

コンピュータは どうやって動くのか？

コンピュータのしくみ ～スマホからスパコンまで～

五島 正裕

DATE : 2016/08/25

概要

■ 概要(コピペ)

今やコンピュータは身の回りに溢れかえっています。ところがコンピュータは、ほぼ完全な「ブラック・ボックス」です。そのしくみをちゃんと理解して使っている人は、実は、情報のプロの中にもほとんどいません。

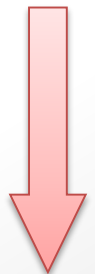
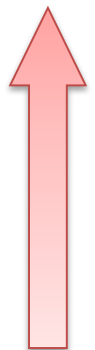
しくみをちゃんと理解するには、大学の情報系の学部でも2年はかかり、数時間の講義ではまったく不可能なことです。

そこでこの講義では、基礎から応用まで、以下のようなトピックをつまみ食いして、コンピュータはどうやって動いているのか、何となく分かったような気になることを目標とします。

大学・大学院 カリキュラム と 本日の内容

学年	講義名	本日の内容
大学院	アーキテクチャ 特論	スーパーコンピュータ
学部 3年	学生実験 コンピュータを作る	?
	コンピュータ(計算機) アーキテクチャ	
学部 2年	ディジタル回路 (論理回路)	二進数, 加算器
	電子回路	トランジスタ, 半導体

後半



前半

つまみ食いのポイント

■ 前半：

- ◆ 0 と 1 で動くってどういうこと？
- ◆ なぜ半導体で作るの？

■ 後半：

- ◆ コンピュータって、たとえば？
- ◆ x86 とか ARM アーキテクチャって何？
- ◆ スパコン, ポスト「京」のポイントは？

二進数

位取り(くらいどり)

■ (大きな)数字をどのように(書き)表すか？

◆ 各言語の伝統的な記数法

- 千九百六十八
- one thousand nine hundred sixty eight

◆ 位取り記数法

- 1968

■ 位取り記数法のメリット

◆ 数字を単純に並べて表す

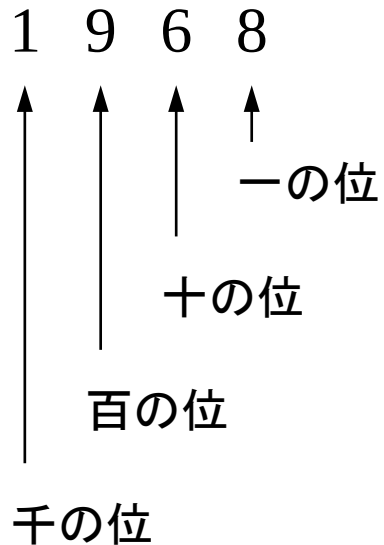
- 短い
- 大きい位を表す単語を定義しなくてよい
 - (億・兆・京・..., million・billion・trillion・...)

n 進表現

十進表現

各桁は 0, 1, 2, 3, ..., 8, 9 の十種類

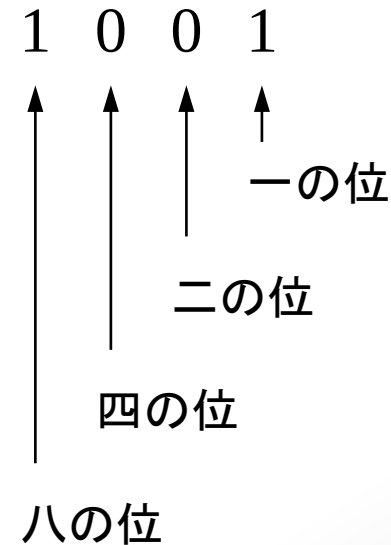
千 + 九百 + 六十 + 八 = 千九百六十八



二進表現

各桁は 0, 1 の二種類

八 + 一 = 九



二進数の加算

$$\begin{array}{r} 0 \\ +) 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ +) 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ +) 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ +) 1 \\ \hline 1\ 0 \end{array}$$

桁上げ

部分和

$$\begin{array}{r} 1 \\ 0\ 1 \\ +) 0\ 1 \\ \hline 1\ 0 \end{array}$$

$$\begin{array}{r} 1\ 1 \\ 0\ 1 \\ +) 1\ 1 \\ \hline 1\ 0\ 0 \end{array}$$

$$\begin{array}{r} 1\ 1 \\ 1\ 1 \\ +) 0\ 1 \\ \hline 1\ 0\ 0 \end{array}$$

$$\begin{array}{r} 1\ 1 \\ 1\ 1 \\ +) 1\ 1 \\ \hline 1\ 1\ 0 \end{array}$$

← 桁上げ

- 十進で $999 + 1$
- 二進で $111 + 1$
- で、桁上がりが走る感を説明

加算器の真理値表

桁上げ

部分和

x	y	c_{out}	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

半加算器 (half adder)

十進数だと, 百行 !

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

全加算器 (full adder)

十進数だと, 千行 !

論理ゲート

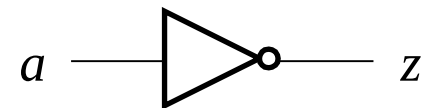
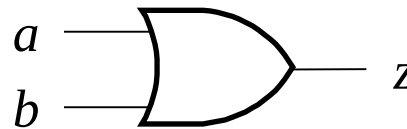
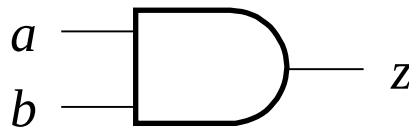
論理ゲート

AND
(論理積)

OR
(論理和)

NOT
(論理否定)

MIL 記号



真理値表

a	b	z
0	0	0
0	1	0
1	0	0
1	1	1

a	b	z
0	0	0
0	1	1
1	0	1
1	1	1

a	z
0	1
1	0

この3種類で、あらゆるデジタル回路を作ることができる！

余談: MIL 記号

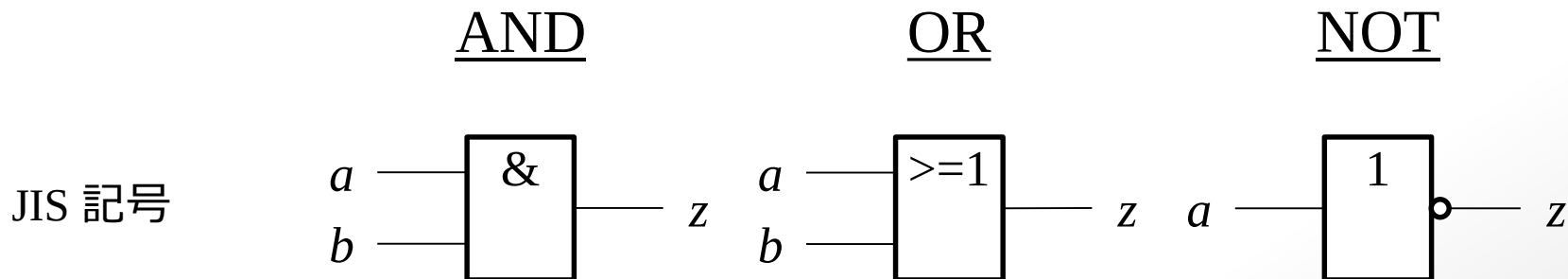
■ MIL 記号

◆ MIL : 米国軍用規格 (Military Standard)

■ JIS 記号

◆ 美しくない, 直感的に分かりにくい

◆ 誰も使わない, 知られていない

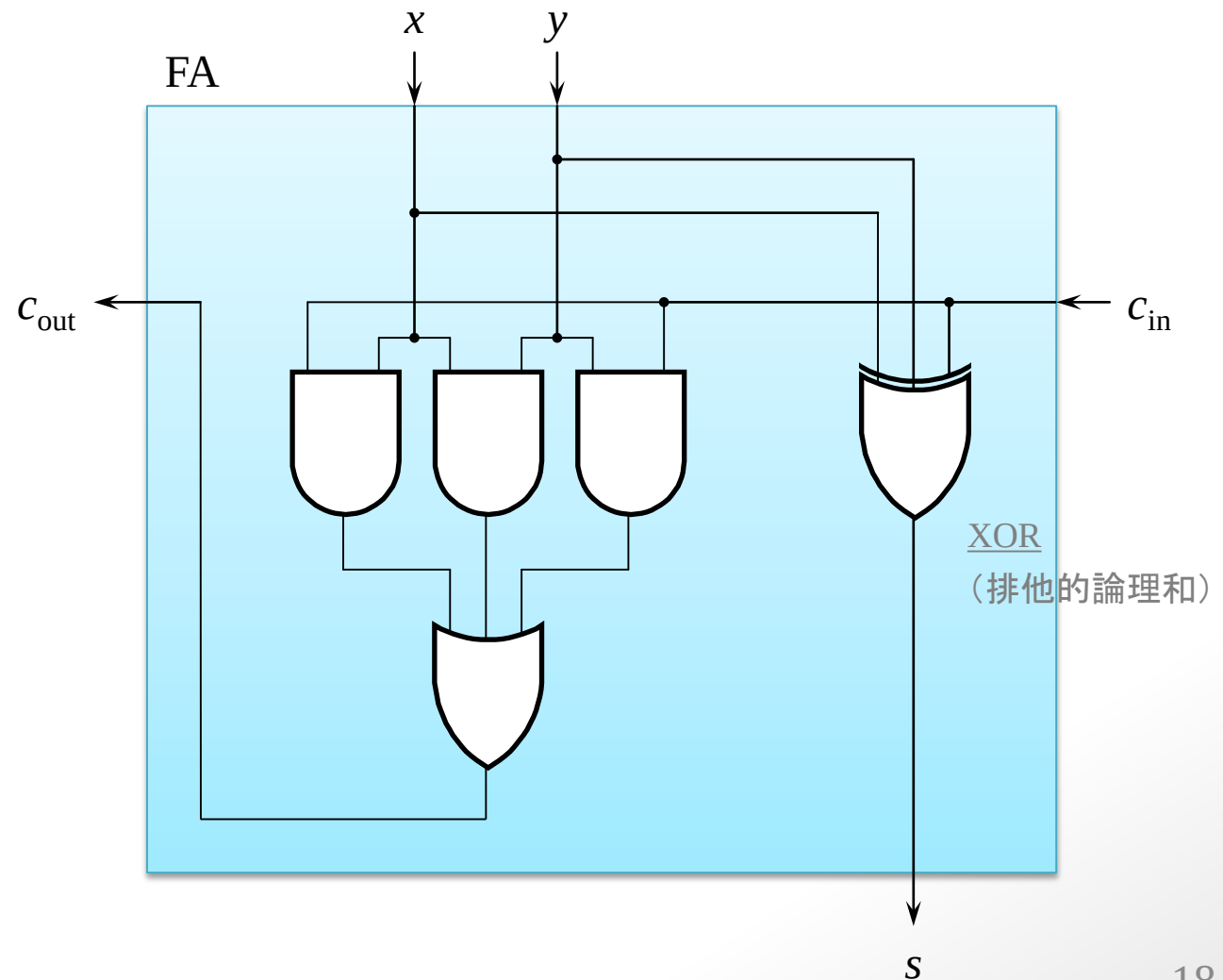


加算器

全加算器

1 の数	C_{in}	x	y	C_{out}	S
0	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
2	0	1	1	1	0
1	1	0	0	0	1
2	1	0	1	1	0
2	1	1	0	1	0
3	1	1	1	1	1

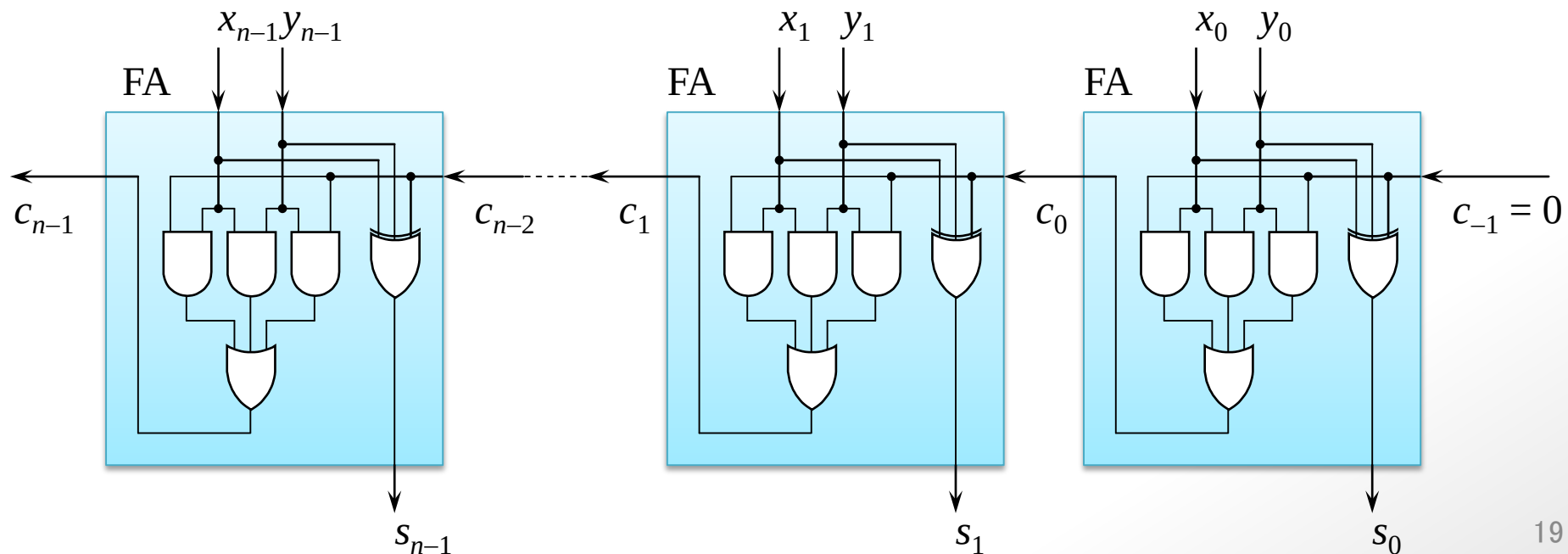
1 の数	C_{in}	x	y	C_{out}	S
0	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	1
2	0	1	1	1	0
2	1	0	1	1	0
2	1	1	0	1	0
3	1	1	1	1	1



桁上げ伝搬加算器 (ripple-carry adder)

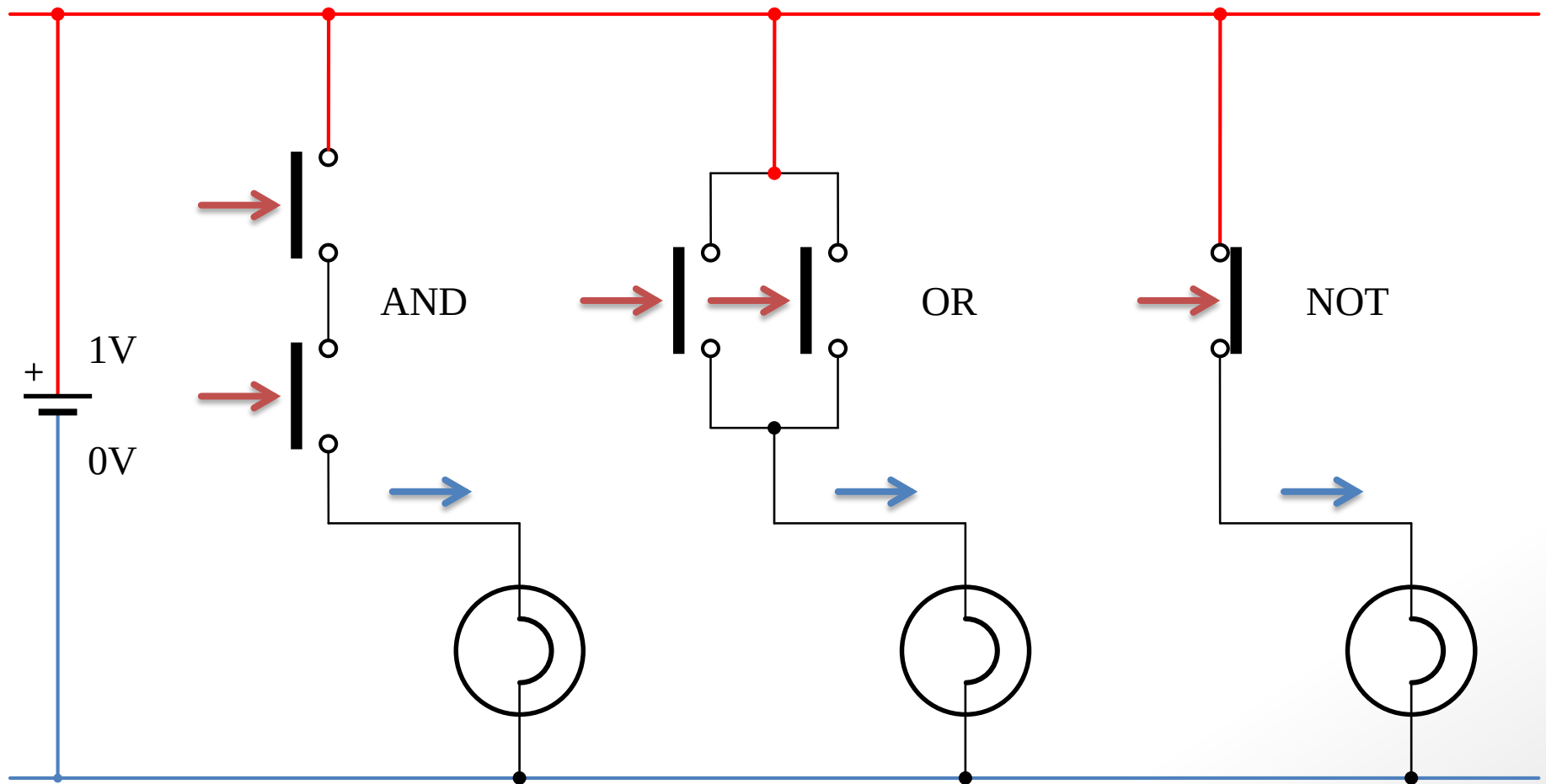
- ◆ n 個の全加算器を数珠つなぎ
- 筆算と同じく、桁上げ (carry) が下位から伝播
- ◆ (実際には、もっと効率のよい方法がある)

$$\begin{array}{r}
 c_{n-1} c_{n-2} \cdots c_1 c_0 \\
 x_{n-1} \cdots x_1 x_0 \\
 +) y_{n-1} \cdots y_1 y_0 \\
 \hline
 s_{n-1} \cdots s_1 s_0
 \end{array}$$



論理ゲートの 作り方

機械式 電気スイッチ



機械式 電気スイッチ の 多段接続性

■ 入出力

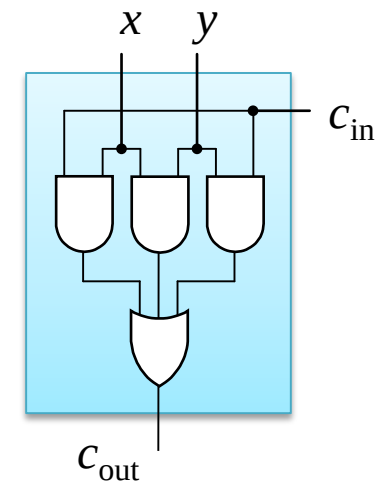
◆ 入力: スイッチを押す力 (物理)

◆ 出力: 電球の光

◆ 多段接続: 不能

■ 「光では次のスイッチを押せない」

■ 入力と出力が「同じ」である必要がある



リレー

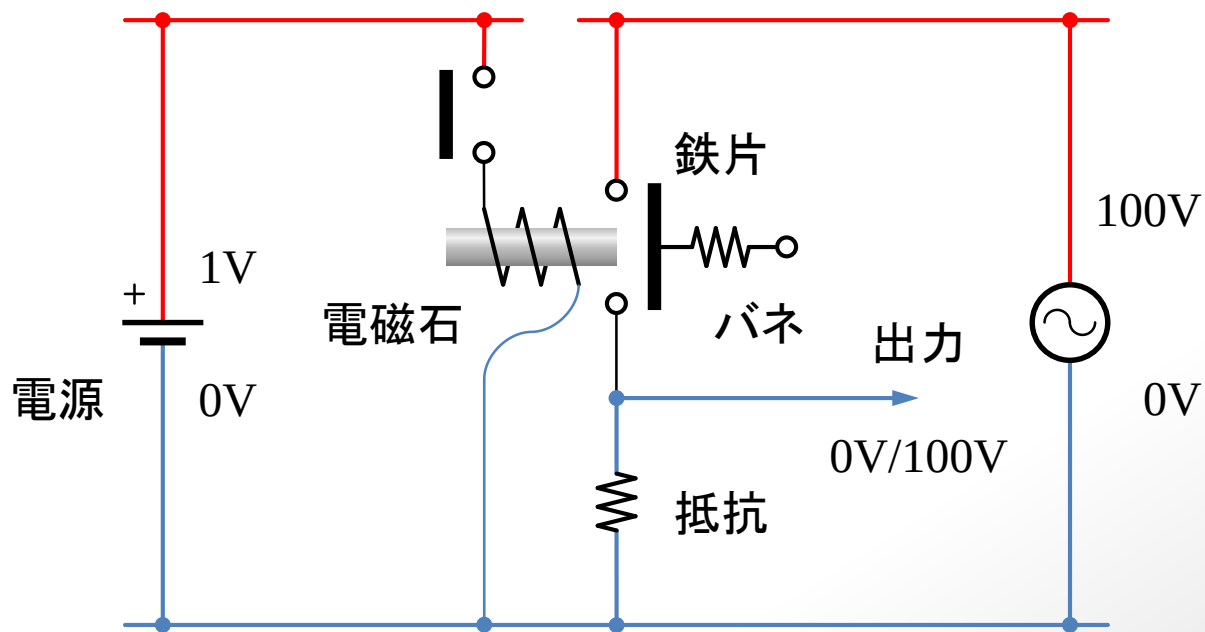


オムロン, パワー汎用リレー【LY2-AC100/110V】

<http://www.marutsu.co.jp/pc/i/3186/>

■ リレー(継電器): 電磁式 電気 スイッチ

- ◆ テレビの電源をリモコンで入れると「パチッ」
- ◆ 普通は, 低電圧 → 高電圧



リレー

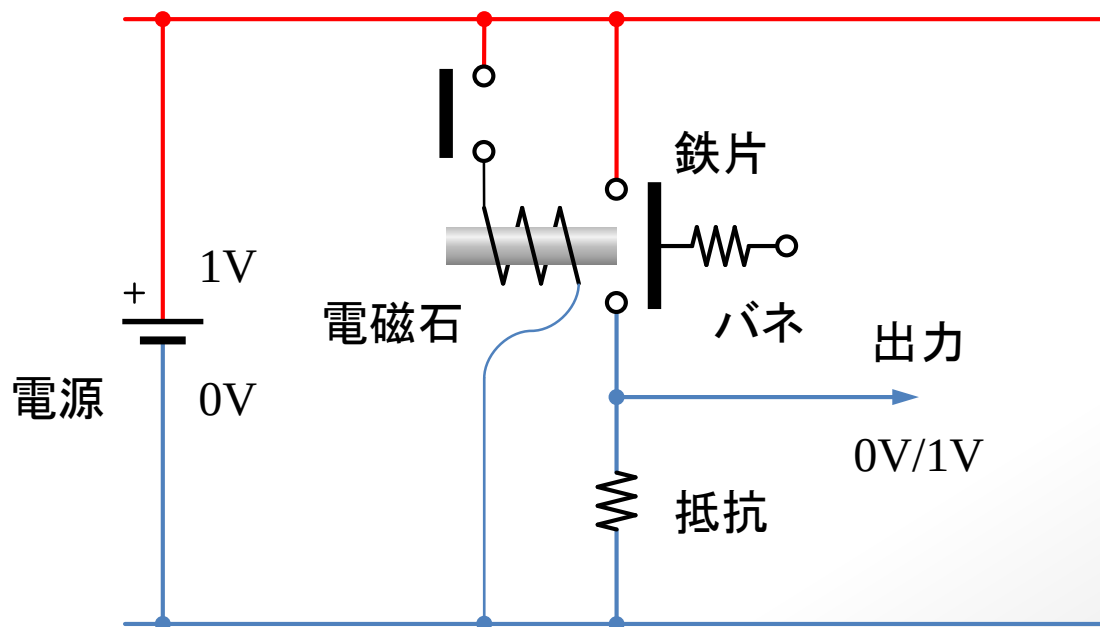


オムロン, パワー汎用リレー【LY2-AC100/110V】

<http://www.marutsu.co.jp/pc/i/3186/>

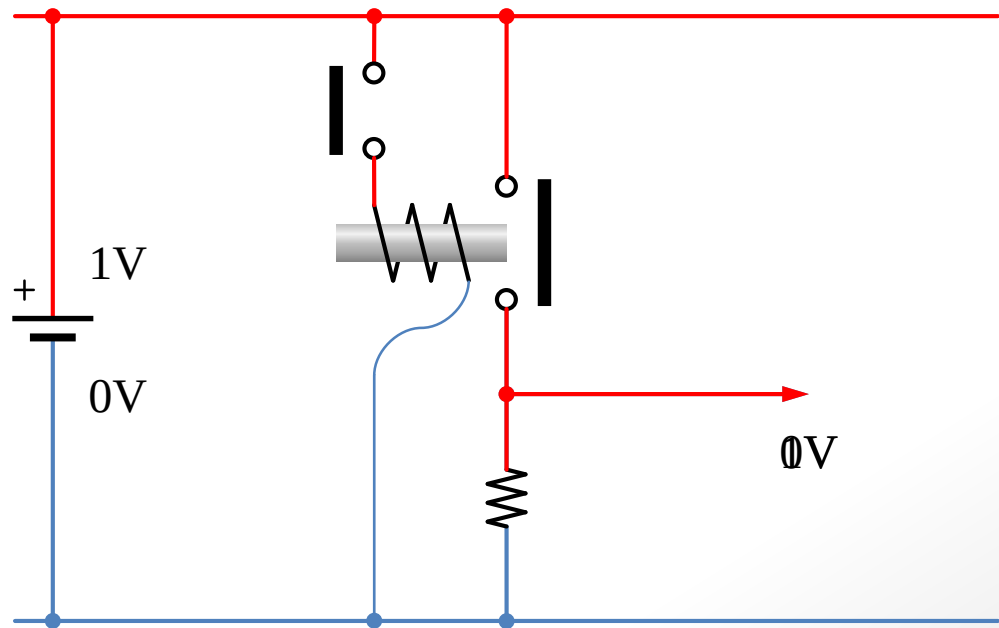
■ リレー(継電器): 電磁式 電気 スイッチ

- ◆ テレビの電源をリモコンで入れると「パチッ」
- ◆ コンピュータとして使うなら, 単一電圧で



リレーの動作

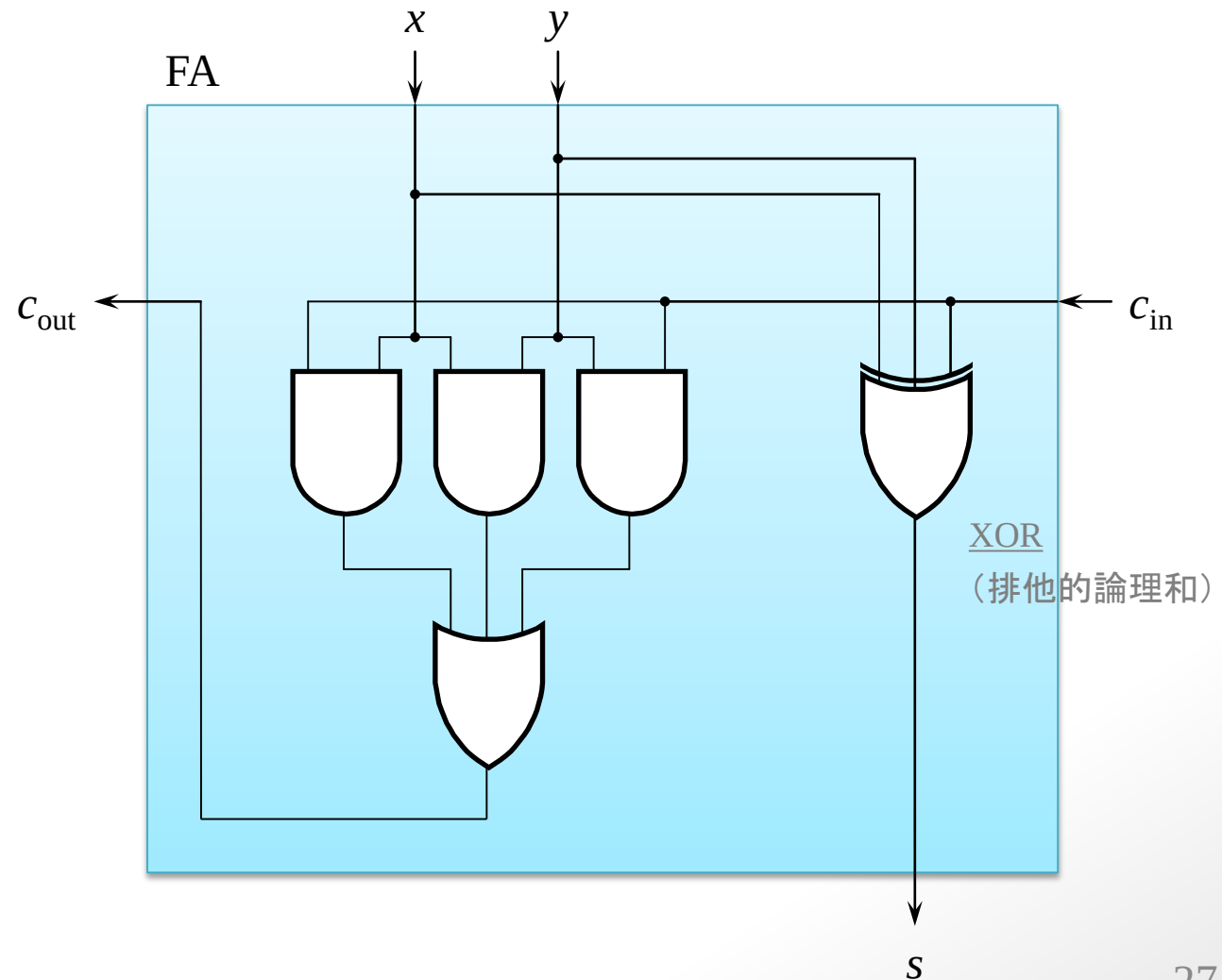
- スイッチ: OFF $\Rightarrow$ 出力: 0V
スイッチ: ON $\Rightarrow$ 出力: 1V
- 二値: 0.5V とかは出てこない
- ◆ 「0/1 で動く」



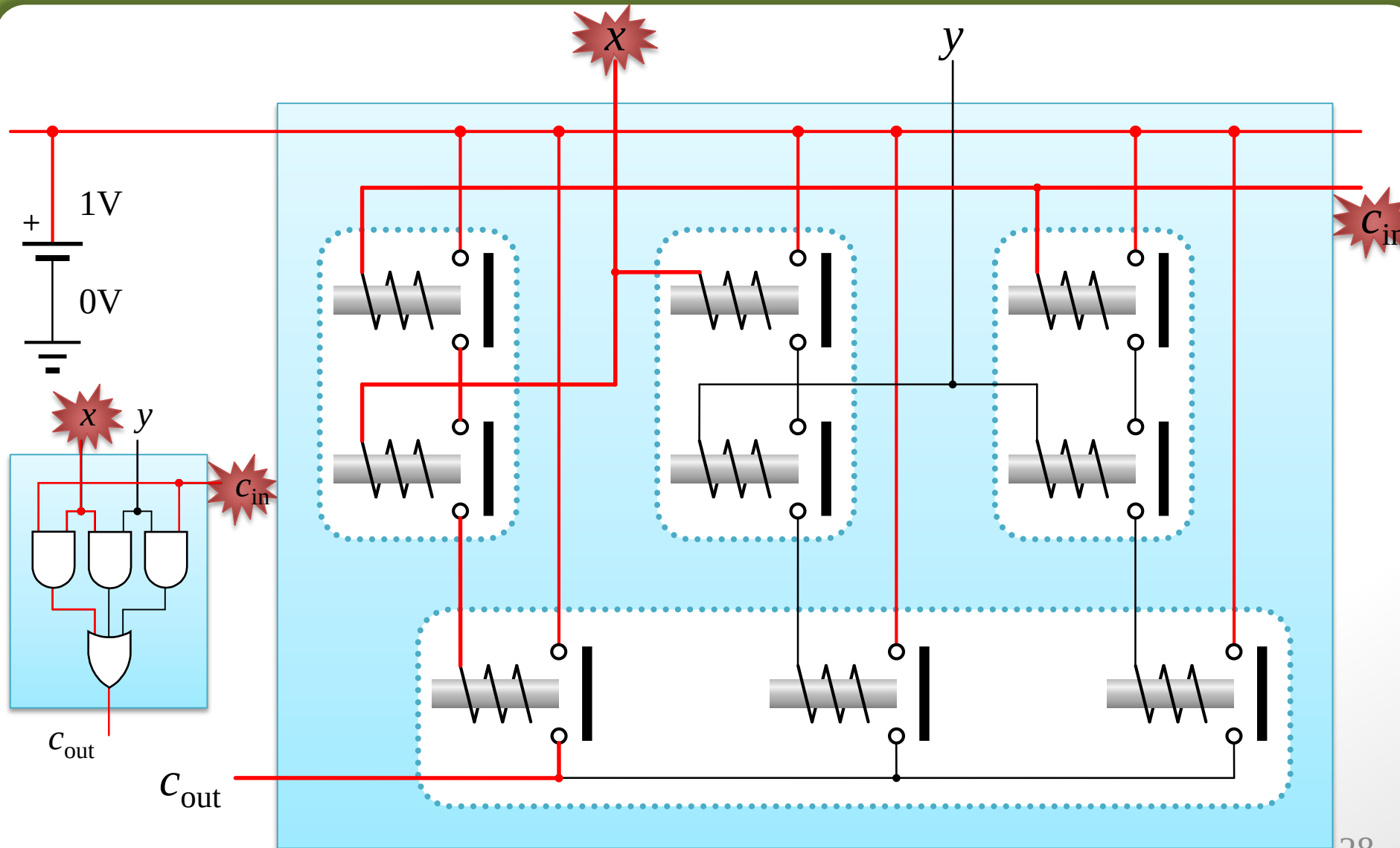
全加算器(再掲)

1 の数	C_{in}	x	y	C_{out}	s
0	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
2	0	1	1	1	0
1	1	0	0	0	1
2	1	0	1	1	0
2	1	1	0	1	0
3	1	1	1	1	1

1 の数	C_{in}	x	y	C_{out}	s
0	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	1
2	0	1	1	1	0
2	1	0	1	1	0
2	1	1	0	1	0
3	1	1	1	1	1



リレー式 桁上げ 計算器



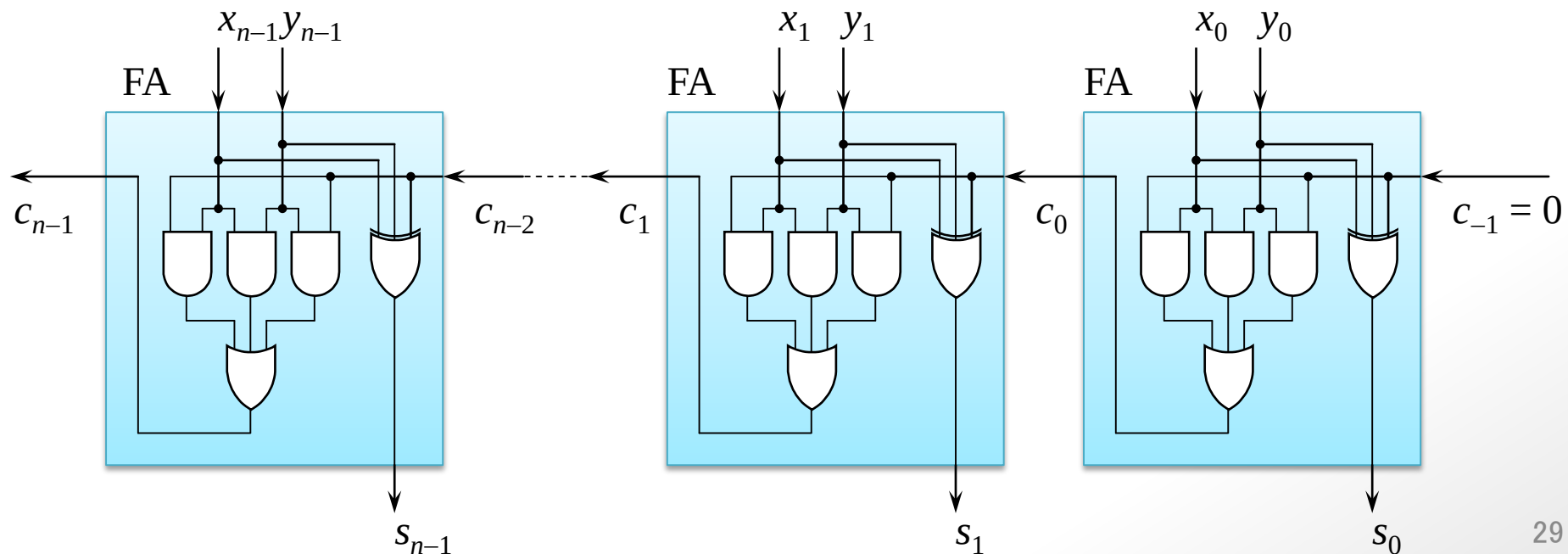
桁上げ伝搬加算器 (ripple-carry adder) (再掲)

◆ n 個の全加算器を数珠つなぎ

■ 筆算と同じく、桁上げ (carry) が下位から伝播

◆ (実際には、もっと効率のよい方法がある)

$$\begin{array}{r}
 c_{n-1} c_{n-2} \cdots c_1 c_0 \\
 x_{n-1} \cdots x_1 x_0 \\
 +) y_{n-1} \cdots y_1 y_0 \\
 \hline
 s_{n-1} \cdots s_1 s_0
 \end{array}$$



リレー式 論理ゲートの 多段接続性

■ 入出力

◆ 入力: 電圧 (0V/1V)

◆ 出力: 電圧 (0V/1V)

◆ 「同じ」⇒ 多段接続 可能

■ スイッチ ON/OFF ⇒ 次のスイッチ ON/OFF ⇒ ...

■ ドミノ倒し？

■ 速度

◆ 遅い(コンピュータに用いるには)

■ 目に見えるくらい 大きい 鉄片 が,

■ 目に見えるくらい 長い距離を 動いて,
スイッチングするから

余談：リレー式計算機

■ 富士通 FACOM128B (1959年製)

◆ CPU に 約 5000個 の リレー

◆ 第一回 情報処理技術遺産 認定



富士通テクノロジーホールに展示されている FACOM138A

■ 富士通 リレー式計算機 技術継承プロジェクト

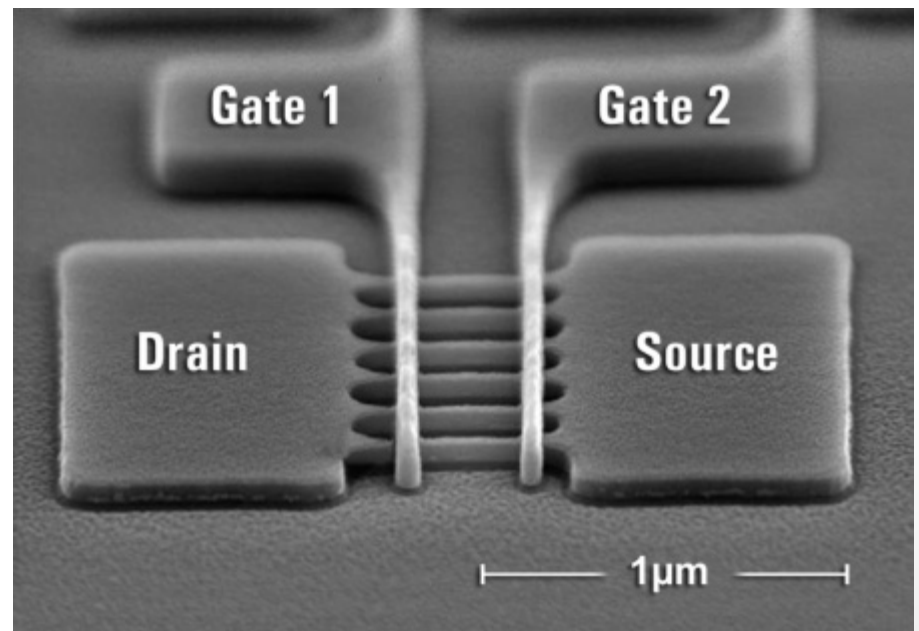
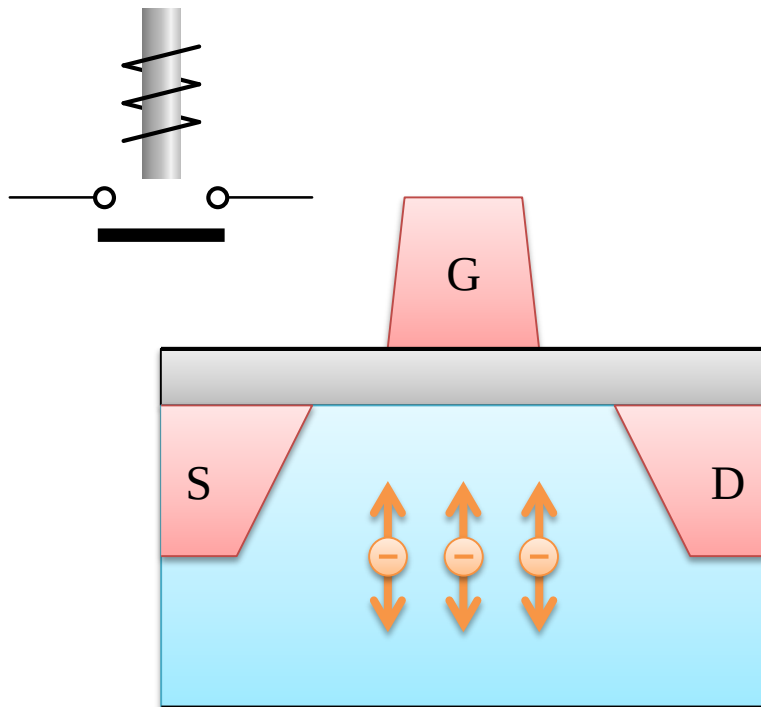
<http://www.fujitsu.com/jp/about/plus/museum/relay/>

トランジスタ と 半導体

トランジスタ

■ FET (Field-Effect Transistor, 電界効果トランジスタ)

- ◆ 電界で電子を動かしてスイッチング
- ◆ 「電子式 電気スイッチ」



半導体

	抵抗率 (Ωm)	On/Off 制御
導体	$10^{-8} \sim 10^{-7}$	Off にならない
半導体	10^3	On (10^{-6}) $\Leftrightarrow$ Off (10^6)
不導体 (絶縁体)	$10^6 \sim 10^{17}$	On にならない

■ 半導体の例:

- ◆ シリコン, ゲルマニウム (Ge), ダイヤモンド?
- ◆ ガリウムヒ素 (GaAs), 窒化ガリウム (GaN), InP
- ◆ SiC, SiGe
- ◆ ZnO, IGZO (In, G, Zn, O)

57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

余談：シリコン

■ シリコン (silicon, ケイ素, 珪素), $_{14}\text{Si}$

- ◆ 地球の主要な構成元素で, 地殻中には酸素に次いで多い
 - 鉱物(石)の主要な構成元素

■ シリカ, SiO_2 , 二酸化ケイ素

- ◆ 石英, クォーツ(結晶)
 - グラスファイバー
- ◆ シリカゲル, 乾燥剤

■ シリコーン (silicone)

- ◆ シリコーン(ゴム), etc.
- ◆ シリコンを含む化合物で, シリコン(単体)とは異なる

半導体トランジスタの多段接続性

■ 入出力

- ◆ 入力: 電圧 (low/high)

- ◆ 出力: 電圧 (low/high)

- ◆ 「同じ」⇒ 多段接続 可能

 - ドミノ倒しのよう

■ 速度:

- ◆ 速い

 - 原子は動かず,

 - 電子が動いて

スイッチングするから

半導体トランジスタのメリット

■ スイッチングが高速

◆ 原子は動かず，電子が動いてスイッチングするから

■ 電子：容易に制御可能な最小の物質

■ 微細化，高集積化が可能

◆ フォト・リソグラフィー：

■ 写真の技術

◆ ムーアの法則：

■ 「半導体の集積密度は18～24ヶ月で倍増する」

◆ 現在の最小加工寸法：14nm

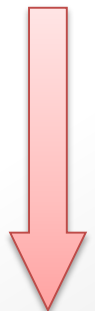
■ (現在のところ)最適な論理ゲート

コンピュータのしくみ？

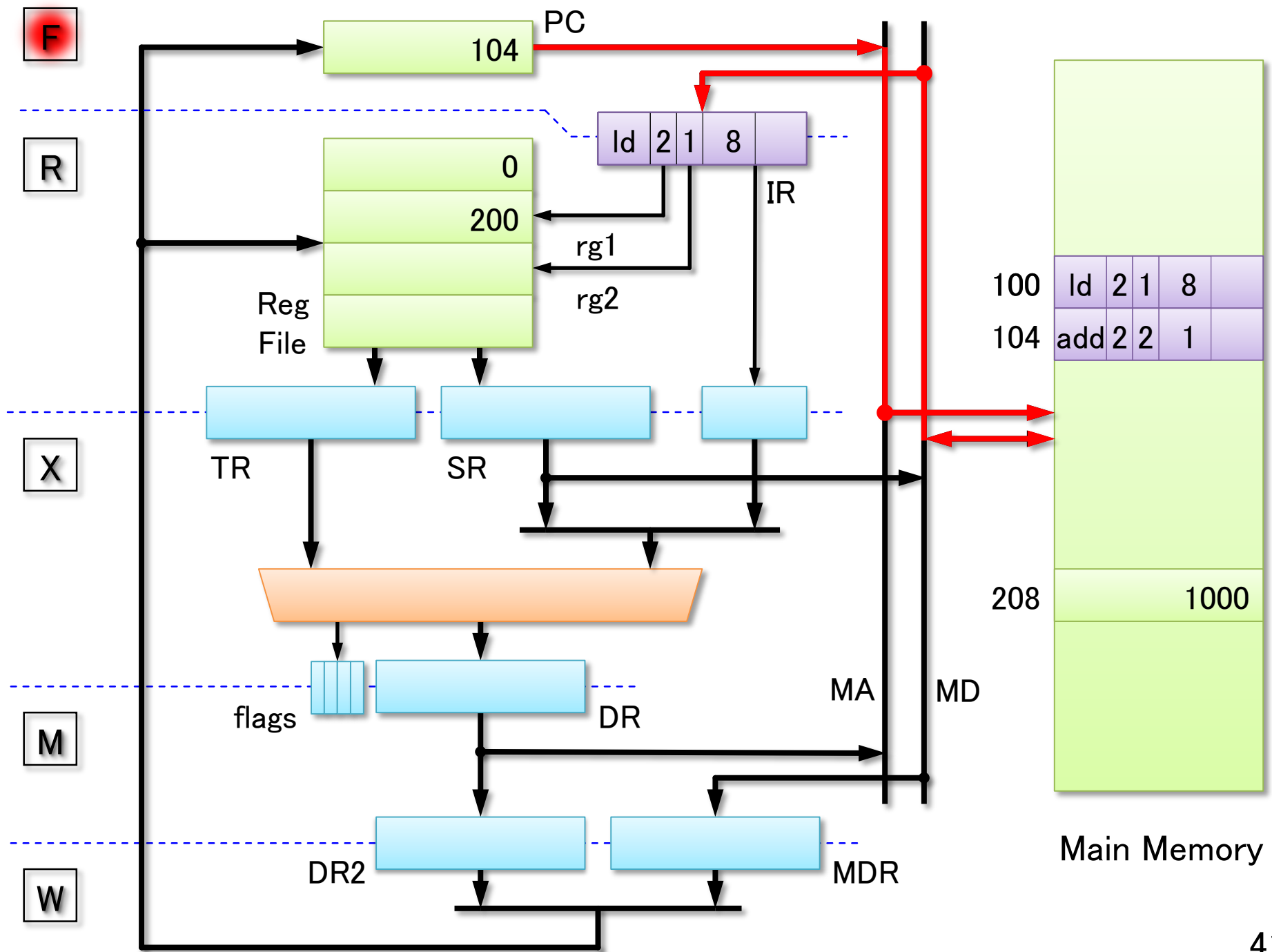
大学のカリキュラム と 本日の内容

学年	講義名	本日の内容
大学院	コンピュータ アーキテクチャ 特論	スーパーコンピュータ
学部 3年	学生実験	(コンピュータの基本的なしくみ), 命令セット・アーキテクチャ
	コンピュータ アーキテクチャ	
学部 2年	ディジタル回路 (論理回路)	二進数, 加算器
	電子回路	トランジスタ, 半導体

後半



前半



コンピュータは、たとえば言うなら...

■ レストラン ⇔ 食堂 ⇔ 食品工場, ただし...

◆ 料理人: 他のジャンルのプロだが, このジャンルはまったく経験なし

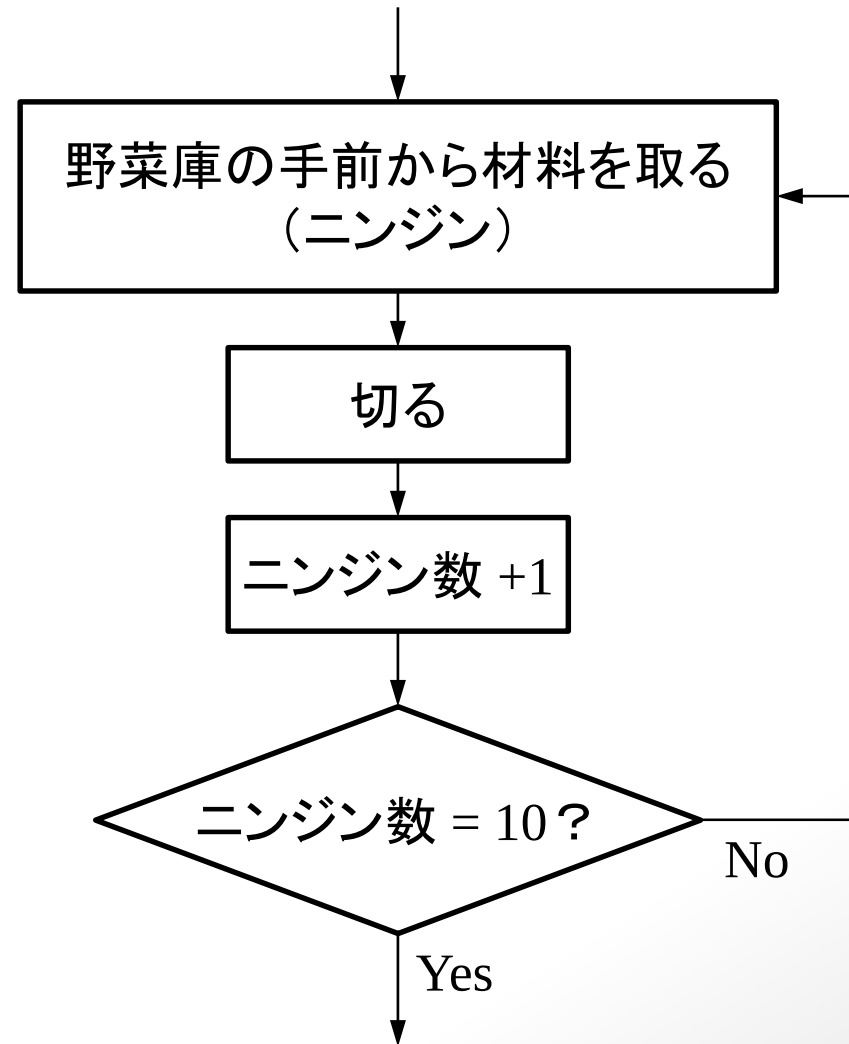
■ レシピを見ながらでないと, 料理ができない!

■ レシピ: プログラム

レシピ：プログラム

■ レシピ：プログラム

- ◆ 料理人の分かる「言葉」で
- ◆ こと細かに(フローチャート)
 - 「ニンジン」を10本切る」は NG



コード例

```
/* absolute diff, |a - b| */  
int absdiff (int a, int b)  
{  
    if (a > b) {  
    THEN:  a -= b;  
    } else {  
    ELSE:  b -= a;  
          a = b;  
    }  
    EXIT:  return a;  
}
```



翻訳

```
absdiff:  LD      4, bp, ax    # ax = *(bp + 4)  
          LD      8, bp, bx    # bx = *(bp + 8)  
          CMP     bx, ax      # ax - bx  
          BLT     ELSE        # LT : less than  
    THEN:  SUB     bx, ax      # ax -= bx  
          B       EXIT  
    ELSE:  SUB     ax, bx      # bx -= ax  
          MOV     bx, ax      # ax = bx  
    EXIT:  ST      ax, 0, bp   # *(bp + 0) = ax  
          RET
```

高級言語プログラム (C 言語)

アセンブリ言語プログラム

命令セット・アーキテクチャ

■ レシピ(プログラム)は, 料理人の分かる「言葉」で

◆ 命令セット・アーキテクチャ: 「言葉」の文法

■ x86 (IA-32, IA-64)

- Windows パソコン, Mac, サーバ (Intel 社, AMD 社)

■ ARM

- スマホ, タブレット(各社)

■ POWER

- サーバ (IBM 社)

■ V850

- 車載等 (ルネサス社)

■ etc., etc.

命令セット・アーキテクチャ

■ 他の「言葉」で書かれたレシピ

◆ そんなに違うわけではない

- 人間なら何となく分かる程度の違いだが、
- コンピュータにはさっぱり分からない！

◆ 「(バイナリ)互換性がない」

■ 逆に、「言葉」が同じなら、会社が違って大丈夫

◆ 「(バイナリ)互換性がある」

- Intel 社の x86 プロセッサ と AMD 社の x86 プロセッサ
- 各社の Android スマホ(ほぼ ARM)

余談: x86

■ 世界最初のマイクロプロセッサ

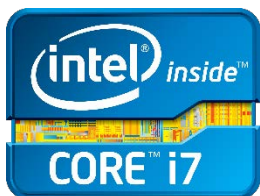
◆ 40年に渡る拡張の歴史

■ 特長:

◆ 最も普及している? (現在)

◆ 最も古い(マイクロプロセッサとして)

◆ 最も複雑(田舎の民宿)



年	名前	bit
1971	4004	4bit
1972	8008	8bit
	8086	16bit
	80286	
	80386	32bit
	80486	
	Pentium	
2004	Pentium 4	64bit
2006	Core	
	⋮	
2016	Core i (Gen. 6)	

余談: ARM



■ ARM 社

- ◆ イギリス, ケンブリッジ
- ◆ ファブレス(工場を持たない, 最終製品を販売しない)
 - ARM アーキテクチャの使用許諾 ⇒ ライセンス収入
 - ARM アーキテクチャに基づくプロセッサの設計の販売
 - 顧客は, 設計を買ってきて, 自社で焼くか, ファウンドリ(工場)で焼いてもらう
- ◆ スマホ(iPhone, Android, Windows Phone)用プロセッサで絶大なシェア(たぶん 100%)
- ◆ 先日, 日本のソフトバンク社に 3兆円で買収された

高速化

高速化

■ レストラン ⇔ 食堂 ⇔ 食品工場 を高速化するには...

◆ 「低レイテンシ」: 注文された料理を早く出す

◆ 「高スループット」: 一定時間内に料理をたくさん出す

■ 2つの方向:

◆ レストラン ⇒ 低レイテンシ

■ 「要領のよい」料理人を雇う

- ただし、「要領のよい」料理人は人件費が高い

◆ 食品工場 ⇒ 高スループット

■ 料理人を大勢雇う

- ただし、個々の料理が早く出て来る訳ではない

「要領がよい」とは？

- スーパースカラ (superscalar)
 - ◆ 八面十六臂：同時に異なる種類の8個の処理を行う
 - ニンジンを切りながら、ソテーしながら、...
 - 低レイテンシ：一つの料理を速く作れる



国宝 阿修羅像 734年

＜撮影:金井杜道 © 興福寺＞ <http://www.kohfukuji.com/property/photo/>

「待ち」と投機

■ 八面十六臂でも、8倍高速化は難しい

◆ いろいろな「待ち」が生じてしまう

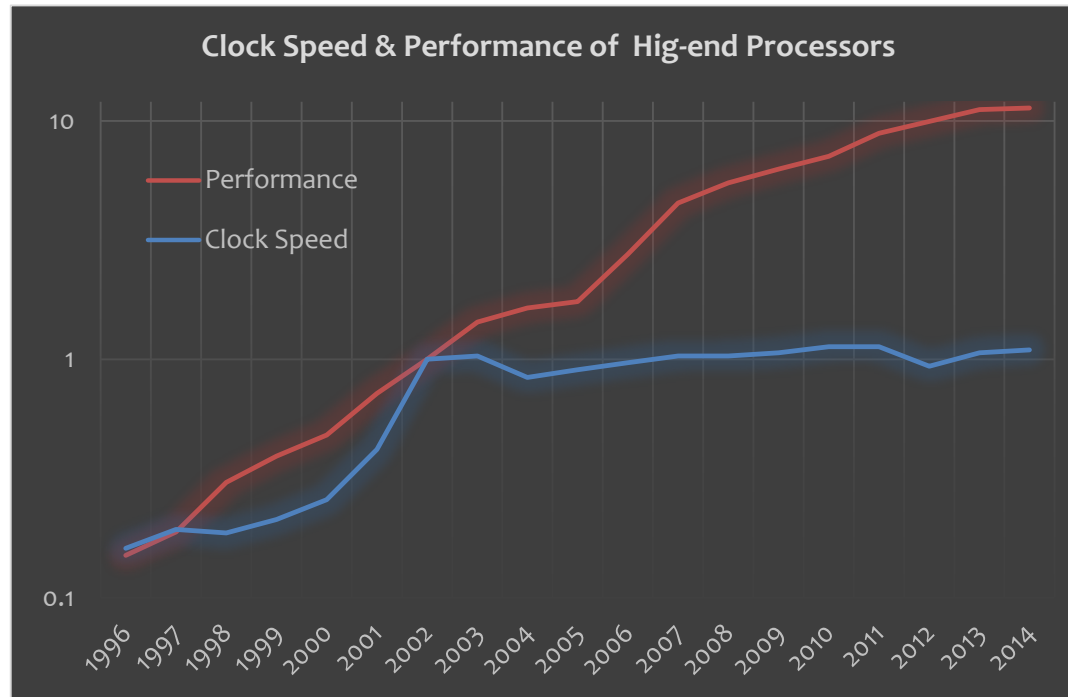
- ソテーしようと思ったら、材料が切れてない
- お湯が沸いてない
- 材料が冷蔵庫にない → 近所のコンビニにもない → 遠くのスーパーに

■ 投機 (speculation)

◆ レシピを先読みして、予めやっておく ⇒ 3分クッキング

- お湯は沸かしておく
- スーパーに行ったら、ついでに買っておく
- 「予め用意しておいたものがこちらです。」

プロセッサの性能



■ 2002年から、効率(要領のよさ)が10倍！

◆ 2002年で、クロックの速度向上は停止

◆ それでも性能は、10倍！

スパコン

高速化(再掲)

■ レストラン ⇔ 食堂 ⇔ 食品工場 を高速化するには...

◆ 「低レイテンシ」: 注文された料理を早く出す

◆ 「高スループット」: 一定時間内に料理をたくさん出す

■ 2つの方向:

◆ レストラン ⇒ 低レイテンシ

■ 「要領のよい」料理人を雇う

- ただし、「要領のよい」料理人はコスト(給料)が高い

◆ 食品工場 ⇒ 高スループット

■ 料理人を大勢雇う

- ただし、個々の料理が早く出て来る訳ではない

食品工場 ⇒ 「高スループット」その1

■ スーパーコンピューティング, ハイパフォーマンス コンピューティング

◆ 科学技術計算, 大規模シミュレーション, ビッグデータ解析, 人工知能?, etc.

◆ 整数 と 浮動小数点数 なら, 後者が多い

■ FLOPS (Floating-point Operations Per Second)

新聞・テレビで, 「1秒間に〇〇億回計算する能力がある」

◆ データ並列処理

■ 異なるデータに対して, 同じ処理を「同時に」大量に

浮動小数点数

■ 浮動小数点数 (floating-point number)

◆ 「すごく大きい/小さい小数を効率よく表す方法」

◆ 科学技術計算の大部分は, これの積和

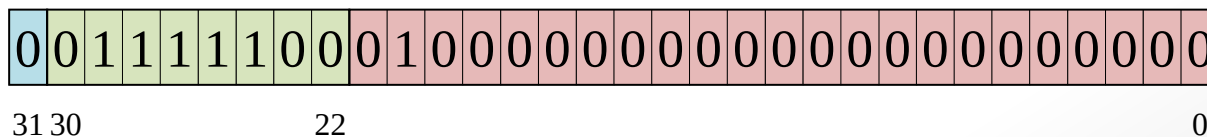
十進表現

$$1234000000000 \Rightarrow + 1.234 \times 10^{12}$$

$$- 0.0000000005678 \Rightarrow - 5.678 \times 10^{-9}$$

IEEE 754 単精度 浮動小数点フォーマット (32bit)

$$1.01 \times 2^{1111100} - \text{十五}$$



ポスト「京」の重点課題

ポスト「京」の重点課題(9課題)

ポスト「京」では「京」が切り拓いたさまざまな領域をさらに拡大するとともに、未踏の分野にも挑戦します。



食品工場 ⇒ 「高スループット」その2

■ グラフィクス

◆ 多数のポリゴンの描画

- ポリゴン → ピクセル などの座標変換
- 浮動小数点演算

■ GPU と GPGPU

◆ GPU (Graphics Processing Unit) ⇔ CPU (Central Processing Unit)

- 最近では, 「CPU」は GPU と対比する場合以外, あまり言わない

◆ General-Purpose GPU, 汎用 GPU

- GPU をベースに, より一般的なデータ並列処理ができるように

n 面 m 臂 (1/2)

■ (ワイド)スーパスカラ (Superscalar)

◆ 八面十六臂: 同時に異なる種類の 8 個の処理を行う

- ニンジン: 切りながら, ソテーしながら, ...
- 低レイテンシ: 一つの料理を速く作れる

■ メニーコア (many-core)

◆ 一面二臂 (普通の人) $\times$ 8 人:

- レシピ: プログラムを, 8 人用に書き直させれば... (並列化)
- 8 人の連携が難しく, 八面十六臂のようにはいかない

n 面 m 臂 (2/2)

■ SIMD (Single-Instruction/Multiple-Data stream, 「シムディー」)

◆ 一面十六臂: 同時に同じ種類の 8 個の処理を行う

■ ニンジンと同時に8本切る

■ 高スループット: 一定時間内に, 同じ料理を 8 倍作れる

■ GPGPU

◆ SIMD と似ているけど, より柔軟

n 面十六臂

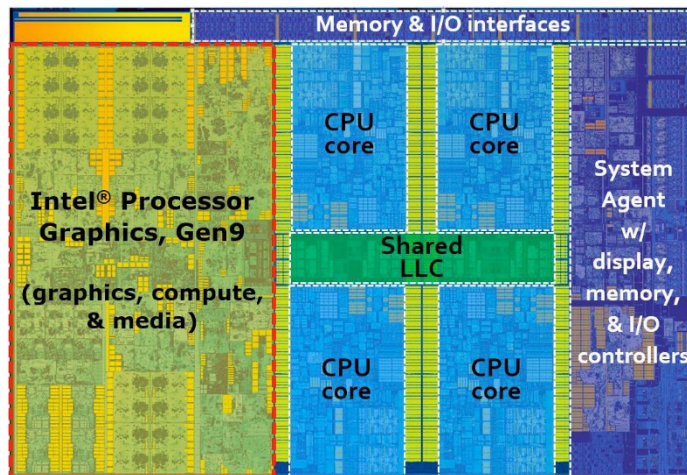
■ コスト(人件費):

◆ (一面二臂) $\times 8 >$ 八面十六臂 $\gg$ 一面十六臂

- 臂(腕)に比べて面(あたま)は高い
- 異論はある

実際は、組み合わせて

用途	コア		×	コア数	GPU
	メイン	SIMD ユニット			
高いスマホ用	(三面 六臂 + 一面 四臂)		×	4 コア	あり
安いパソコン, ノート用	(四面 八臂 + 一面 八臂)		×	2 コア	あり
高いパソコン用	(八面十六臂 + 一面十六臂)		×	4 コア	あり
高いサーバ用	(八面十六臂 + 一面十六臂)		×	20 コア	なし



Intel "Skylake" Die Layout Detailed

<https://www.techpowerup.com/215333/intel-skylake-die-layout-detailed/>

■ 浮動小数点:

- ◆ メイン だと, 二面四臂分くらい
- ◆ SIMD ユニットの 方で稼ぐ

スパコンのアプローチ

■ コアをすごくたくさん(数十万～1千万個)並べる

◆ 最近では、(資金と)電力がネック

■ 大/中/小コア×コア数:

◆ ↑「少数精鋭」

- (八面十六臂 + 一面十六臂) × 少な目
- (?面??臂 + 一面??臂) × 中くらい
- (二面 四臂 + 一面十六臂) × 多め
- GPGPU × 多め
- (一面 二臂) × 超多め

... 次世代クラスタ

... 京, ポスト「京」

... アメリカの次世代スパコン

... アメリカ, 中国

... 中国?, Pezy?

◆ ↓「人海戦術」

■ 比較指標:

◆ 電力効率: 最大性能 (FLOPS) ÷ 消費電力

... 「人海戦術」の方がよい

◆ プログラマビリティ, 「プログラムが簡単に書けるか?」

... 「少数精鋭」の方がよい

Top 500 (2016年 7月) と Graph 500 (2016年6月)

Rank	国	System	Core	Cores	$R_{\max}$ (PFlop/s)	R_{peak} (PFlop/s)	Power (MW)
1	中国	神威 太湖之光	神威 26010	10,649,600	93.0	125.4	15.4
2	中国	天河 2	Intel Xeon Phi	3,120,000	33.9	54.9	17.8
3	米	タイタン	NVIDIA K20x	560,640	17.6	27.1	8.2
4	米	セコイア	IBM BlueGene/Q	1,572,864	17.2	20.1	7.9
5	日本	京	富士通 SPARC64	705,024	10.5	11.3	12.7

Rank	国	System	Core	GTEPS
1	日本	京	富士通 SPARC64	38,621
2	中国	神威 太湖之光	神威 26010	23,756
3	米	セコイア	IBM BlueGene/Q	16,599
4	米	ミラ	IBM BlueGene/Q	14,328
5	独	JUQUEEN	IBM BlueGene/Q	5,848

今日のまとめ

つまみ食いのポイント

■ 前半：

◆ 0 と 1 で動くってどういうこと？

- スイッチだから, on/off

◆ なぜ半導体で作るの？

- 速いスイッチが作れるから

- 導体だと, off にならないので, 物理的に切断する必要がある
- 半導体だと, 電子が動いて on/off

つまみ食いのポイント

■ 後半:

◆ コンピュータって、たとえば？

■ レシピを見ながら料理する料理人

◆ x86 とか ARM アーキテクチャって何？

■ 料理人の「言葉」

◆ スパコン, ポスト「京」のポイントは？

■ 電力効率(人海戦術)とプログラマビリティ(少数精鋭)の バランス

おわりに (1/4)

- 「ブラックボックス」であること自体は、悪いことではない
 - ◆ 自動車のしくみは分かってなくても、運転できる
 - ◆ 飛行機のしくみは分かってなくても、海外旅行はできる
 - ◆ コンピュータのしくみは分かってなくても、ポケモン GO はできる
 - しくみが分かったところで、〇〇デポにはだまされる？

おわりに (2/4)

■ 抽象化, 階層化

◆ 各階層の研究・開発に注力できる

- 特に, 情報の分野は, 階層化を大事にしてきた

◆ プレイヤーが参入しやすい ⇒ エコシステム, コミュニティ

- Linux, Android

■ むしろ, 日本の(電機)メーカーは, 自前主義がヒドすぎる

◆ 階層をまたぐ最適化が好き?

おわりに (3/4)

■ ただし、コンピュータに限っては？

- ◆ 「コンピュータをどう使うのかが重要であって、コンピュータ自体は買ってくればいい」
- ◆ 「選択と集中」, 「アジアの成長を取り込む」, 「金融立国・観光立国」
- ◆ 「2位じゃダメなんですか？」
- ◆ 「最もきちんと説明されてこなかったのがスパコンだ」

■ 結果...

- ◆ 日本の DRAM(メモリ)メーカはなくなった
- ◆ 最先端の半導体工場はなくなった
 - ポスト「京」のプロセッサの製造は、台湾

おわりに (4/4)

■ 一旦 技術を失うと、復活はたいへん

◆ MRJ, Honda Jet, 「心神」(ATD-X)

■ 言いたいこと:

◆ コンピュータをどう使うのかも重要だけど、
コンピュータをどう作るのかも重要だと思ってくださいよw

◆ コンピュータ(プロセッサ)も、国産にこだわってくださいよw

◆ コンピュータだけ国産じゃなくていいと思うのは、
分かってないからじゃないんですか？w

参考文献

■ ドンピシャの本は知らない

■ 大学向け教科書:

◆ 前半:

■ 五島 正裕: デジタル回路, 数理工学社 (2007).

◆ 後半:

■ Randal E. Bryant, David R. O'Hallaron: Computer Systems: A Programmer's Perspective (2016)

コンピュータ・システム —— プログラマの視点から, 丸善出版 (翻訳中)