0 or 1 ?

$$p_0 = |\alpha|^2$$

$$p_1 = |\beta|^2$$

絶対値の2乗が確率



量子もつれ  
(重ね合わせ)



超並列化の実現



# コンピュータ言語

## ■低水準言語

- 機械語、アセンブリ言語

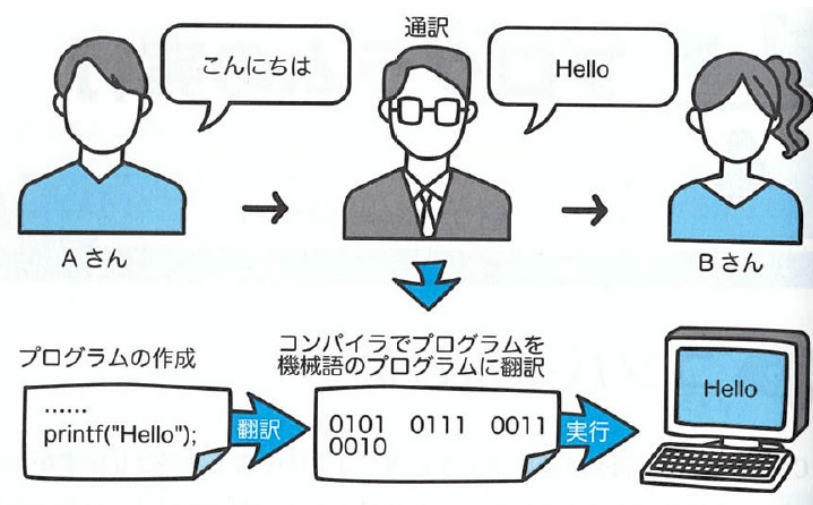
## ■高水準言語

- コンパイラ型 (Fortran, C言語など)

- インタプリタ型 (R, python, Perlなど)

## 例:C言語プログラミング手順

1. プログラムをファイルに書く
2. そのプログラムをコンパイルする
3. コンパイルにより作成された実行ファイルを実行する



# コンパイラ言語でのプログラムの実行

プログラムをエディタで書く

```
1 // rule.c
2 #include<stdio.h>
3 int main(void)
4 {
5     printf("C言語ははじめました\n");
6     return 0;
7 }
8 |
```

ターミナルで

% gcc rule.c → gccの直後にスペース、rule.c

% ./a.out → ドット(.)スラッシュ(/)a.out (全て繋げて入力)

と入力してみよう(%は打たないこと)

ターミナル出力

「C言語ははじめました」

エディタ: プログラムや文書を作成・編集するアプリケーション (プログラム作成にはVS code, emacs, Xcodeなどが良い)

ターミナル: ファイルやディレクトリの編集したり、プログラムを実行することができるアプリケーション

(linuxやmacは備付, windowsはVS codeでターミナルを開ける。windows terminalやcygwinもOK)

次週からインタプリタ型言語であるpythonを使います