cs-13. プロセッサ，メモリ，文字コード，論理演算，２の補数

（コンピューターサイエンス）

URL: https://www.kkaneko.jp/cc/cs/index.html

金子邦彦

謝辞：この資料では「いらすとや」のイラストを使用しています

① **コンピュータの仕組みの基礎**について、ビジュアルな説明で理解

② コンピュータシステムの構造と原理：**OS、プロセッサ、メモリ、文字コード、論理演算、2の補数**

③ 知識とスキル：**コンピューターの基本構造と動作原理、デジタルデータの表現方法（二進数、十六進数）、論理演算の基本と応用、メモリとアドレッシング、文字コード**

④ プログラミング、システム設計、デジタルリテラシーなど、将来につながる内容

アウトライン

1. OS
2. コンピュータの構成
3. デジタル
4. メモリとアドレス
5. 文字コード
6. 論理積と論理和
7. 論理演算と足し算
8. ２の補数

- **プロセッサ**: **コンピュータの頭脳**にあたる部分
- **メモリ**: コンピュータの中で**記憶**を行う
- **文字コード**: 「A」や「0」や「!」や「あ」などの**文字を，数字に置き換えるルール**
- **論理演算**：「ビット」に関する演算．AND, OR, NOT など．足し算や掛け算の基礎．

13-1 OS

OS（オペレーティングシステム）

OSは**コンピュータシステム**の**中心部分**

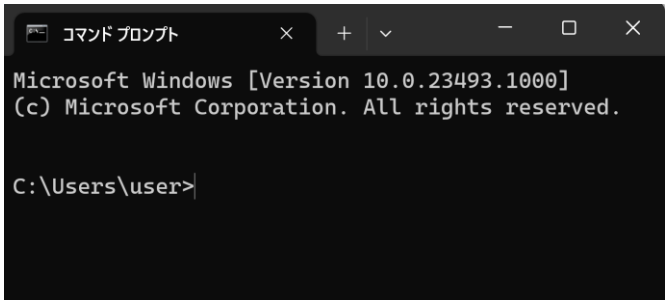
- **ハードウェア**と**他のソフトウェア**との**間**の**インターフェイス（接続部）**となる

例：アプリケーションがファイルを保存する際，OSはハードディスクの空き領域を見つけ，適切に配置

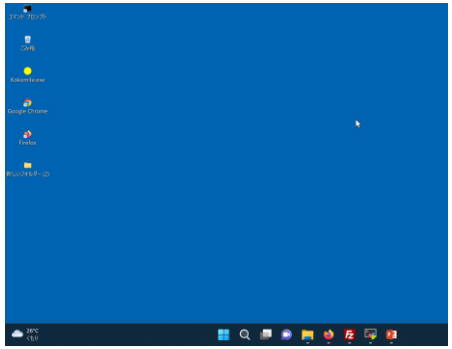
- ハードウェアの詳細を気にすることなく，**OSが提供する統一的な方法で，ハードウェアを利用**できる

OS の特徴

- **種類が豊富**：Windows、Linux（AndroidもLinuxの一種）、macOSなど。
- **基本的な操作**は**コマンドラインインターフェイス**で可能



- **多くのOS**では、グラフィカルユーザーインターフェイス（画面やメニューやアイコンなどのビジュアルなインタフェース）が主となっている。



ソフトの起動や終了，ファイルの操作，ソフトのインストールなど

OS の主な機能

- **マルチタスク**：複数のアプリケーションを**同時に実行**。それぞれにCPUを割り当てる
- **マルチスレッド**：1つのアプリケーション内で、**複数のタスク（スレッド）**を並行して実行
- **セキュリティ**: ユーザー認証（IDやパスワード）やアクセス制御（ファイル、デバイス、プログラムへのアクセス権限の管理）など、**システムとデータの保護**を行う
- **ネットワーク**: コンピュータをネットワークに接続し、**データの送受信**を行う

13-2 コンピュータの構成

プロセッサとメモリ

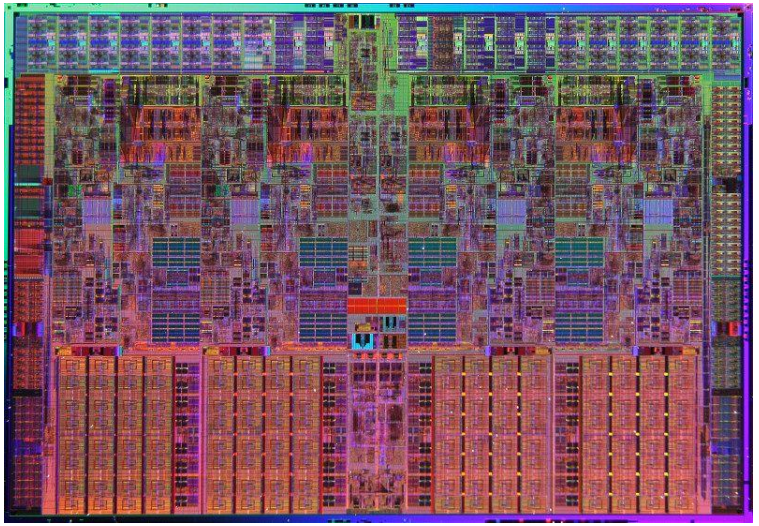
プロセッサとメモリはコンピュータの基本的な動作を支える重要な部分。

- **プロセッサ**：**計算**と**制御**を行う
- **メモリ**：**データやプログラムの一時的な保存**を行う

プロセッサ

- **プロセッサ**は**コンピュータの「脳」**で、すべての計算とデータ処理を行う.
- プロセッサは精密な電子回路で、数億個以上の**微小な電子スイッチ（トランジスタ）**で構成されている.
- これらの電子スイッチを使って、**「AND」「OR」「NOT」などの論理演算**を行う
- 単純な**論理演算を組み合わせる**ことで，プロセッサは複雑な計算や判断を高速に実行する.

インテル
Core i7 プロセッサ



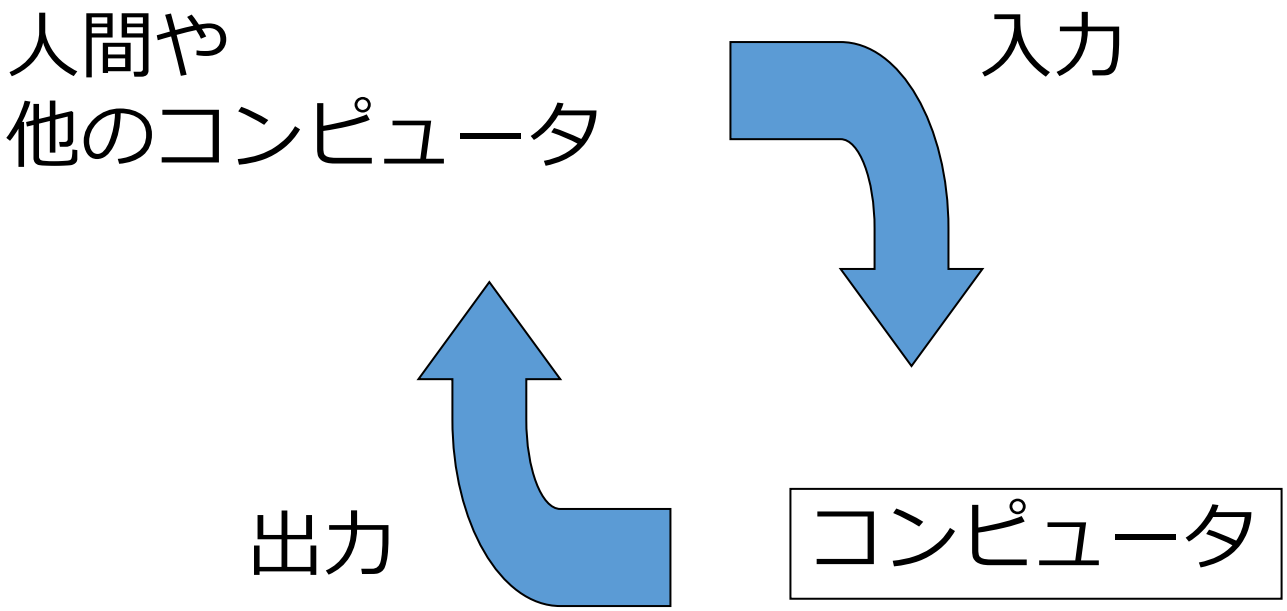
メモリ

- **メモリ**は**コンピュータの「作業台」**で、**データやプログラムを一時的に保存**します
- メモリは高速だが，電源が切れるとデータが消失する

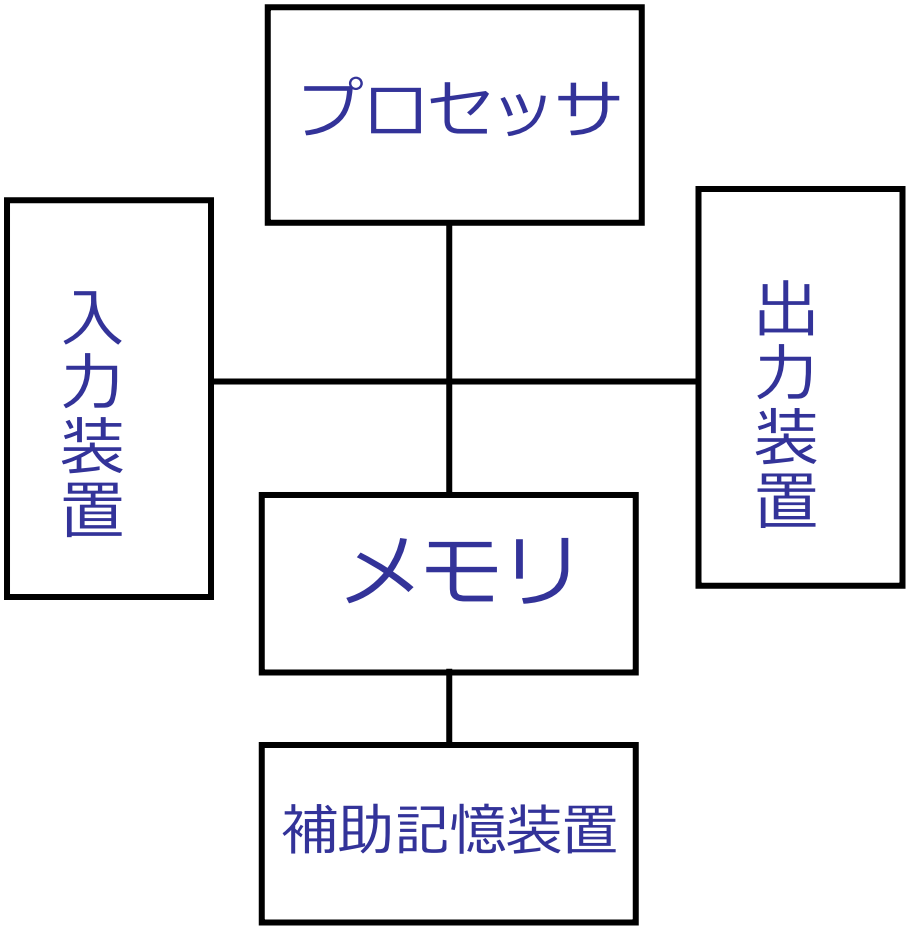
  （ハードディスクやSSDは電源が切れてもデータを保持）
- **プロセッサはメモリにアクセス**し、**データの書き込み（記憶）と読み出し、プログラムの読み出し**を行う．プロセッサは必要なデータに素早くアクセスでき、結果を素早く保存できる．

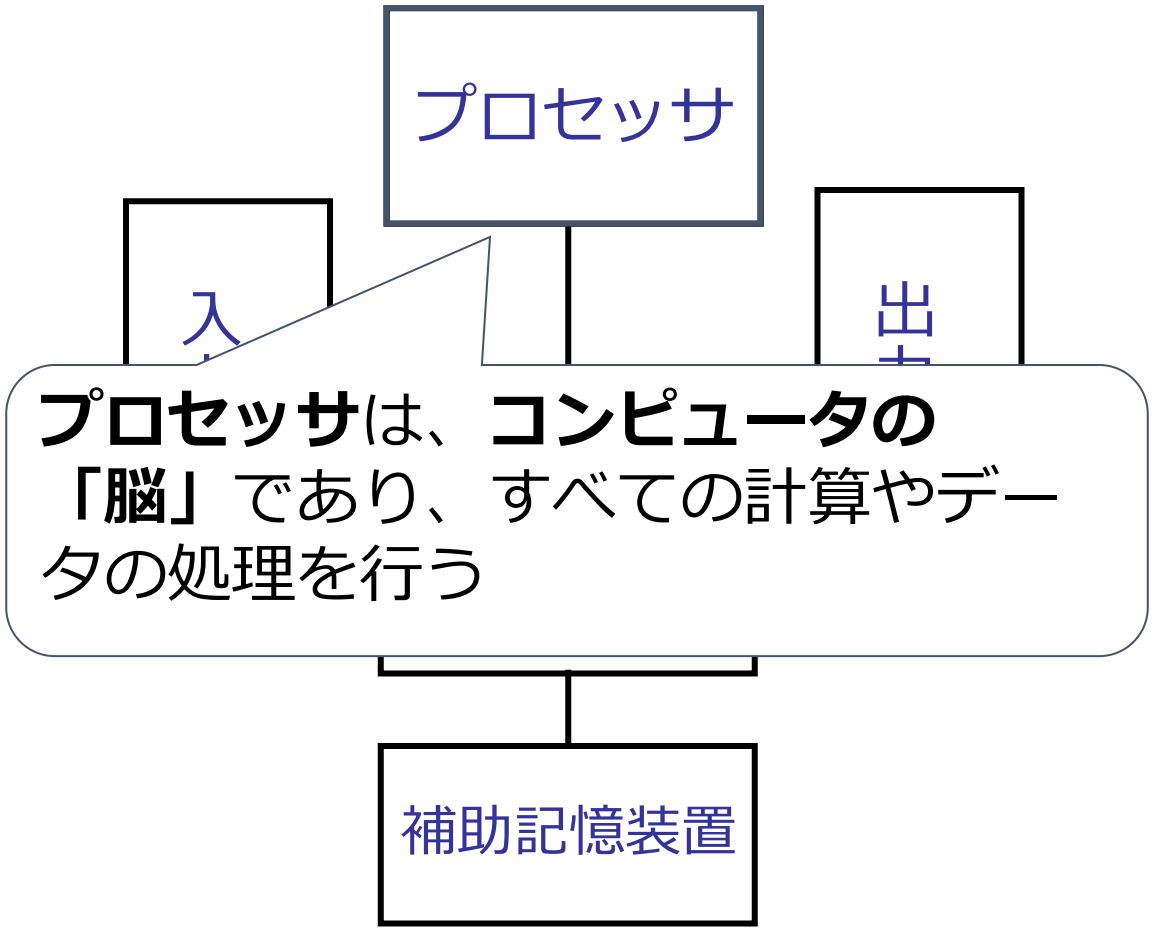
メモリもプロセッサと同様に電子回路で構成されているが，その役割は一時的なデータの保存である．

コンピュータの入力と出力

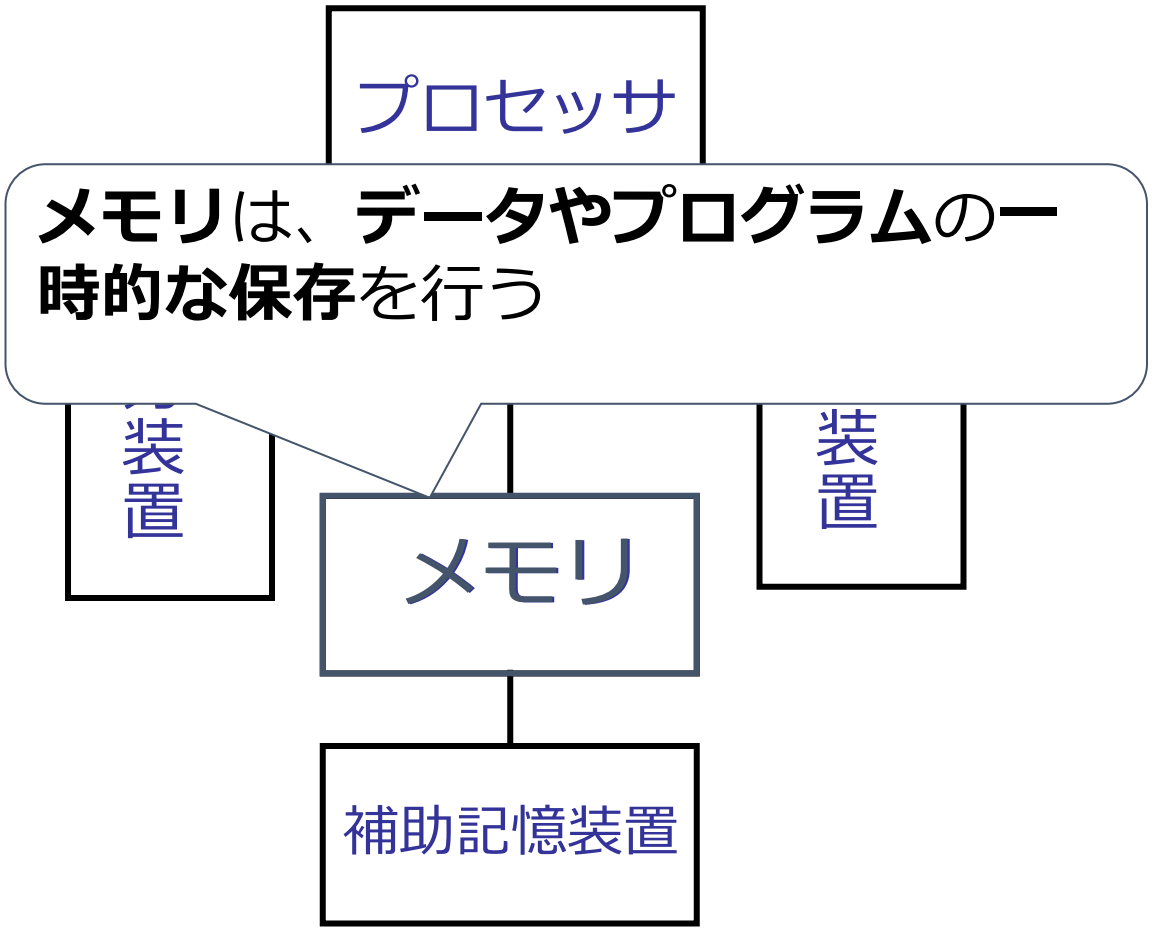


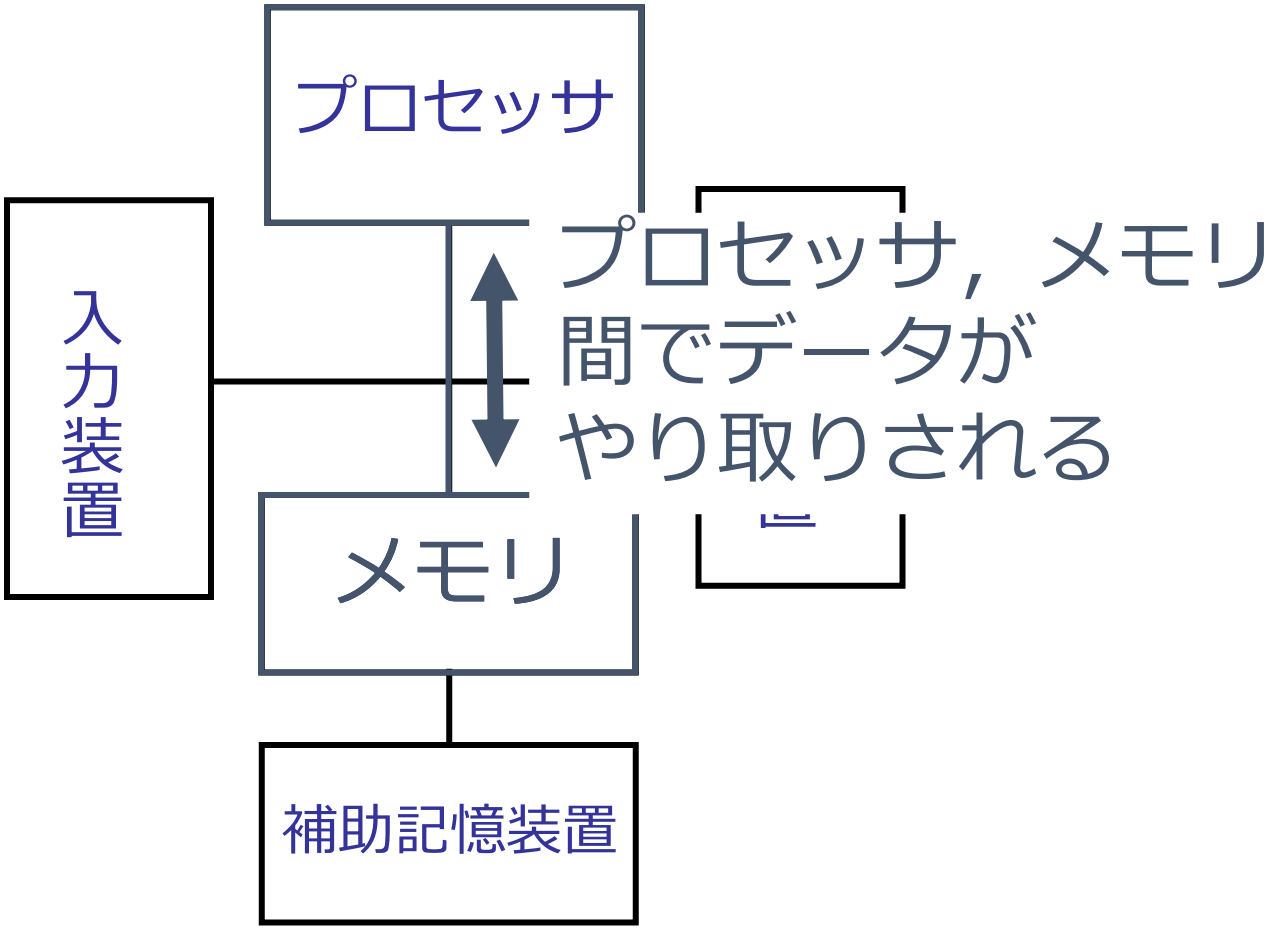
コンピュータの構成





プロセッサは、**コンピュータの「脳」**であり、すべての計算やデータの処理を行う


プロセッサ
メモリは、**データやプログラム**の**一時的な保存**を行う
装置
装置
メモリ
補助記憶装置


プロセッサ
入力装置
メモリ
補助記憶装置
プロセッサ，メモリ間でデータがやり取りされる

13-3 デジタル

デジタル

- **デジタル**と**二進数**は**コンピュータが情報を表現・処理**する基本的な方式

- **コンピュータ**は、**すべてのデータとプログラム**を二進数、つまり**「0」と「1」の形式で表現**

- すべてが**「0」または「1」の組み合わせ**、つまり**ビット列**で表現される

0 0 1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1

ビット

- **一つの「0」または「1」**は**一ビット（Bit）**と呼ばれる
- **ビット**は、**情報の最小単位**

00110101111101110101011　・・・ 23ビットのデータ

- **ビット**は**二進数の一桁**

１６進数

- **１６進数**は**データを効率的に表現**するための重要な概念

- １６個の記号0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F を使う

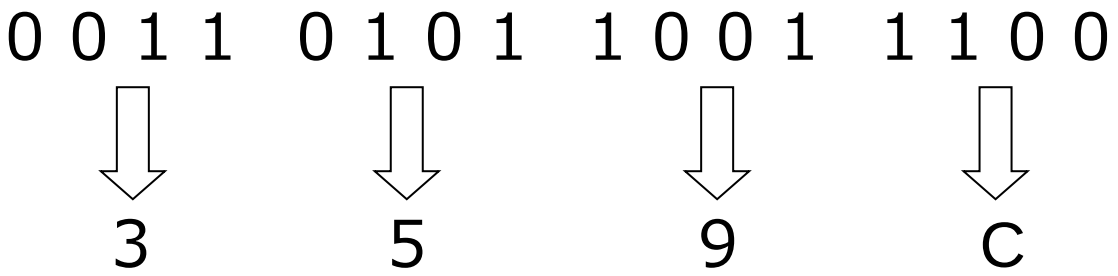
  例） 0065FDF0

１０進数と１６進数の対応

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	１０
B	１１
C	１２
D	１３
E	１４
F	１５

２進数と１６進数の関係

- **二進数の4桁**は、**16進数の1桁**に相当（**16が2の4乗**）

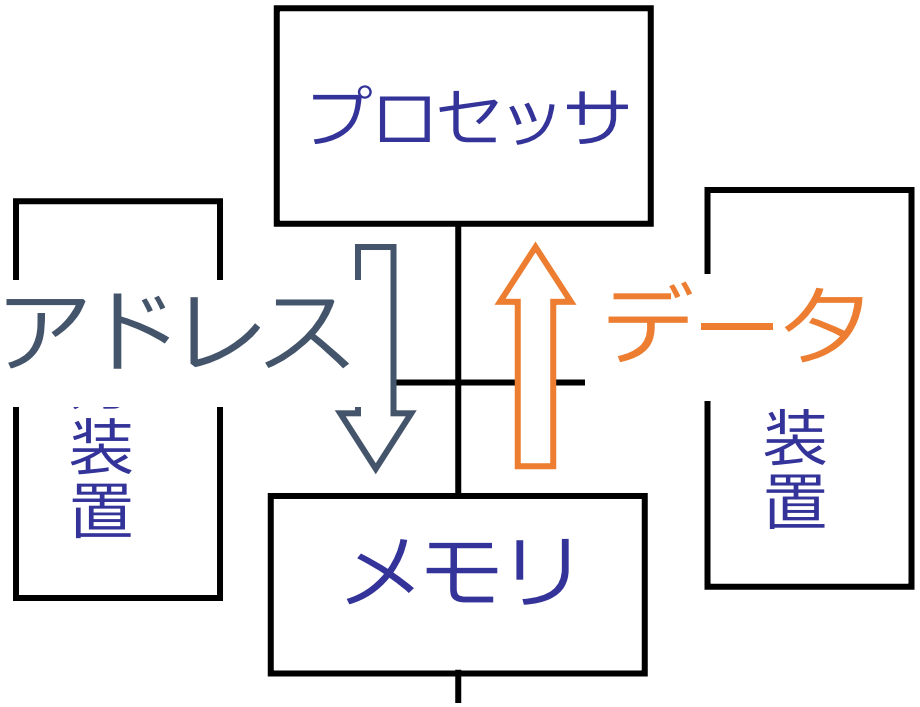


- 二進数の16桁は、16進数の4桁に相当

２進数と１６進数の関係

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

13-4 メモリとアドレス



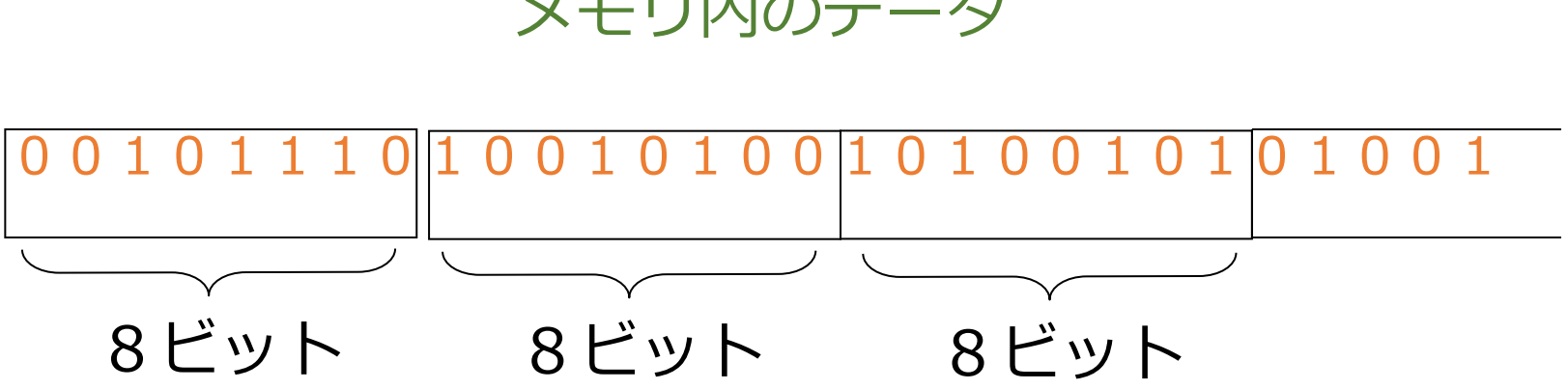
メモリからプロセッサへの読み出し

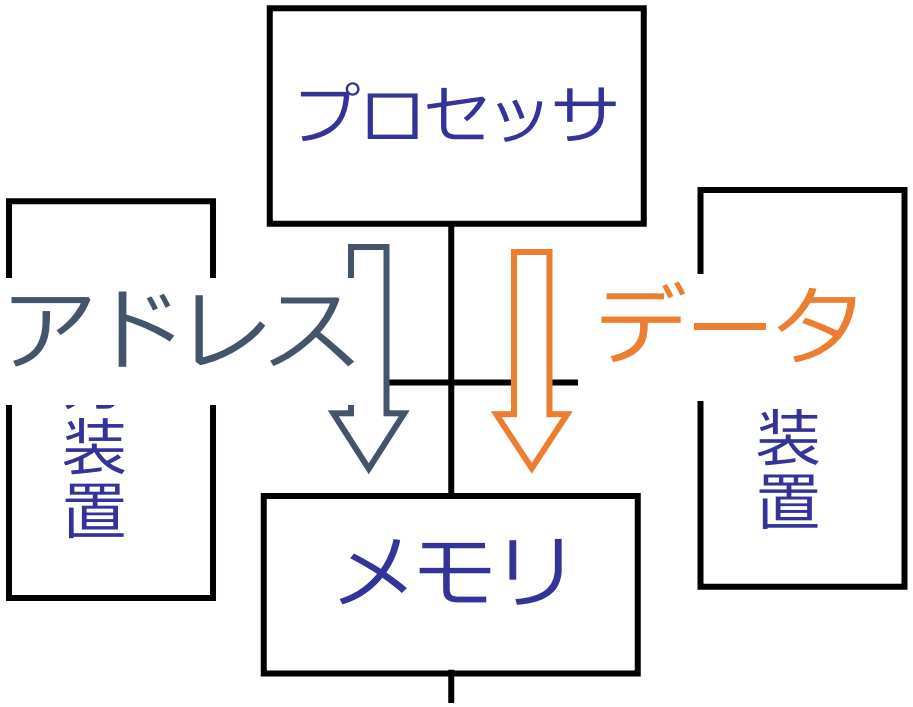
メモリとアドレス

- **メモリ**は**バイト（８ビット）**単位に**区切られて**いる

- 各**バイト**には**０から始まる通し番号**が付けられている。

  アドレスという（番地ともいう）



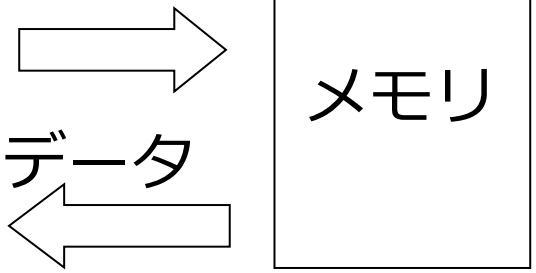


プロセッサからメモリへの書き込み

メモリへの操作

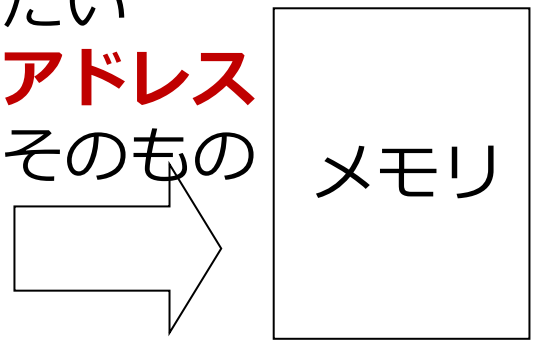
- **読み出し**

読み出したい
データの**アドレス**

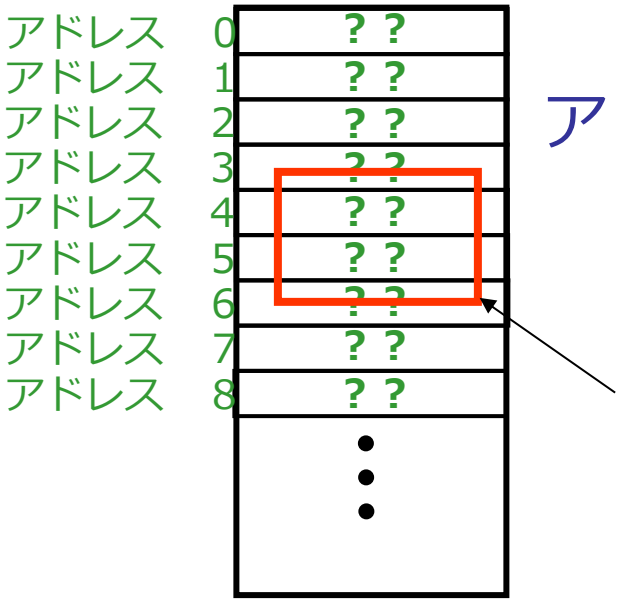


- **書き込み**

書き込みたい
データの**アドレス**
とデータそのもの



読み出し



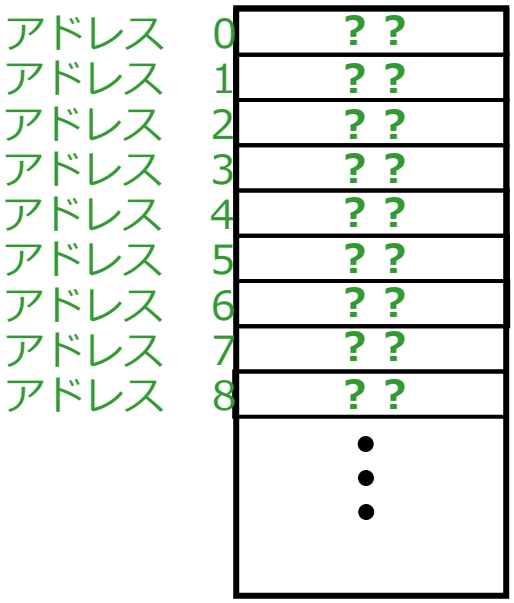
アドレス４番地，５番地
から２バイト分
読み出すとき

メモリの値は変化
しない

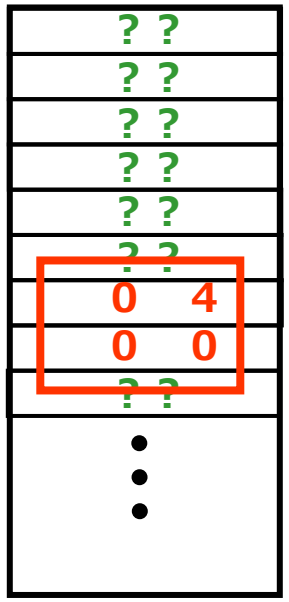
メモリの各区画は１バイト
（１６進数で２桁）

書き込み

アドレス６番地，
７番地に
「０４００」を
書き込むと

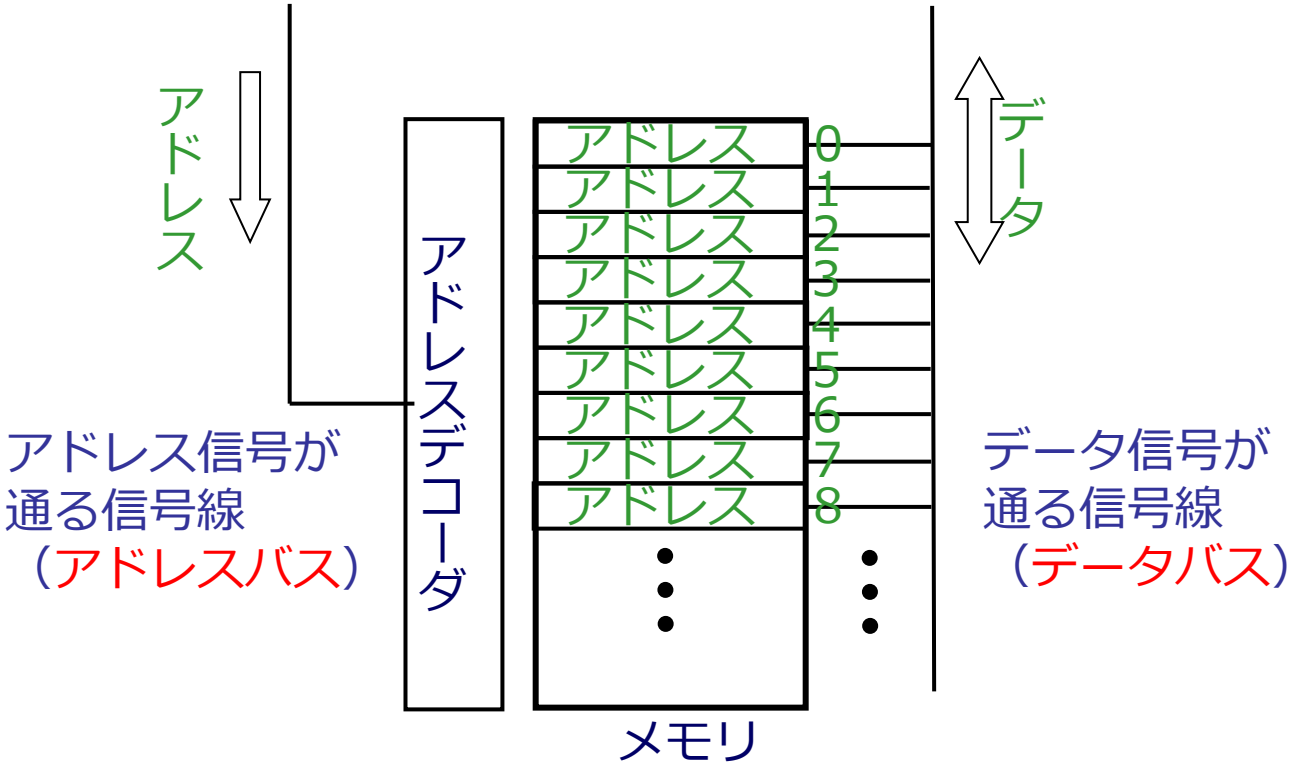


メモリの各区画は１バイト
（１６進数で２桁）

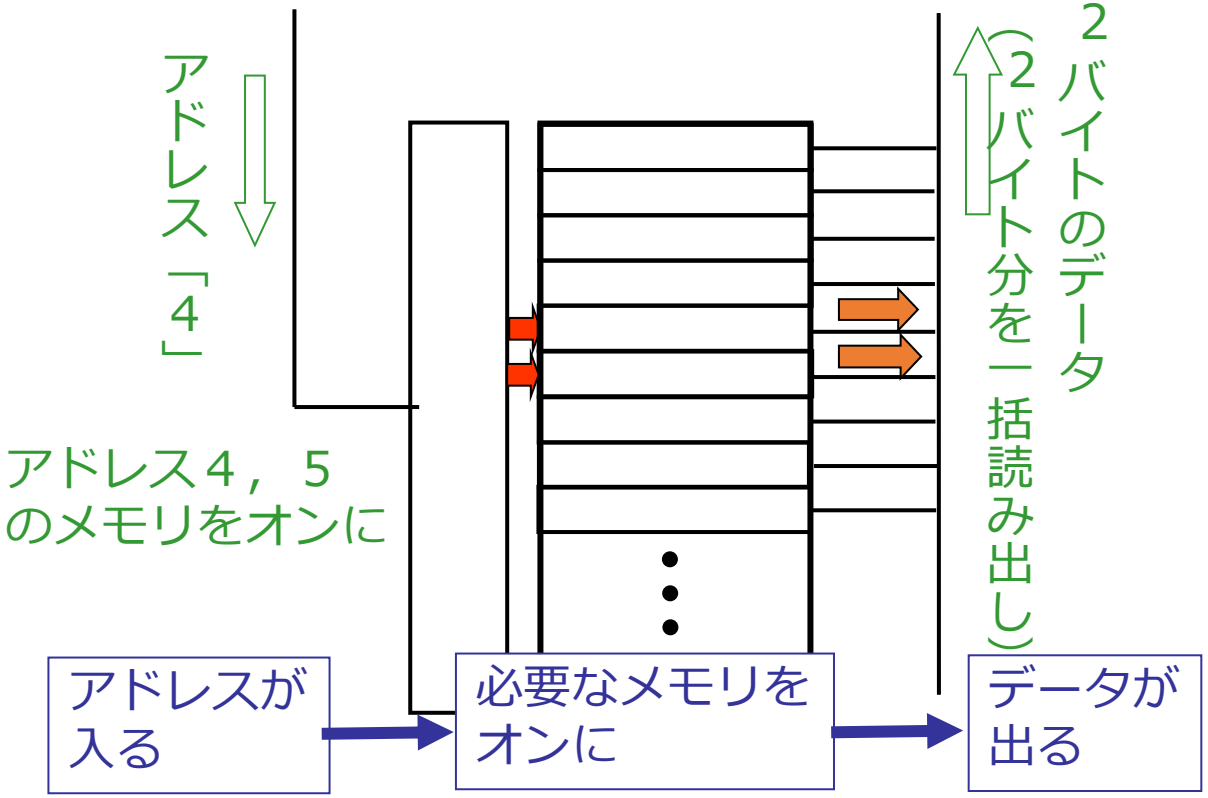


前の値は上書きされる

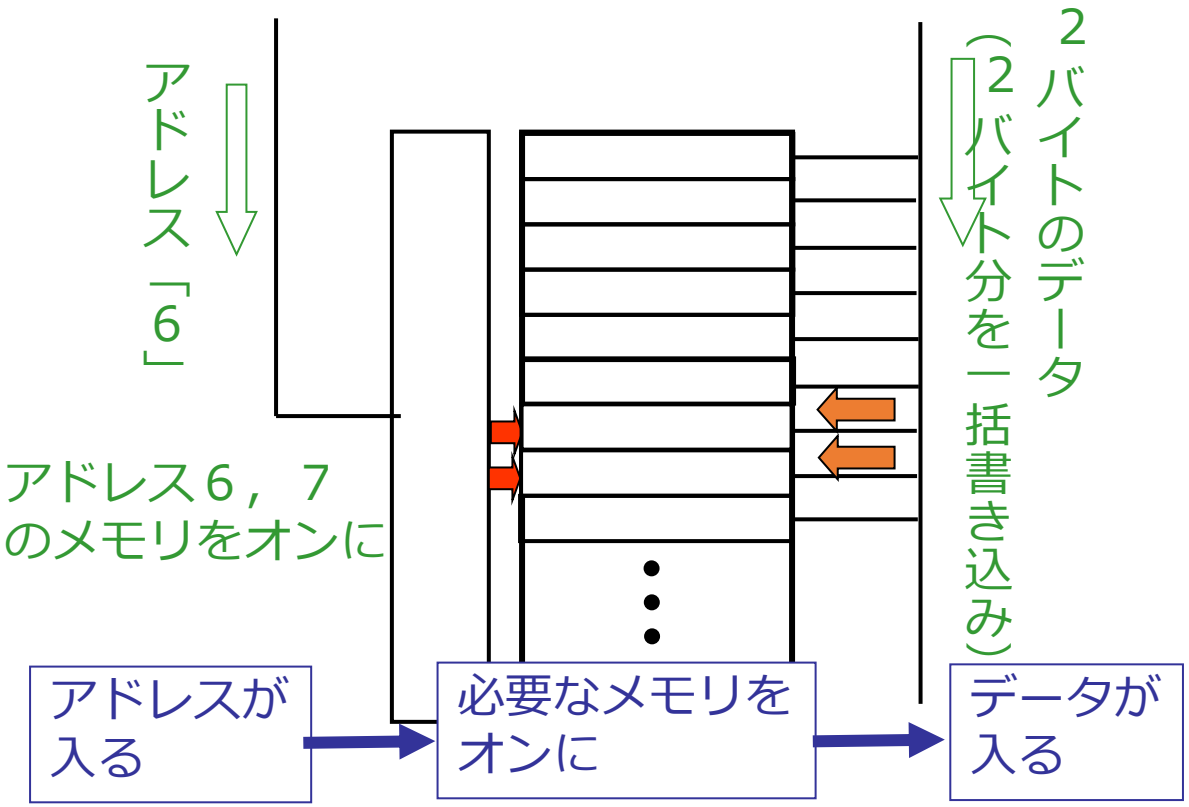
メモリの仕組み


アドレス
アドレスデコーダ
アドレス
アドレス
アドレス
アドレス
アドレス
アドレス
アドレス
アドレス
0
1
2
3
4
5
6
7
8
データ
アドレス信号が
通る信号線
（アドレスバス）
データ信号が
通る信号線
（データバス）
メモリ

アドレス４，５番地から，2バイト分読み出す



アドレス６，７番地に，2バイト分書き込む



13-5 文字コード

ASCII

- **ASCII**は**文字情報を数値で表現する**ためのもの
- １つの文字を，**7ビットの二進数**で表現
- 英数字や記号、制御文字など**128種類の文字**を表現することができる

	0	1	2	3	4	5	6	7
0	NULL	DEL	SP	0	@	P		p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	(BS)	CAN	(	8	H	X	h	x
9	(HT)	EM	)	9	I	Y	i	y
A	(LF)	SUB	*	:	J	Z	j	z
B	(VT)	ESC	+	;	K	[	k	{
C	(FF)	(FS)	,	<	L	¥	l	|
D	(CR)	(GS)	−	=	M	]	m	}
E	SO	(RS)	.	>	N	^	n	~
F	SI	(US)	／	?	O	_	o	DEL

演習１
文字コード

ページ３７～３８

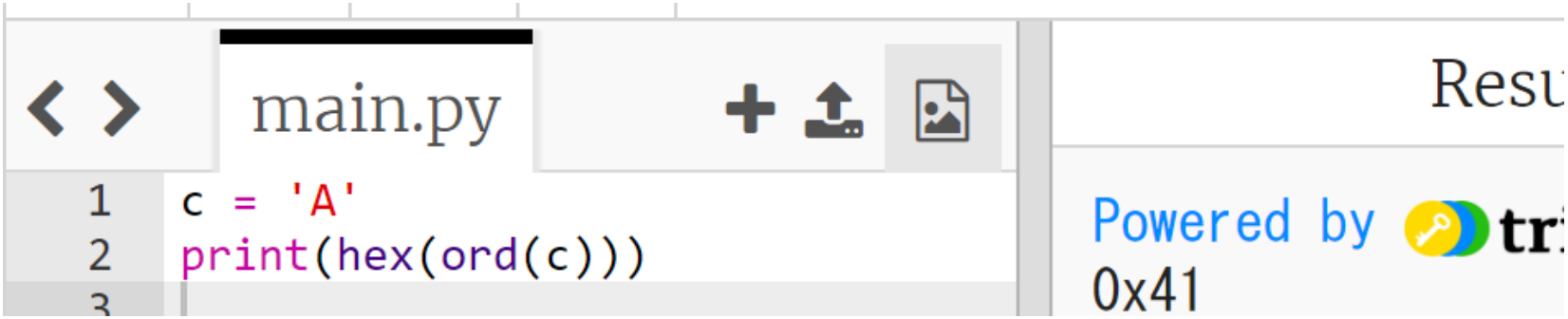
【トピックス】

- **trinket の利用**
- **文字コードの表示**

① trinket の次のページを開く

https://trinket.io/python/595c091dd9

② このプログラムは，**「A」の文字コードを表示**する．なお「**0x**」は，**１６進数を示す目印**である．実行結果が，次のように表示されることを確認

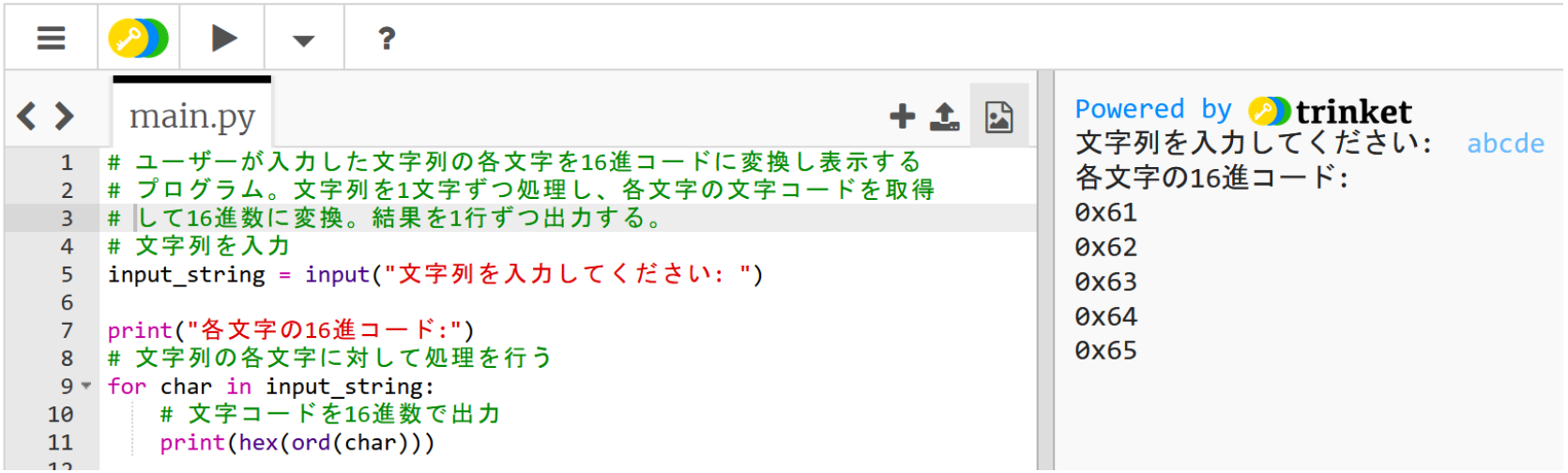


③ trinket の次のページを開く

https://trinket.io/python/595c091dd9

④ このプログラムは，あなたが入力した文字や記号の並び（例： **abcde**）を文字コードに変える。

プログラムを実行したら、左側の画面をクリックし、文字や記号の並びを入力して、Enter キーを押す。
コンピュータが文字をどう扱うか知るのに役立つ。



13-6 論理積と論理和

二進数は 0 または 1

右手が下がっている

右手が上がっている

二通り

二進数は 0 または 1

右手が下がっている

0

右手が上がっている

1

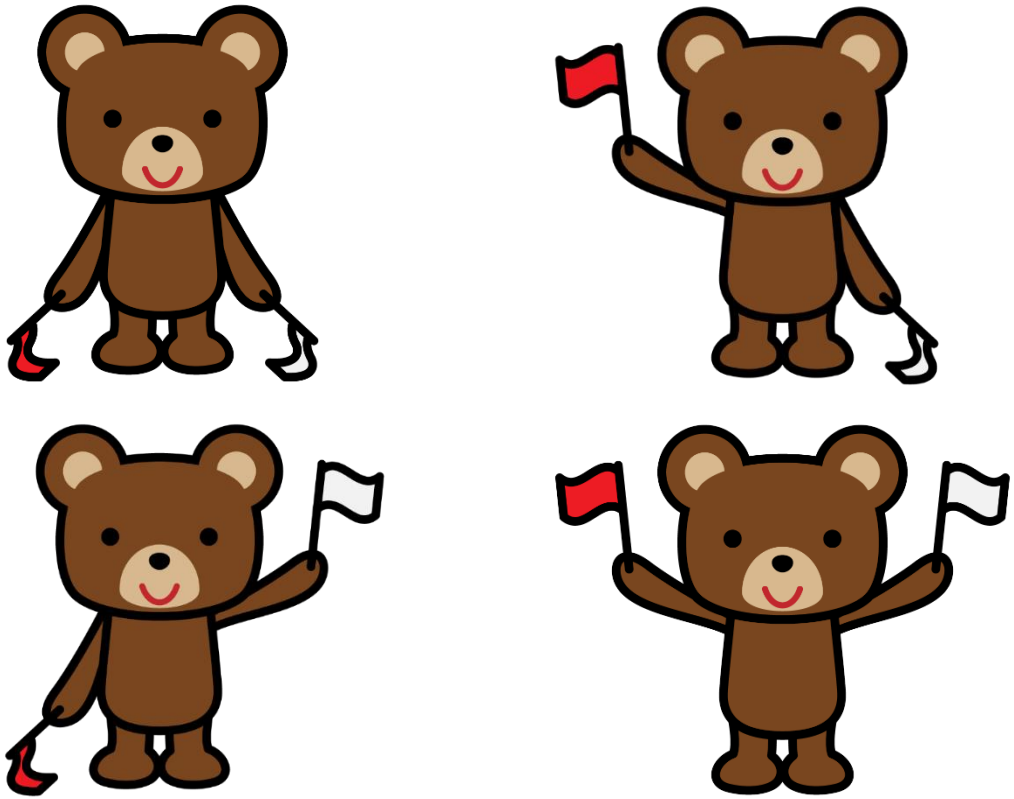
※ 0 と 1 が逆になる場合もある

変数が２つ

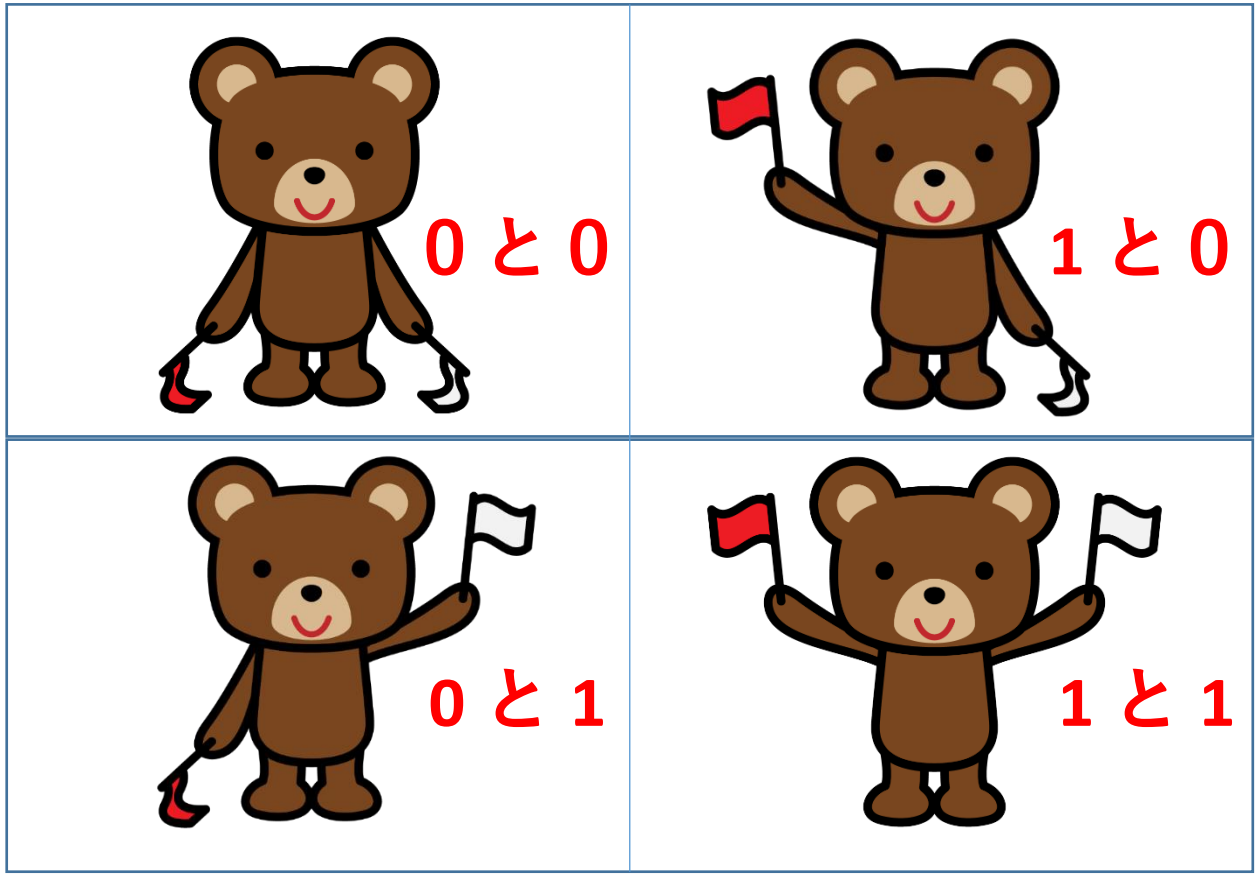
右手と左手の
両方を考えると

↓

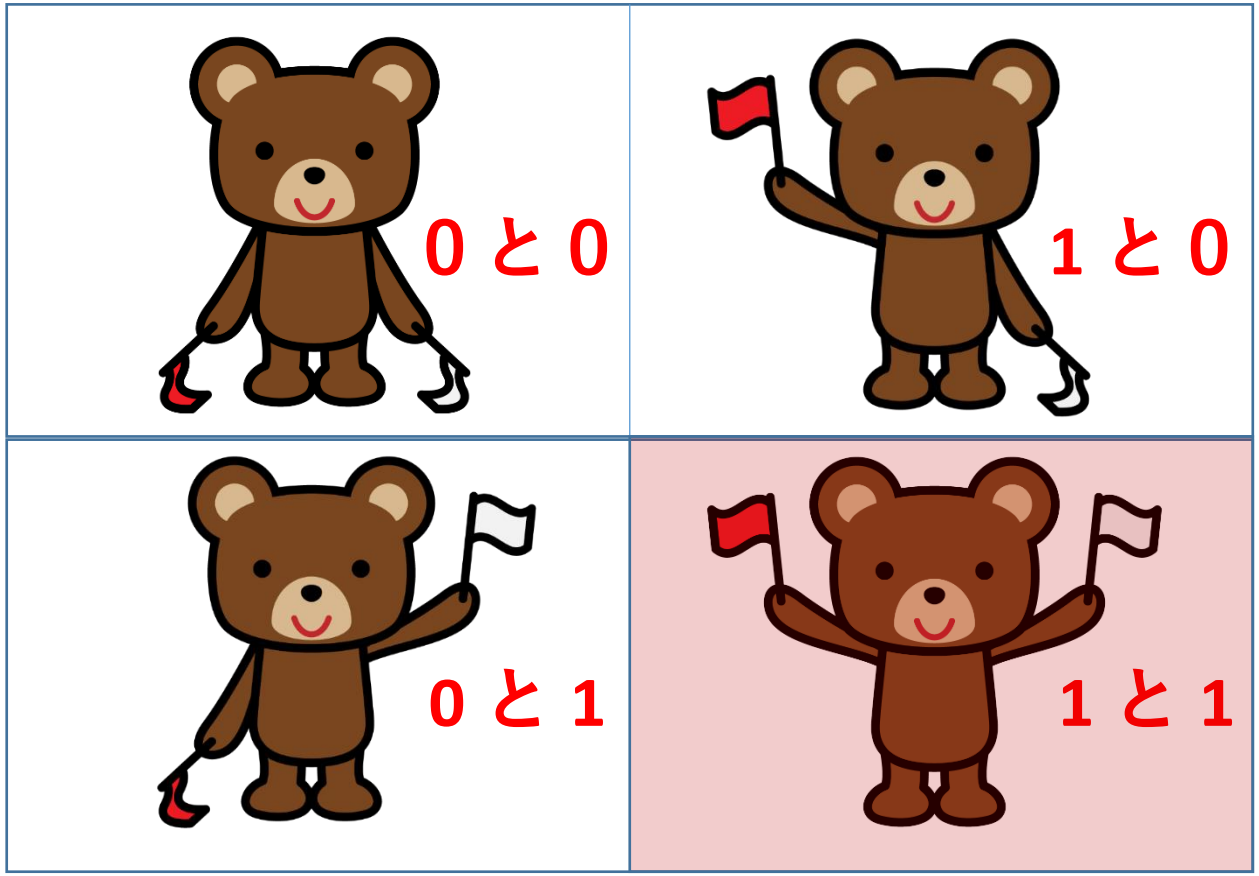
４通り



変数が２つ

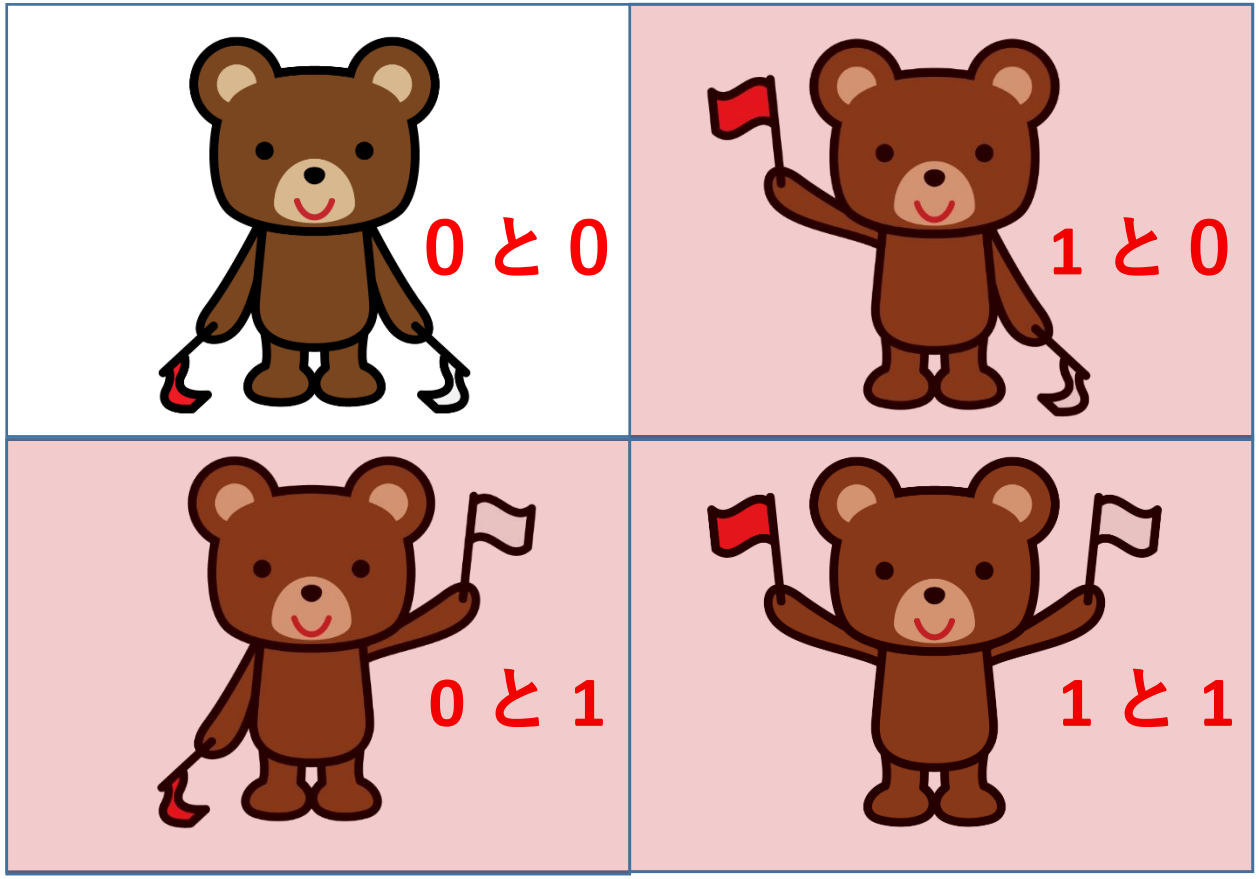


論理積



論理積は
両方とも 1

論理和



論理和は
少なくとも
片方には
１がある

論理和と「選択」は違う

・焼き芋大会があるんだけど、
・土曜日と日曜日、どっちが良い？

両方、申し込んでよ♡

土曜日＼日曜日	落選：0	当選：1
落選：0		
当選：1		

<u>両方参加しても OK!</u>
土曜日と日曜日の選択では**<u>無い</u>**

論理積と論理和のまとめ

	0	1
0	0	0
1	0	1

論理積 AND

	0	1
0	0	1
1	1	1

論理和 OR

演習２
論理演算

ページ４９～５０

【トピックス】

- **trinket の利用**
- **AND**
- **OR**

① trinket の次のページを開く

https://trinket.io/python/7f31113af9

② このプログラムは，AND と OR の結果を表示する．実行結果が，次のように表示されることを確認

```
1  a = True
2  b = False
3
4  print("a and b =", a and b)
5  print("a or b =", a or b)
```

```
Powered by trinket
('a and b =', False)
('a or b =', True)
```

複数のビットを一括して論理積、論理和を
求める場合があります

x　0 0 1 1

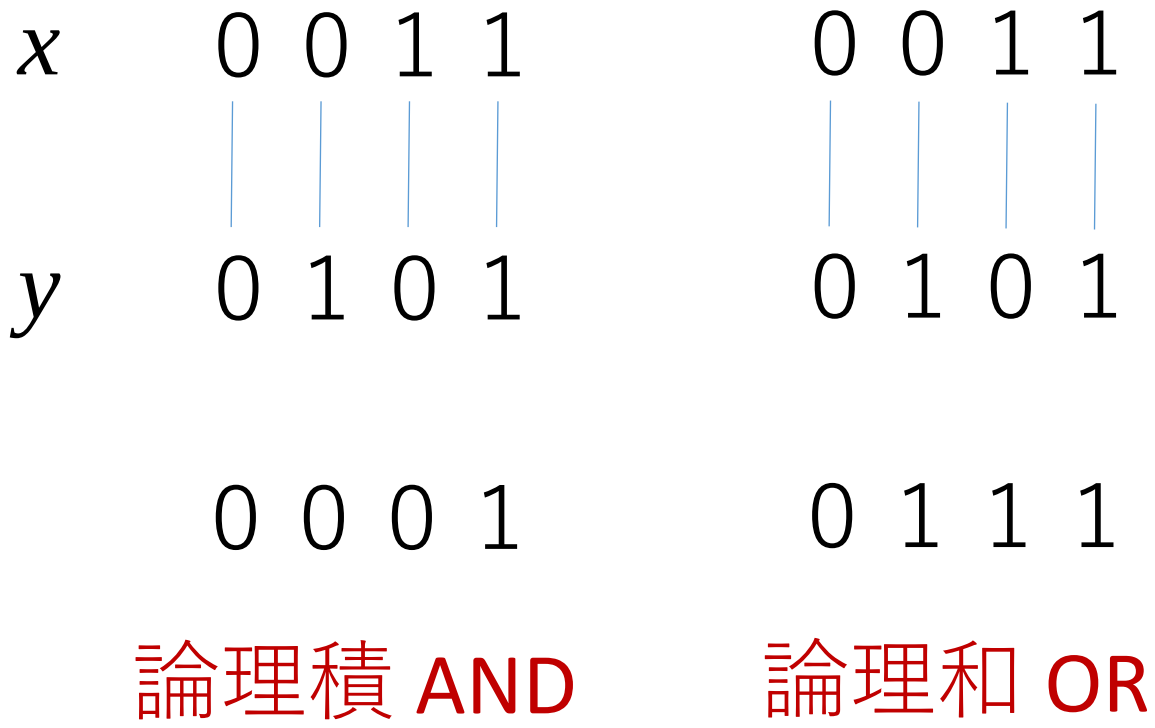
y　0 1 0 1

全部で４ビット

クイズ

x と y の論理積は？
論理和は？

複数のビットを一括して論理積、論理和を求める場合があります



多数決

Aさん，Bさん，Cさんは「0」か「1」に投票

A	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1
C	0	0	0	0	1	1	1	1
多数決	0	0	0	1	0	1	1	1

多数決は，論理積（AND）と論理和（OR）の組み合わせでできる

A	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1
C	0	0	0	0	1	1	1	1
多数決	0	0	0	1	0	1	1	1

A AND B	0	0	0	1	0	0	0	1 ①
B AND C	0	0	0	0	0	0	1	1 ②
C AND A	0	0	0	0	0	1	0	1 ③
① OR ②	0	0	0	1	0	0	1	1 ④
③ OR ④	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**

13-7 論理演算と足し算

論理積と論理和

	0	1
0	0	0
1	0	1

論理積 AND

	0	1
0	0	1
1	1	1

論理和 OR

否定（NOT)

NOT(0)　=　1

NOT(1)　=　0

否定 NOT

足し算

0	+	0	=	00
0	+	1	=	01
1	+	0	=	01
1	+	1	=	10

2つのビットから　　2ビットができる

足し算

				上位ビット	下位ビット
0	+	0	=	0	0
0	+	1	=	0	1
1	+	0	=	0	1
1	+	1	=	1	0

足し算は
AND, OR, NOT
の組み合わせで可能

	0	1
0	0	0
1	0	1

上位ビット
x AND y

	0	1
0	0	1
1	1	0

下位ビット
AND((x OR y), NOT(x AND y))

論理演算（AND, OR, NOT）

- 複雑な算術演算も、**単純な論理演算の組み合わせ**で実現できる

例：足し算は AND, OR, NOT の組み合わせで可能

- **コンピュータ**は**論理回路の組み合わせ**だけで、複雑な計算や処理を行うことができる
- コンピュータの基本的な動作原理を理解する上で，この知識は不可欠である

13-8 ２の補数

２の補数

- ２の補数は，負の整数も扱いたいときに便利
- ２の補数では，最上位ビットが符号ビット

０　→　正の整数または０

１　→　負の整数

８ビットの整数データの場合

10進数の２	0	0	0	0	0	0	1	0
10進数の１	0	0	0	0	0	0	0	1
10進数の０	0	0	0	0	0	0	0	0
10進数の－１	1	1	1	1	1	1	1	1
10進数の－２	1	1	1	1	1	1	1	0

－４５　と　４５　を足すと　０

８ビットの２の補数

45	0	0	1	0	1	1	0	1
-45	1	1	0	1	0	0	1	1
1								
45 + (-45)	0	0	0	0	0	0	0	0

２の補数では，
マイナスの数は
最上位ビットが１

１繰り上がる

全体まとめ

- **OS（オペレーティングシステム）**は、**ハードウェア**と**ソフトウェア**の間で**インターフェイス**を提供する。
- **プロセッサ**は、**コンピュータの頭脳**であり、計算やデータの処理を行う。
- **メモリ**は、**一時的にデータやプログラムを保存する場所**で、プロセッサが直接アクセスして読み書きを行う。**各バイト（8ビット）**には**通し番号（アドレス）が割り当てられている。**
- **16進数**は**データを効率的に表現するための重要な概念**で、16個の記号（0～F）を使用する。
- **文字コード**は、**文字（'A'や'0'、'!'、'あ'など）**を**数字に置き換える**ルール。
- **論理演算**（AND, OR, NOTなど）は**ビットに関する演算**で、**情報処理の基礎**となる。
- **複数のビットに対する論理演算**もあり、例えば**多数決**などは**論理積（AND）と論理和（OR）の組み合わせ**で実現できる。
- **2の補数**は、**負の整数**を扱うために使用される。**最上位ビットが符号ビット**となり、**0なら正または0**、**1なら負**を表す。

今回の授業の学ぶ意義と満足感

①**技術的理解と基礎スキルの向上**：コンピューターの基本原理（OS、プロセッサ、メモリ）、デジタルデータの表現（二進数、十六進数、文字コード）、論理演算、数値表現（2の補数）

② プログラミングやデータ分析の基礎となる**デジタルリテラシーの向上**

③ **テクノロジーへの興味の向上：コンピューター内部動作の理解**によるテクノロジーへの興味向上、デジタル社会での自信と適応力の向上

④ **将来の新技術への適応力：確かな基礎を理解.** そのことで，将来の技術革新にも対応.