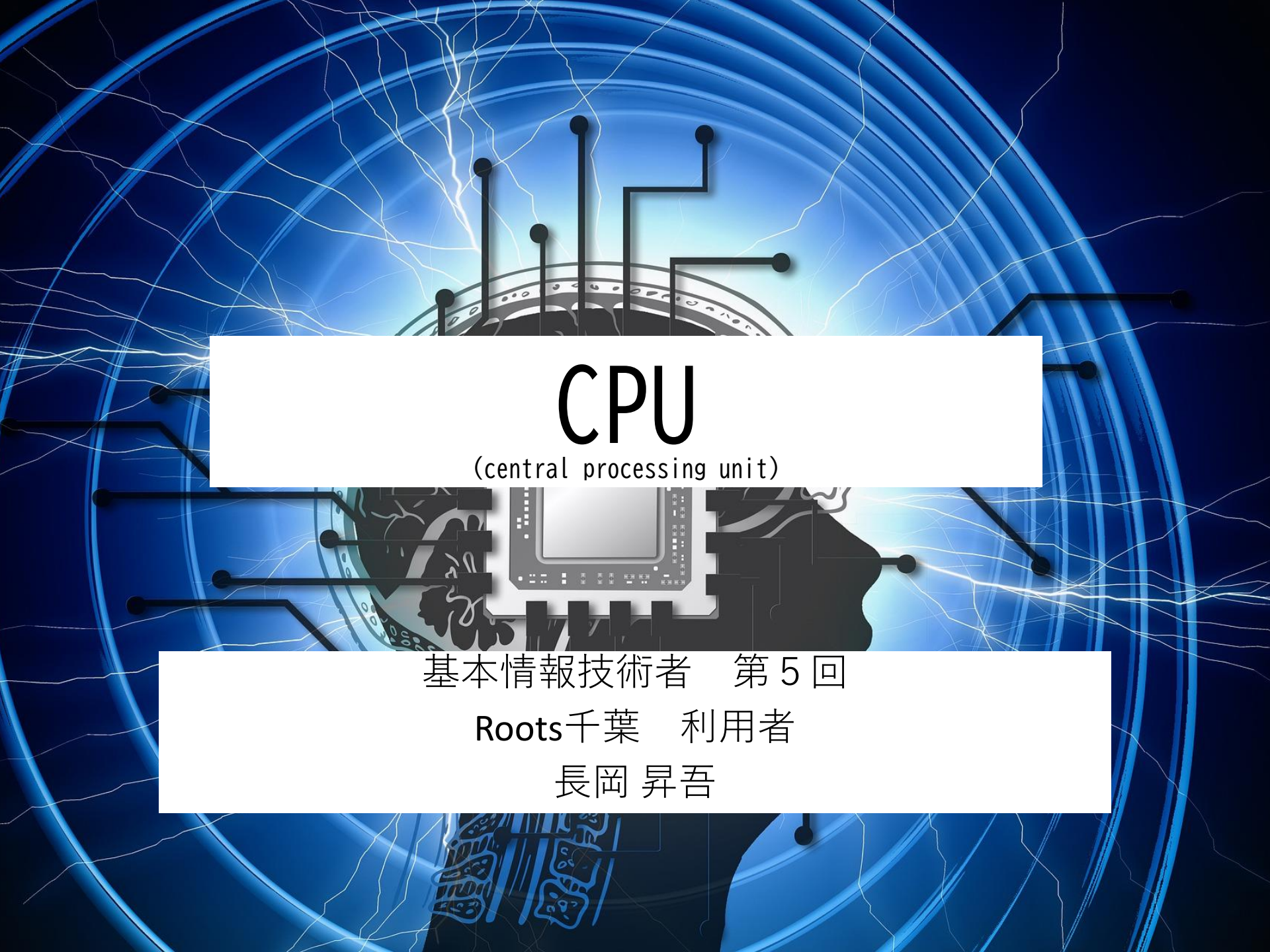


CPU

(central processing unit)

基本情報技術者　第５回

Roots千葉　利用者

長岡 昇吾

今回の章の進み方

スライド
①CPUについて
(実際にパソコンに使われているCPUを見てみる)

②コンピュータの５台装置
(CPUの制御、揮発性と不揮発性)

③ノイマン型コンピュータ
(プログラム内蔵方式、逐次方式、アドレス)

④CPUの命令実行手順
(レジスタ、フェッチ、プログラムカウンタ、命令デコーダ)

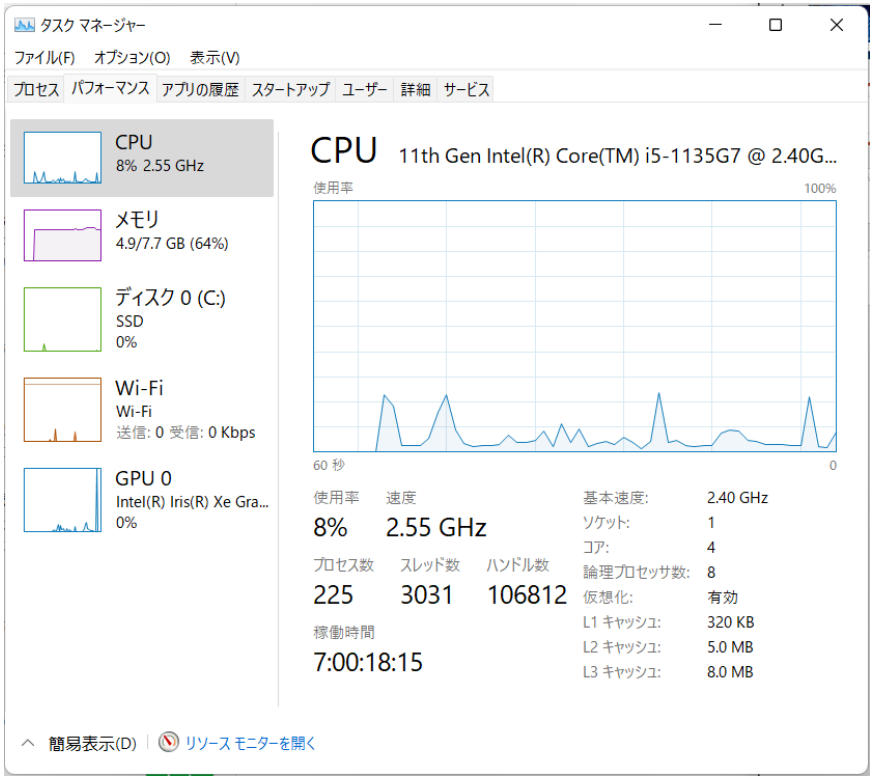
⑤機械語のアドレス指定方式
(機械語とは、それぞれのアドレス指定方式)

⑥CPUの性能指標
(クロック、CPI、MIPS、命令ミックス)

⑦CPUの高速化技術
(パイプライン処理、CISC、RISC)

CPUについて

CPUとは中央処理装置といわれる部分でコンピュータにおいてマウスやキーボードなどの処理を行う部分です。



皆さんのパソコンではシステムなどのあらゆる場所で確認することができます(cmd中にwmic cpu list fullと入力してみてください。)

CPUについて

cpuの細かい情報を見ることが可能です。
自分のパソコンについて、
皆さんも見てみましょう。

CPUの種類

Intel

- Xeon
- Coreシリーズ(i9,i7,i5,i3) (x86)
- Pentium
- Celeron

AMD

- EPYC
- Ryzen(threadripper,9,7,5)(Zen2)

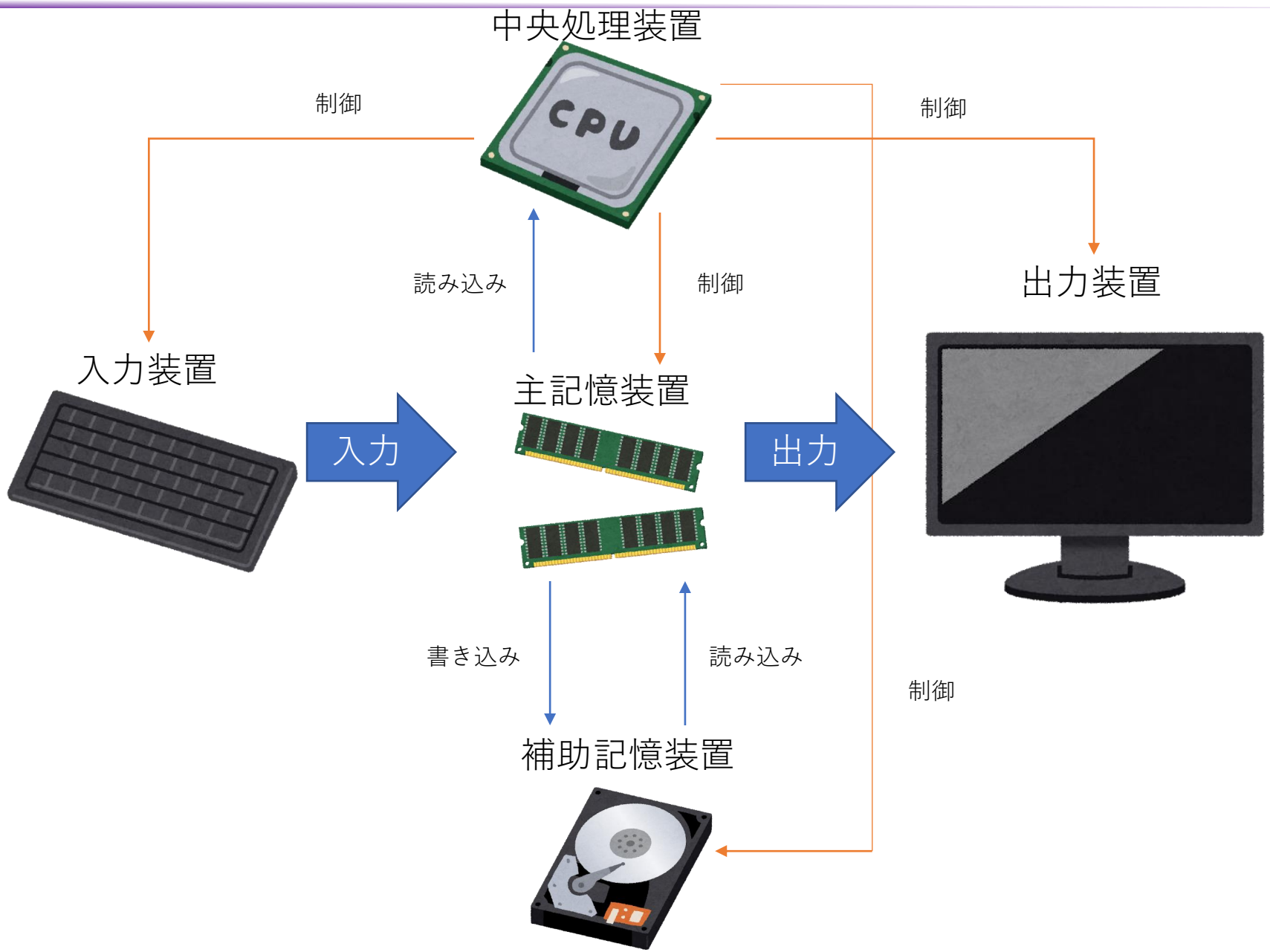
コンピュータの５台装置

自作パソコンを作る必要なものです。

（中身としてはケースファン、CPU、グラフィックカード、OS、メモリ、SSD、電源装置、CPUグリス、マザーボード、CPUクーラー、ケース）

今回は主に重要なCPU、主記憶装置（メモリ）、補助記憶装置(HDD、SSD)、入力装置(マウス、キーボード)出力装置(ディスプレイモニター、プリンター)について説明を行います。

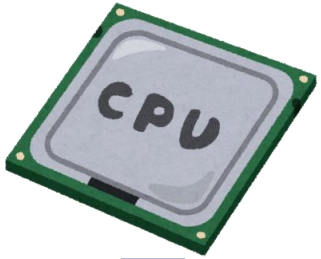
コンピュータの５台装置



コンピュータの５台装置

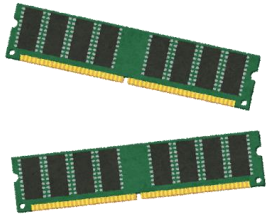
パソコン内における役割を簡単に表すと

CPU



作業員

メモリ



机

HDD

本棚

メモリは揮発性のものを使っているため電源を切ると中のデータが飛んでしまうため、保存ボタンによって内容をHDDやSSDに保存する。

ノイマン型コンピュータ

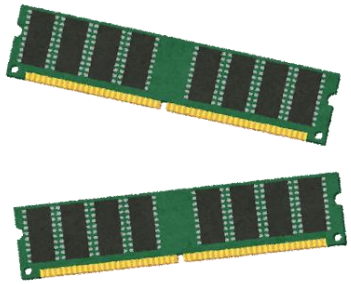
ノイマン型コンピュータとは、プログラムをデータとして記憶装置に格納し、これを順番に読み込んで実行するコンピュータのことです。

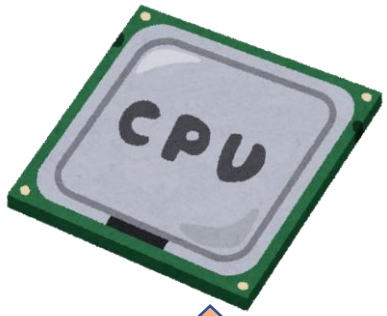
ジョン・フォン・ノイマンが提唱したコンピュータの基本構成である。
(計算手順や入力値をハードウェアから独立させて、外部からデータを入れて処理する方式を考え出した)

ノイマン型コンピュータ

データは基本的にHDDに蓄えられている

プログラム内蔵方式	逐次制御方式
必要なプログラムを入れておく	プログラムを命令に分けて順番に処理する

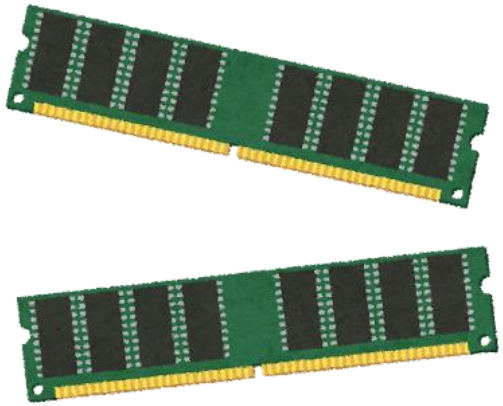




プログラム

命令１
命令２
プログラム

ノイマン型コンピュータ

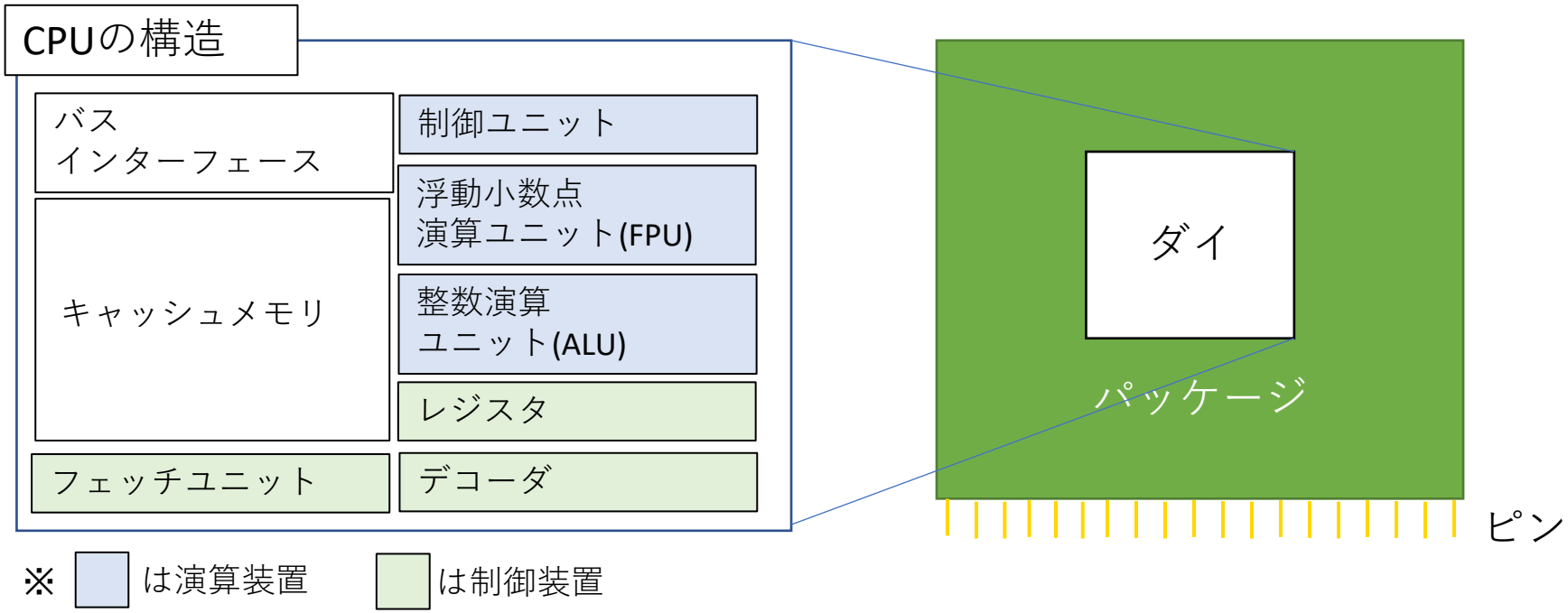


主記憶装置にはプログラム以外にも処理中の計算結果などが格納されています。

CPUが演算するのにどこにどのデータが入っているか分からなくては…

アドレス	メモリ空間
0000	
0001	
0002	
0003	
…	
FFFF	

CPUの命令実行手順



・バスインターフェース　・・・　データのやり取りをする伝送路(内部バスと外部バス)

・キャッシュメモリ　・・・　バスインターフェースから受け取ったデータを格納しておく場所

・フェッチユニット　・・・　メモリ上の命令を外部バスインターフェースを通じて制御ユニットに読み込む

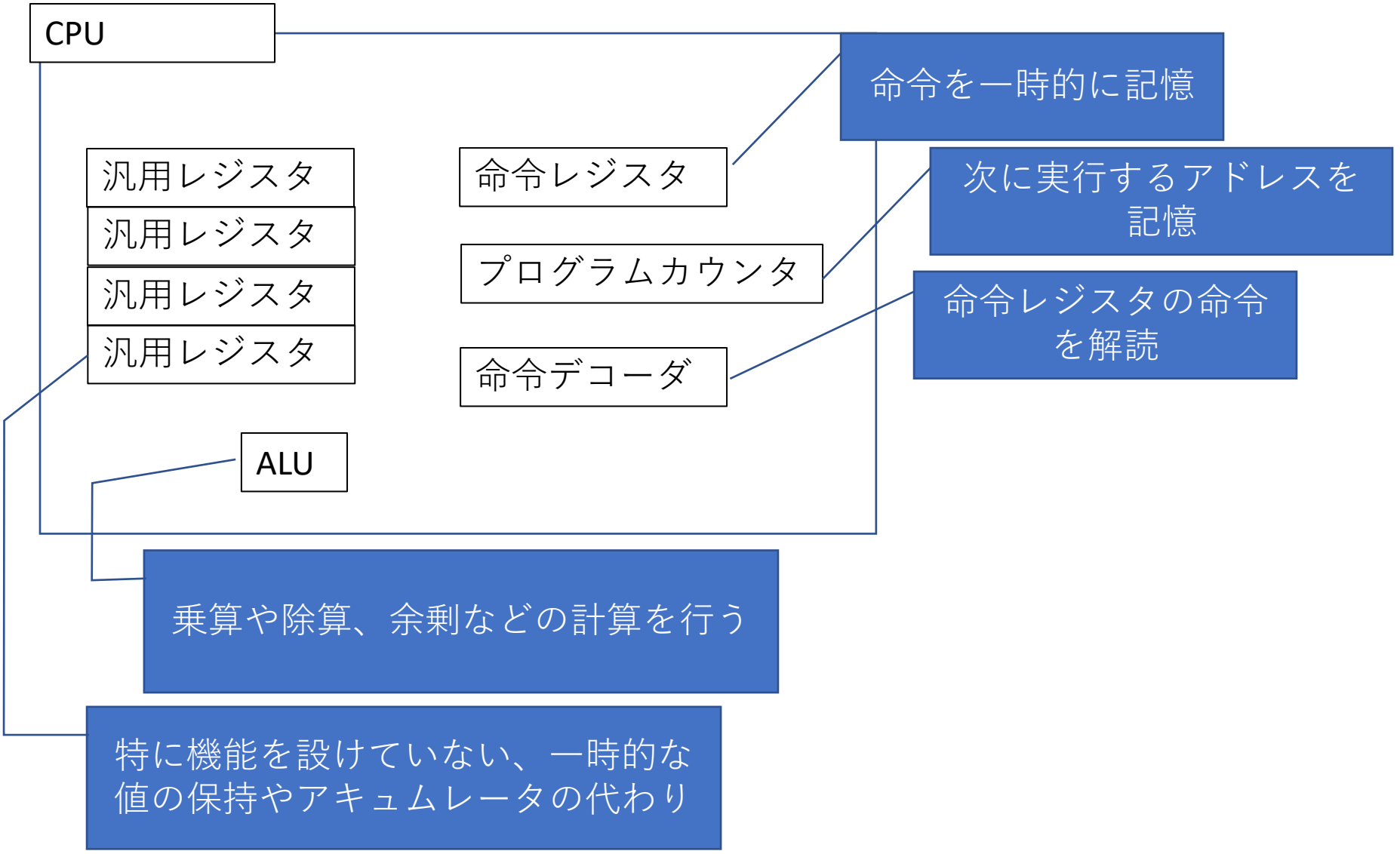
・デコーダ　・・・　フェッチユニットでフェッチされた命令を具体的な情報に解説する部分

・演算装置(制御ユニット、浮動小数点演算ユニット、整数演算ユニット)　・・・演算を行う部分
・レジスタ　・・・　途中経過などを保管しておいたり、メモリのアドレスを示したりする場所

CPUの命令実行手順

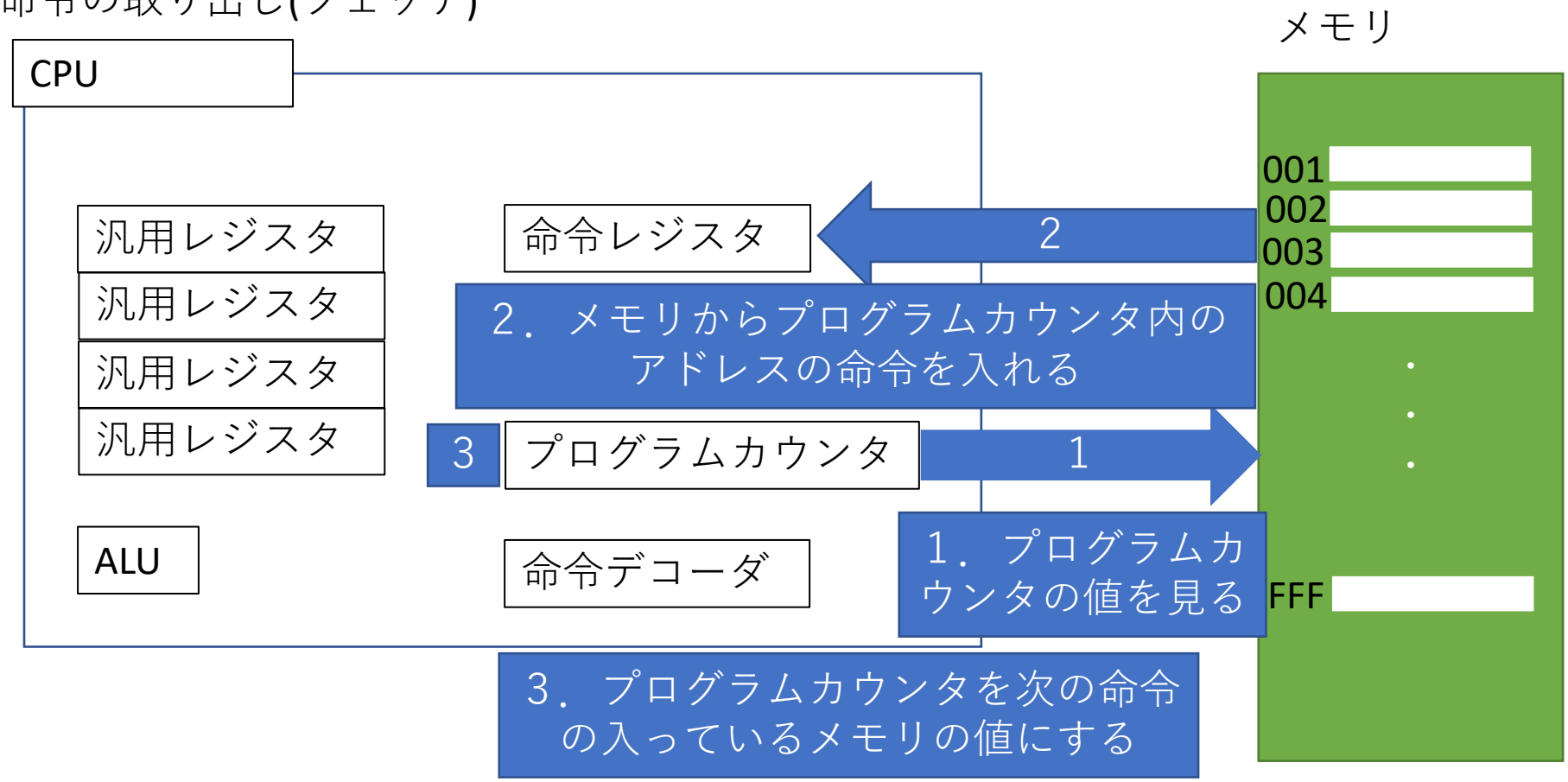
名称	役割
プログラムカウンタ	次に実行するべき命令が入っているアドレスを記憶するレジスタ
命令レジスタ	取り出した命令を一時的に記憶するためのレジスタ
インデックスレジスタ	アドレス修飾に用いるためのレジスタで、連続したデータの取り出しに使うための増分値を保持する。
ベースレジスタ	アドレス修飾に用いるためのレジスタで、プログラムの先頭アドレスを保持する。
アキュムレータ	演算の対象となる数や、演算結果を記憶するレジスタ
汎用レジスタ	特に機能を限定していないレジスタ。一時的な値の保持や、アキュムレータなどの代用に使ったりする。

CPUの命令実行手順


CPU
汎用レジスタ
汎用レジスタ
汎用レジスタ
汎用レジスタ
命令レジスタ
プログラムカウンタ
命令デコーダ
ALU
命令を一時的に記憶
次に実行するアドレスを記憶
命令レジスタの命令を解読
乗算や除算、余剰などの計算を行う
特に機能を設けていない、一時的な値の保持やアキュムレータの代わり

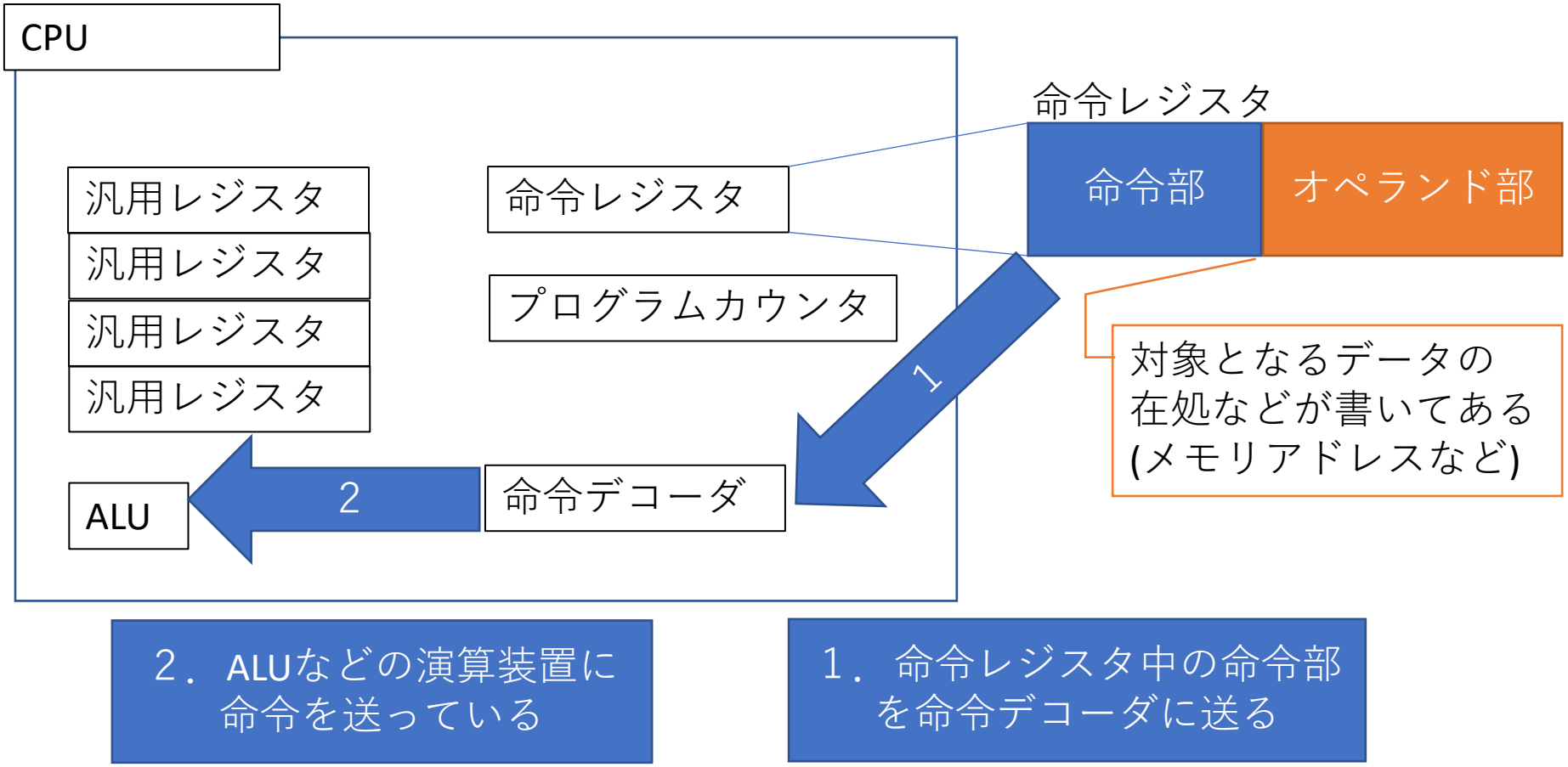
CPUの命令実行手順

命令の取り出し(フェッチ)



CPUの命令実行手順

命令の解読

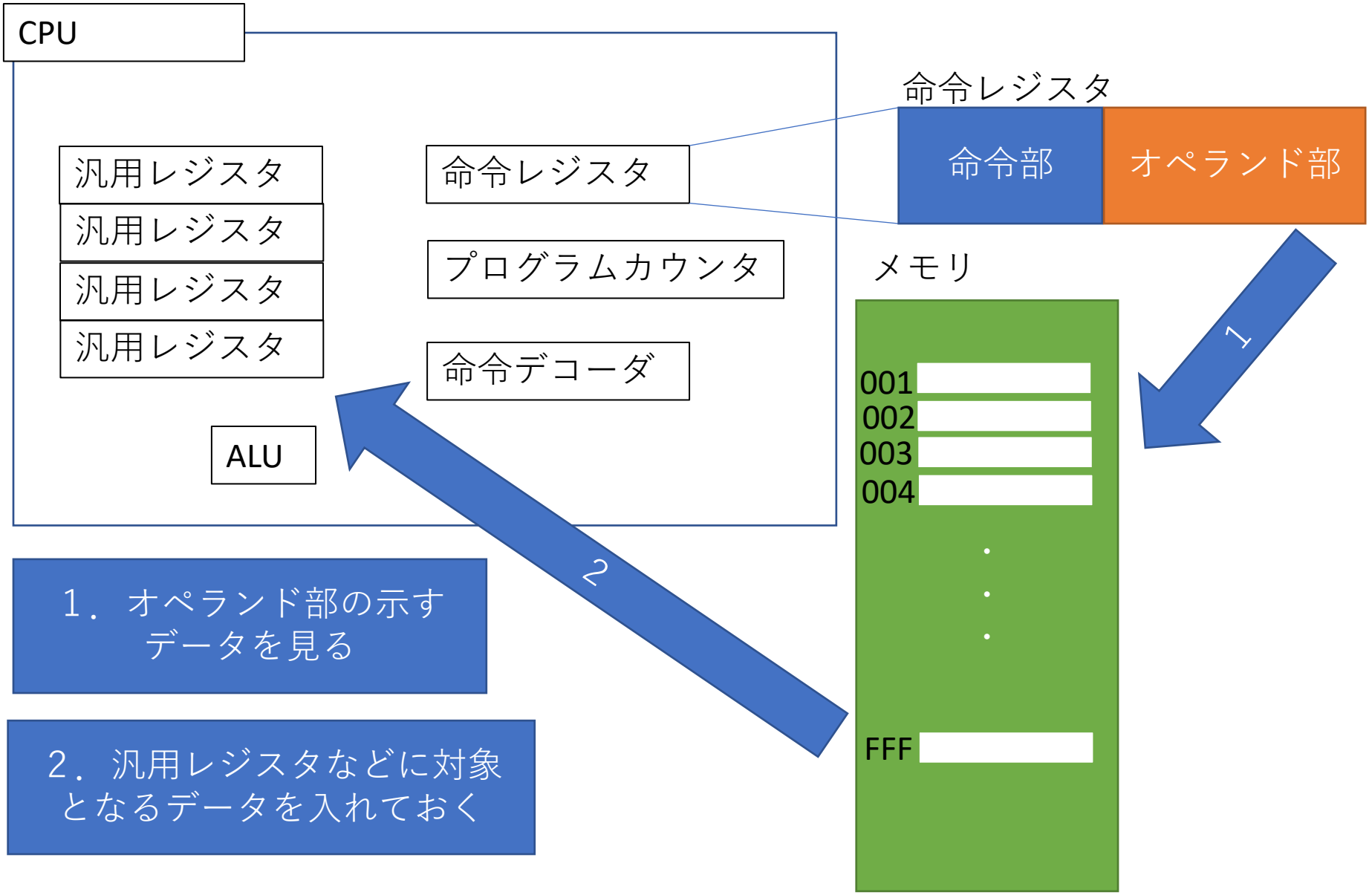


2．ALUなどの演算装置に命令を送っている

1．命令レジスタ中の命令部を命令デコーダに送る

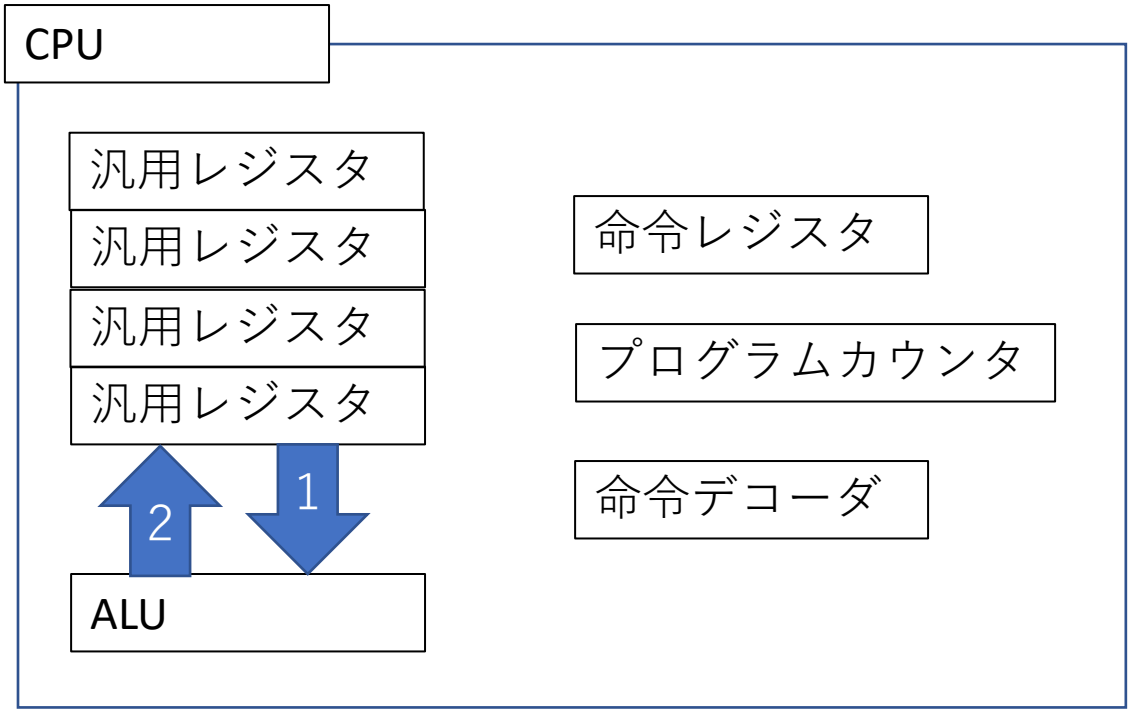
CPUの命令実行手順

対象データ(オペランド部)読み出し



CPUの命令実行手順

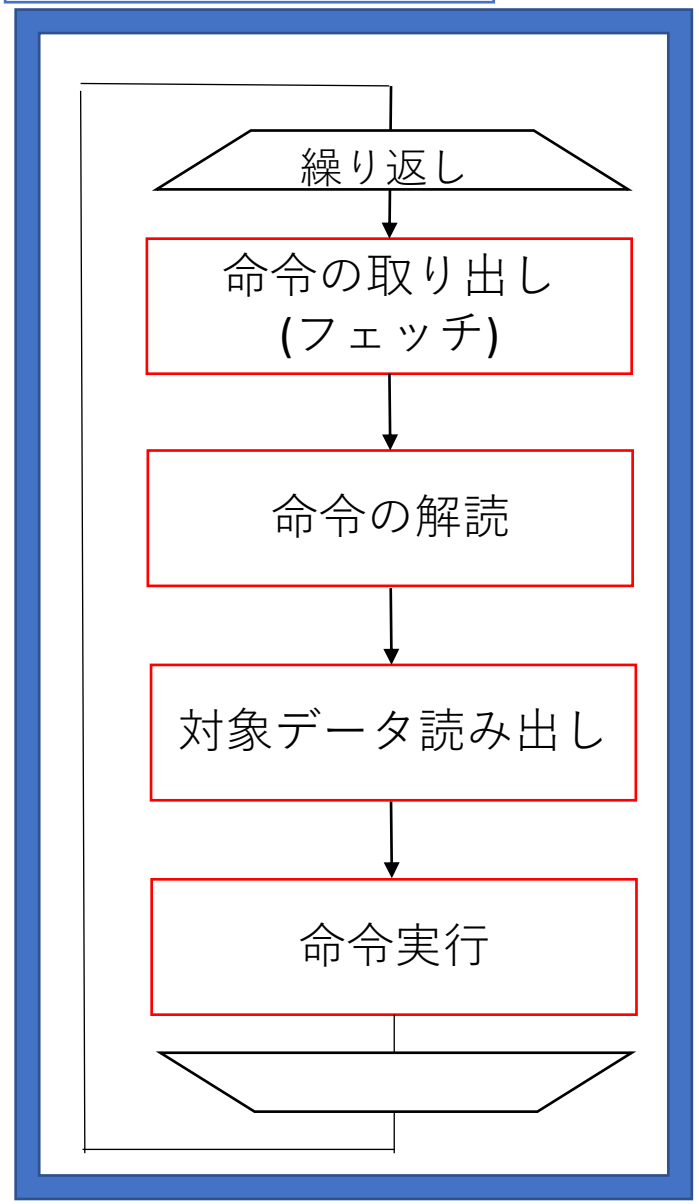
命令実行



1．汎用レジスタなどに格納されたデータを演算装置に読み込む

2．演算結果を汎用レジスタなどに書き戻す

CPUの命令実行手順



機械語のアドレス指定方式

機械語

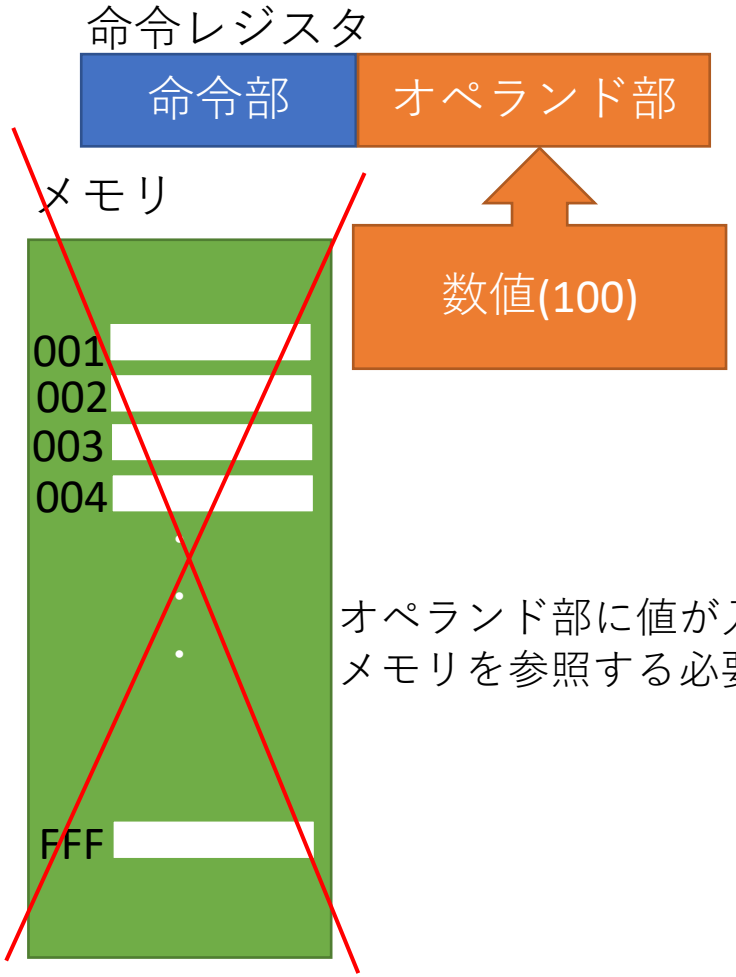
機械語とは、１と０で出来たコンピュータが理解できる命令語となります。
人は理解できないので、下のように理解できる形に持っていきます。
機械語→アセンブリ言語(アセンブラ)→C言語

スライドでは…

オペランド部は対象データの在処を示しているため、命令レジスタでの命令は「何を(オペランド部)どうしろ(命令部)」となる事が分かる。
ただ、オペランド部に入る対象はメモリのアドレスだけではない。
そのため、この後のスライドで幾つか説明する。

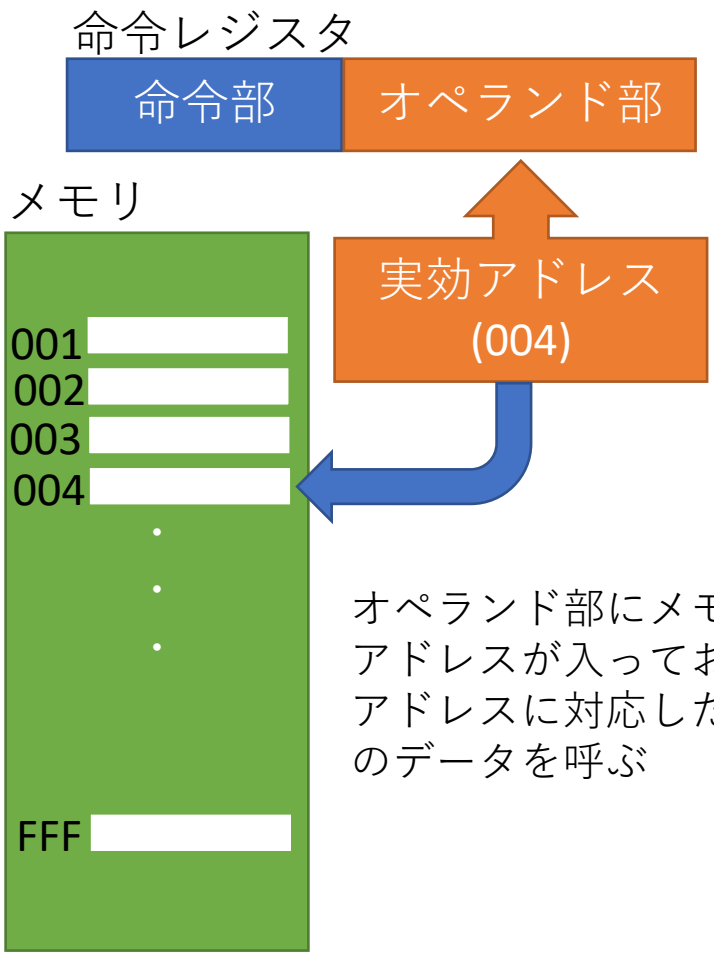
機械語のアドレス指定方式

即値アドレス指定方式



オペランド部に値が入っており、
メモリを参照する必要がない

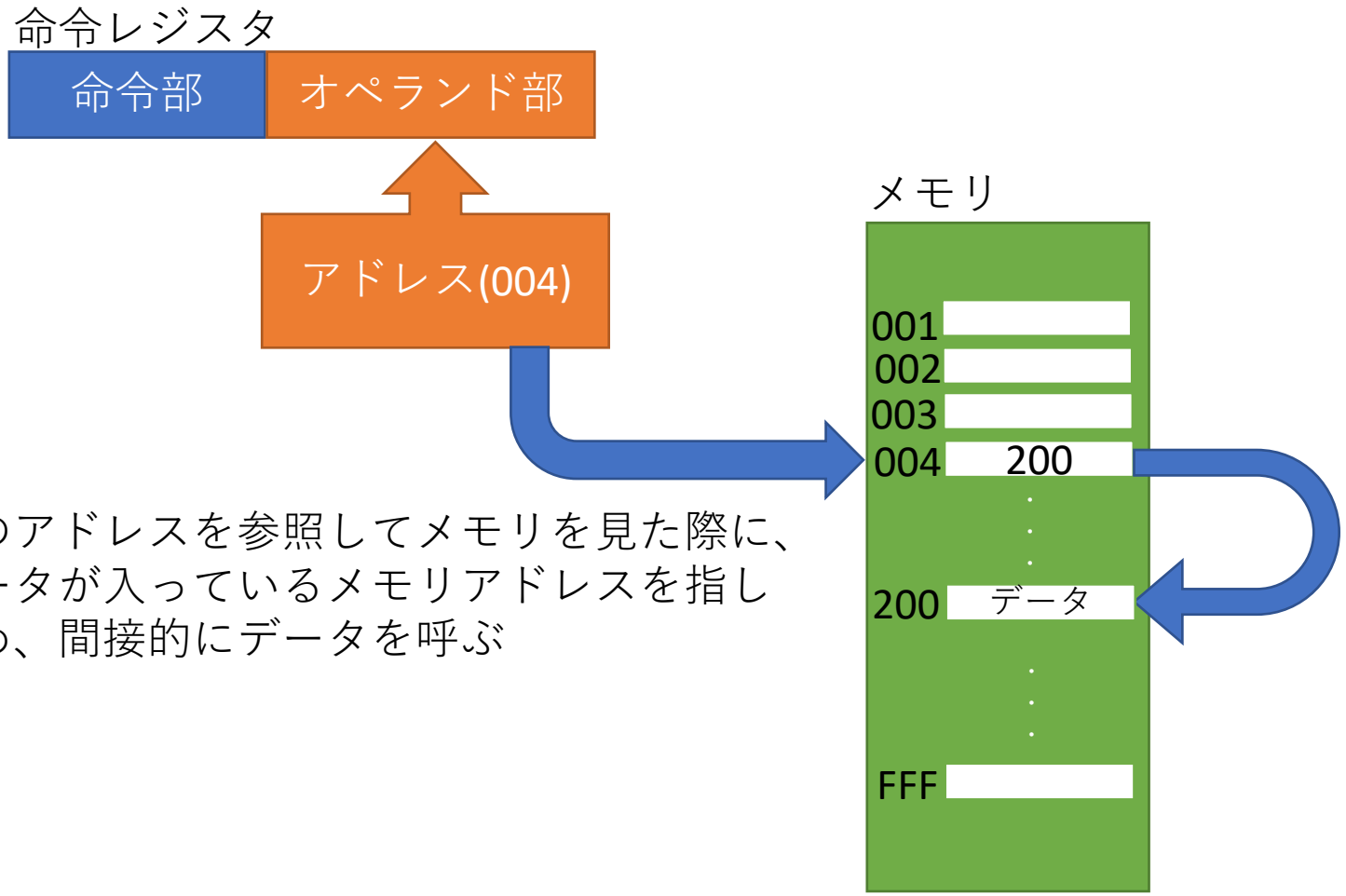
直接アドレス指定方式



オペランド部にメモリの
アドレスが入っており、
アドレスに対応したメモリ
のデータを呼ぶ

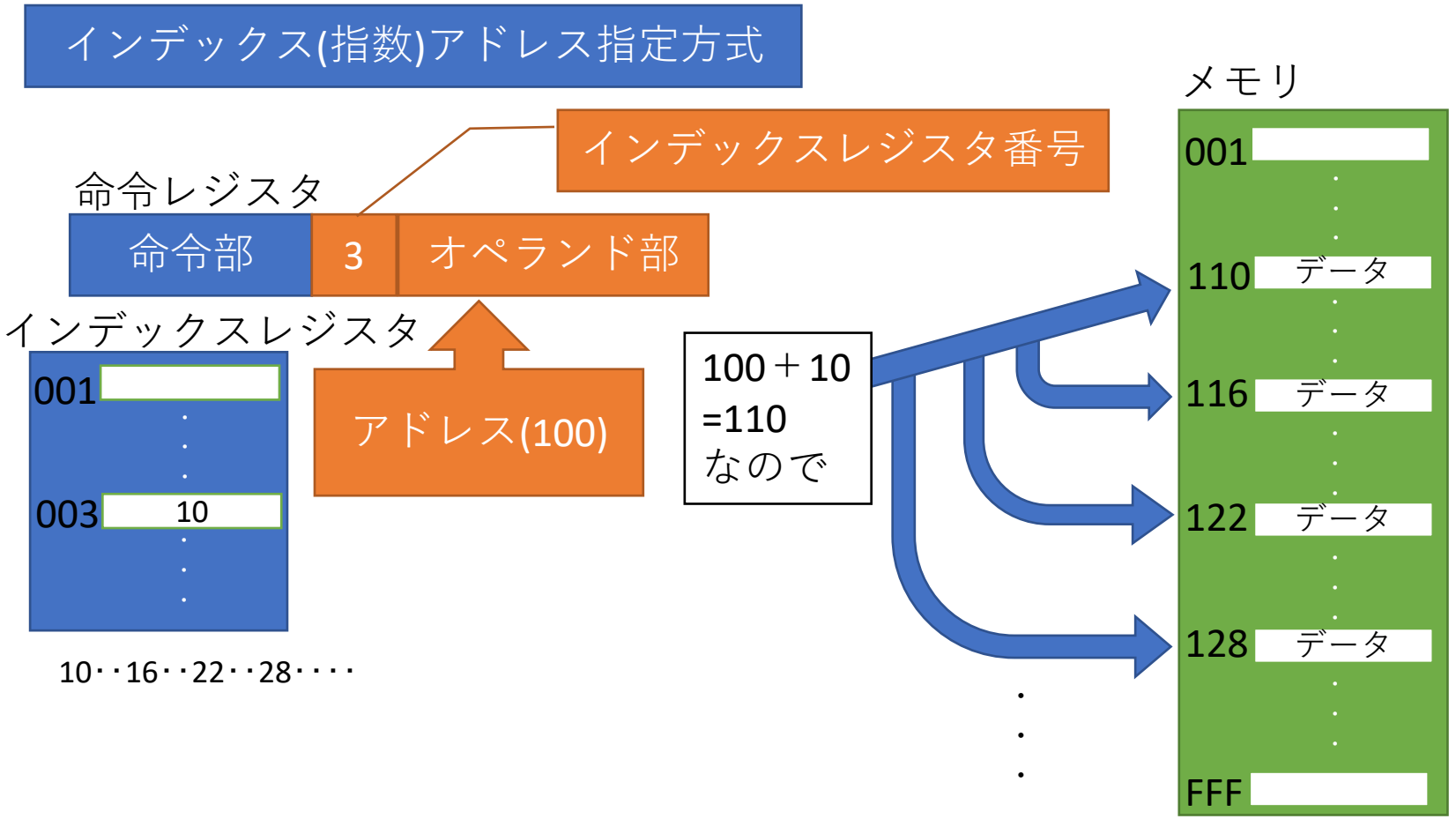
機械語のアドレス指定方式

間接アドレス指定方式



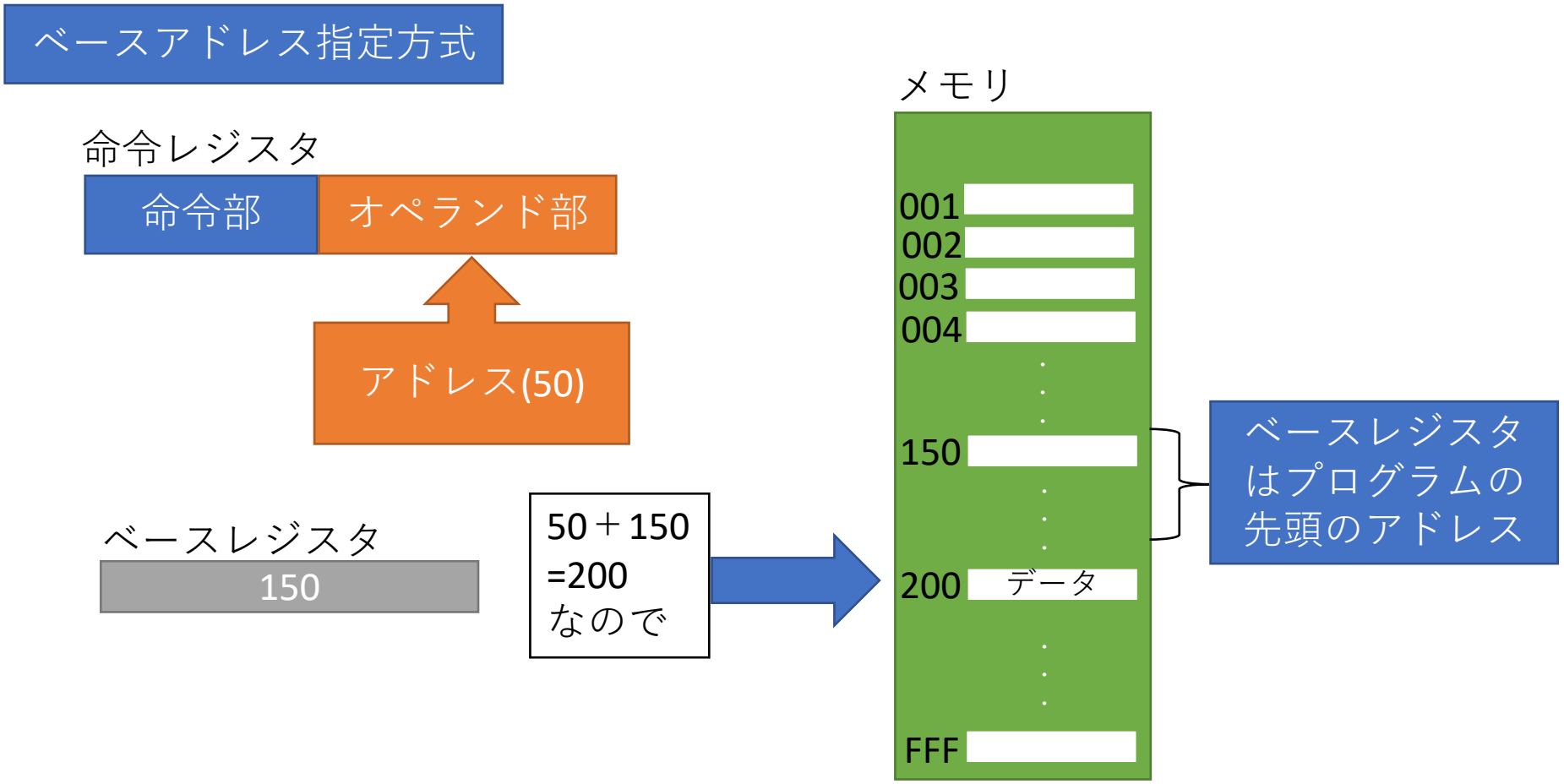
オペランド部のアドレスを参照してメモリを見た際に、対象となるデータが入っているメモリアドレスを指し示しているため、間接的にデータを呼ぶ

機械語のアドレス指定方式



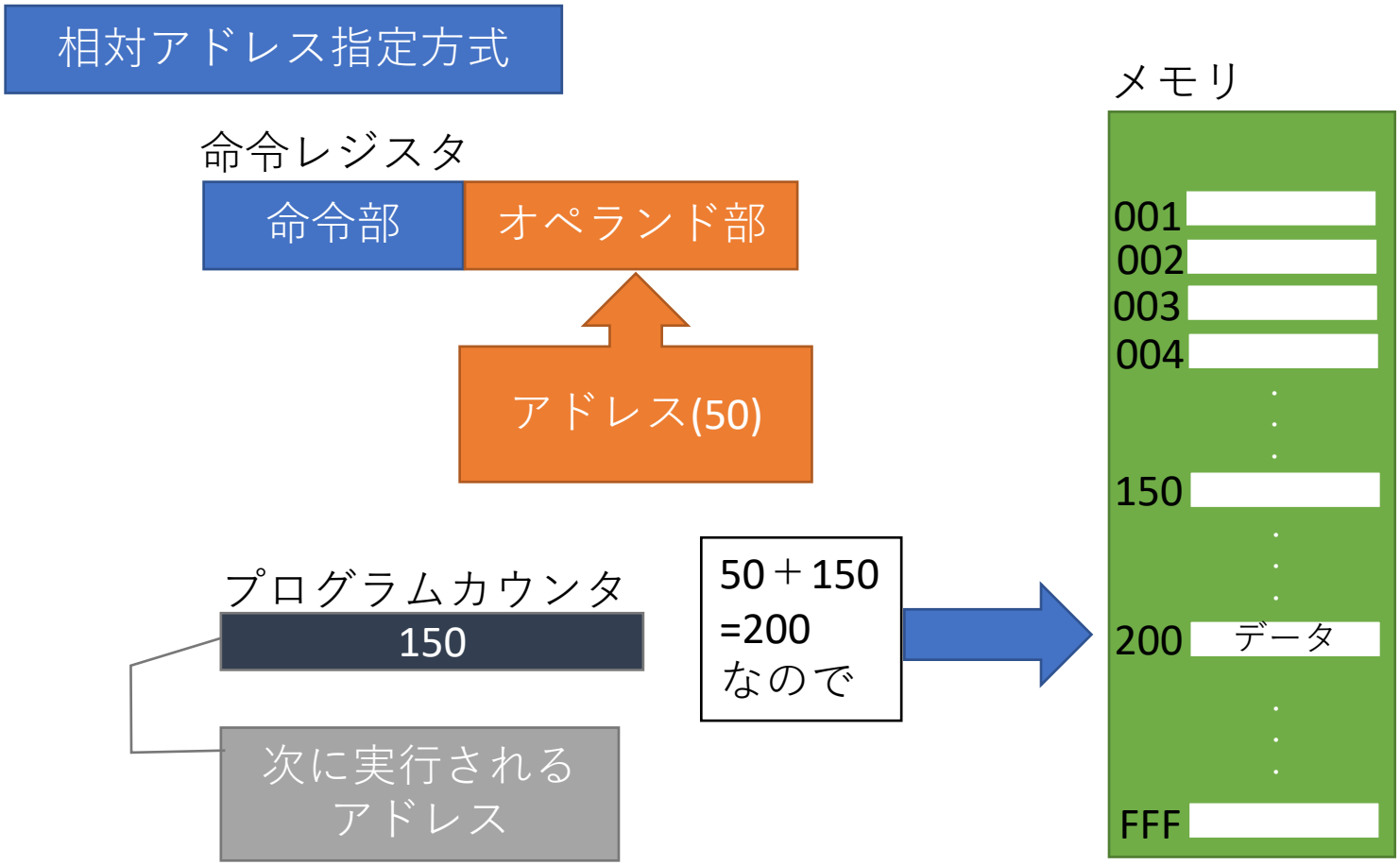
オペランド部のインデックスレジスタ番号を見て、インデックスレジスタを参照する。
インデックスレジスタ内の数字とオペランド部の数字を足したものを、実効アドレスとする。
その後、インデックスレジスタ内の数値を増幅させることで等間隔に並ぶアドレスに同じ命令を繰り返し行うことができる。(配列型のデータ処理で使われる)

機械語のアドレス指定方式



オペランド部に、ベースレジスタの値を加算した数字を実効アドレスとする。
ベースレジスタはメモリ上にプログラムを入れた、ときの先頭部分である。
プログラムがどこにいても命令を変えなくて済む

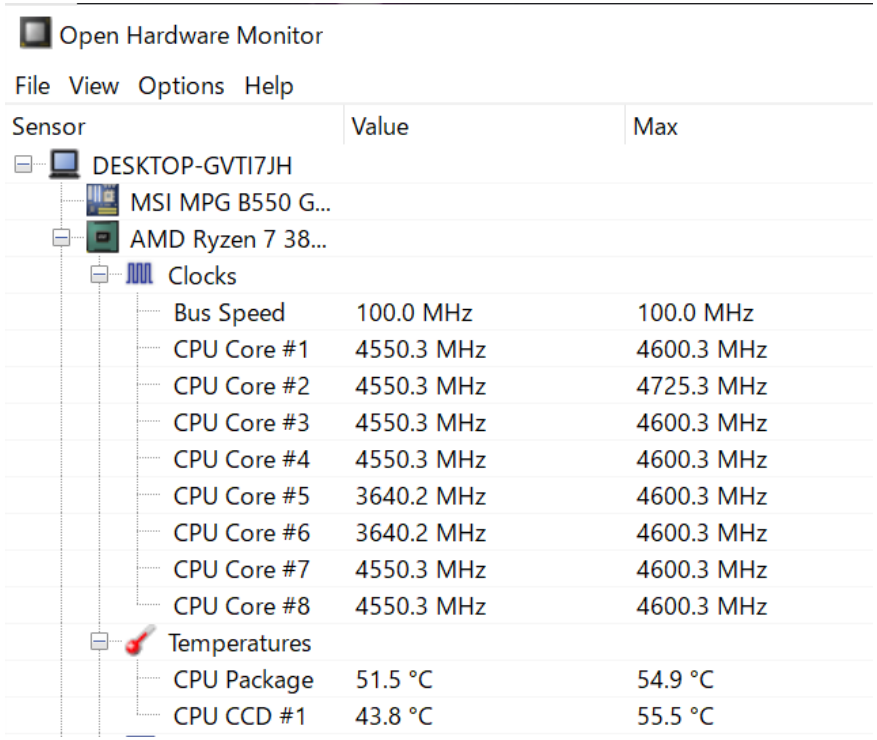
機械語のアドレス指定方式



プログラムカウンタの値とオペランド部の値を加算したものを実効アドレスとする。
プログラムカウンタに入っているのは次に命令レジスタに送られるアドレスである。
ベースアドレス指定方式と同様にどこにロードされても大丈夫

CPUの性能指標

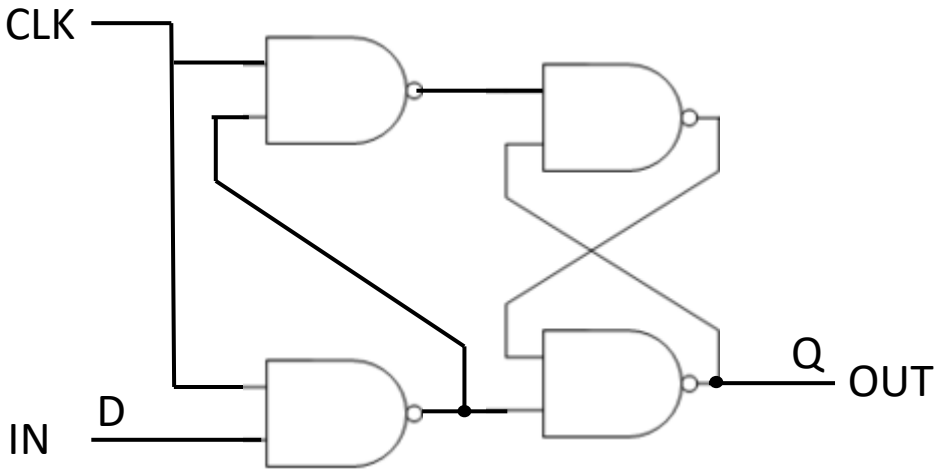
パソコンを購入もしくは作成する際に重要な物の１つであるCPUの性能について、指標をもとに考えてみます。



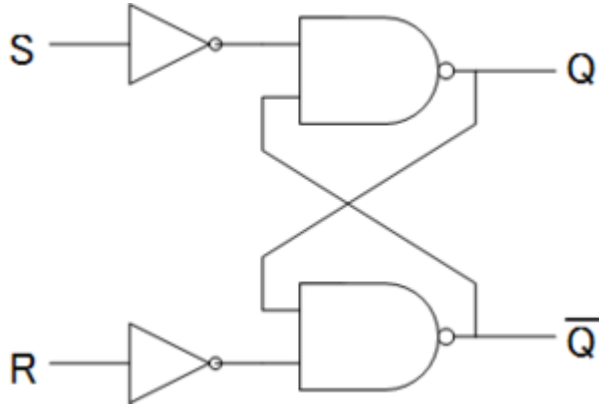
[CPU性能比較表【2022年最新版】| PC自由帳 (pcfreebook.com)]

CPUの性能指標

先に順序理論回路の説明をします。
理論回路で計算やビット反転についての考え方を勉強したと思います。
理論回路を組み合わせたもので過去の入力や初期値によって影響を受ける回路を順序理論回路といいます。



Dラッチ

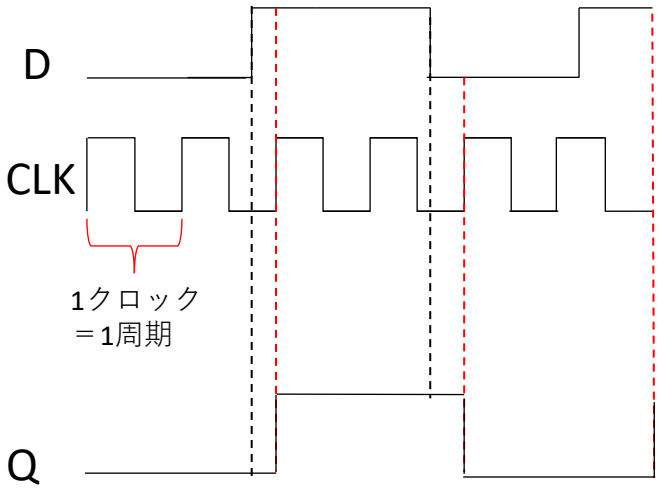


RSフリップフロップ

実はCPUやコンピュータの装置はクロックと呼ばれる周期的な信号に合わせて動くようにできています。
(Dラッチ回路はクロックが立ち上がった時に入力を出力に反映する)

CPUの性能指標

先ほどのDラッチを考えてみましょう



Qに注目するとクロックに合わせて動いている事が分かると思います。
CPUも同じようにクロックに合わせて命令を処理しています。
(人間でいうところの脈みたいですね…)

このクロックが１秒間に繰り返される回数のことをクロック周波数といいます。単位はHz(ヘルツ)
例えば、１GHzのクロック周波数は10^9回チクタク繰り返していることになります。

CPUの性能指標

ところで、1クロックにかかる時間について考えてみましょう。

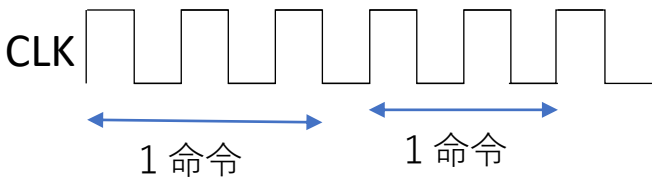
2 Hzは1秒間に2回クロックしたと言い換えられます。

つまり、逆数を取れば所要時間を算出できる事になります。
(最近では3GHz～4GHzのCPUもありますから相当速いことがわかるでしょう)

次にCPUに命令を流した時を考えてみましょう…

CPI

命令が入って来るとCPUはクロックに合わせて動いているので…
1命令に何クロックでうごくのかと考えることができます。
この、何クロックで命令を行うかをCPI(Clock cycles per instruction)といいます



CPUの性能指標

MIPS(Million Instructions Per Second)

1秒間に実行できる命令の数を表したものです。
数字が大きくなりがちなので百万単位であらわしております。

実際にどれくらい早く処理ができるかがこの値によって分かります。
(ただし、アーキテクチャが同じ場合)

実はCPUによって同じような処理をするのにかかるクロック数が異なる。
そのため、異なるアーキテクチャ(後記)のCPUを比較する場合ベンチマークのスコアなどを使う

命令ミックス

命令によって、クロックサイクル数が異なっているので、よく使われる命令をひとつのセットにしたものです。
(MIPSでCPUの性能差を見れるように)

CPUの高速化技術

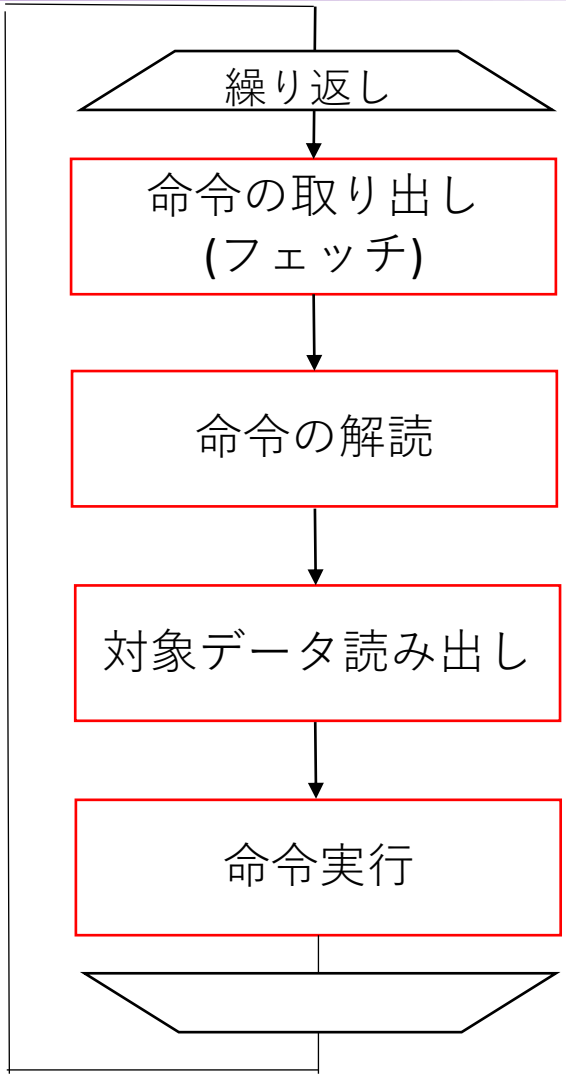
パイプライン処理

プログラムカウンタ→命令レジスタ

命令レジスタ(命令部)→命令レコーダ

命令レジスタ(オペランド部)→汎用レジスタ

汎用レジスタ→演算装置

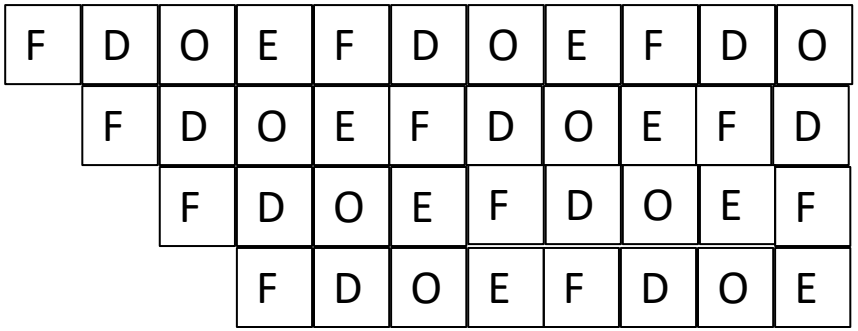


複数の処理をクロックに沿って一つづつ行っていくと、1クロックで
動作していない場所ができるので非効率に感じる

CPUの高速化技術

パイプライン処理

F：命令の取り出し
D：命令の解読
O：対象データ読み出し
E：命令実行

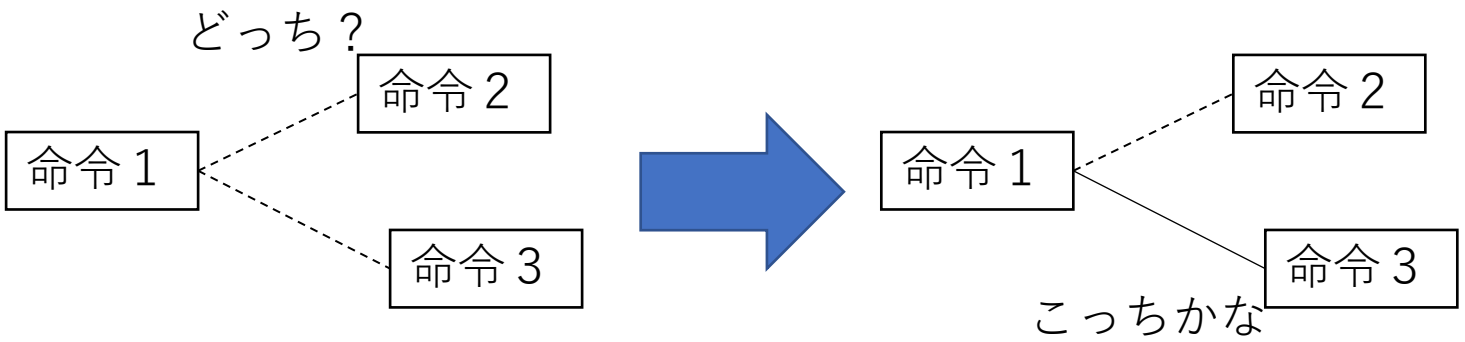


1つずつずらして処理すると並列で上図のように数クロックごとに4つの動作をすることができるこれをパイプライン処理という

CPUの高速化技術

分岐予測と投機事項

先ほどのパイプライン処理の際に命令の分岐があった場合、分岐先の命令がどちらか確定するまで処理が開始出来なくなってしまうこれはパイプライン処理にとっては困ったこととなる。



そのため、待ち時間を発生させないために次の命令を予測しておくことを分岐予測といいます。
その予測に基づいて無駄になってしまうかもわからない状態で分岐先の命令を実行開始する手法が投機実行です。

また、先読みが無駄になってしまうことを(分岐ハザード)といいます。

CPUの高速化技術

CISCとRISC(アーキテクチャ)

アーキテクチャとはCPUの基本設計です。
例としては、intelのx86やAMDのZenアーキテクチャ、組み込み式アーキテクチャのArmなどがあります。
これらのアーキテクチャにはCISCやRISCと言った考え方の違いによって種類分けされている。

CISC

・ひとつの命令で複雑な処理をできる
・命令の長さや実行速度がバラバラ
・機械語のプログラム作成が楽

Intelのx86
AMDのzenアーキテクチャ

パイプライン処理の恩恵を受けにくい

RISC

・単純な命令ののみで構成
・命令の長さがほとんど一緒
・機械語のプログラム作成は難しい

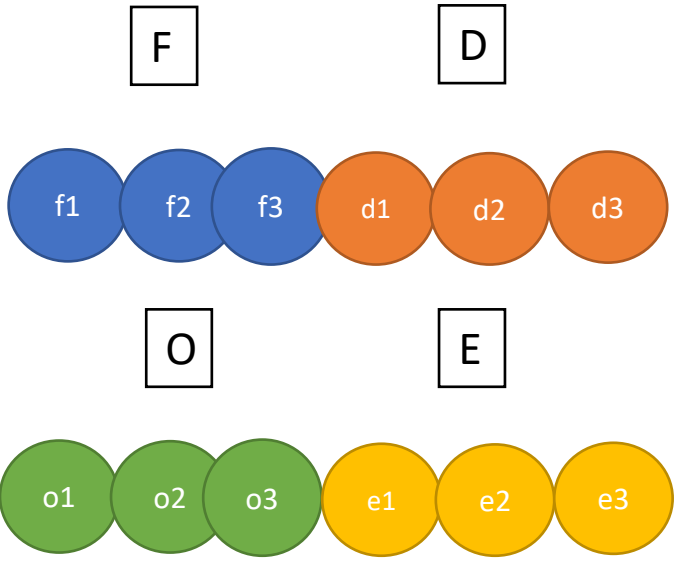
Armアーキテクチャ
(富岳、スマートフォンなど)

パイプライン処理の恩恵を受けやすい

CPUの高速化技術

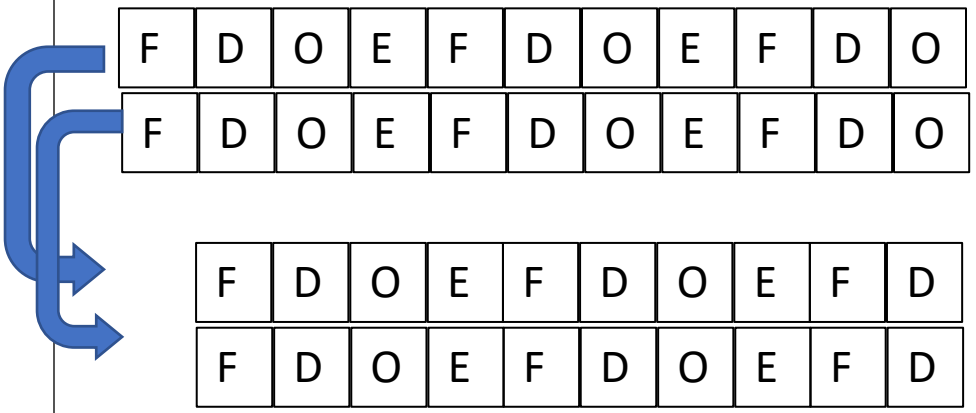
スーパーパイプラインとスーパースカラ

スーパーパイプライン



それぞの命令を細かくすることで
並列で処理できる範囲を増やして、
幾つもの命令を処理する

スーパースカラ



パイプライン処理を行う回路を複数持
たせることで全く同時に複数の命令を
実行できるようにしたもの