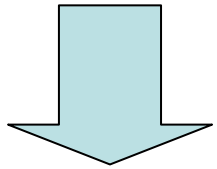
コンピュータ(計算機)

CPUとメモリ、
記憶装置と入出力装置

3章の概要

- 情報の表し方、情報の演算の仕方は1,2章で説明した
- では、情報の演算の指示のやり方は？
  - どのような形で演算を指定するのか？
  - どのような形で演算するデータを指定するのか？



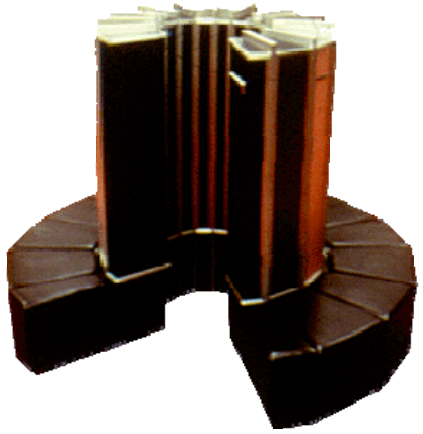
- コンピュータの構成とそれへの演算の指示方法
- コンピュータの性能向上、性能評価、など

節概要

- コンピュータの構成
- 半導体と集積回路
- CPU(Centrai Processing Unit)の構成
  - 命令セット
  - 基本的な構成
  - アドレッシングモード
  - 高速化技術
- メモリアーキテクチャ
- 入出力装置
  - 補助記憶装置(磁気ディスク、光ディスク)
  - 入出力インタフェース
  - 代表的な入出力装置

コンピュータの種類と特徴

- スーパーコンピュータ
  - 大規模な科学計算技術計算用に設計
  - 最も高速・高性能なコンピュータ
- 汎用コンピュータ(メインフレーム)
  - 事務処理から技術計算までの多目的に利用できるよう設計された大型コンピュータ
  - 高い耐故障性や故障時の代替処理の迅速化を考えて設計
- ワークステーション
  - 高度な処理能力が求められる技術分野やネットワークサーバ分野で利用



Cray-1

ラックマウント型ワークステーション

コンピュータの種類と特徴

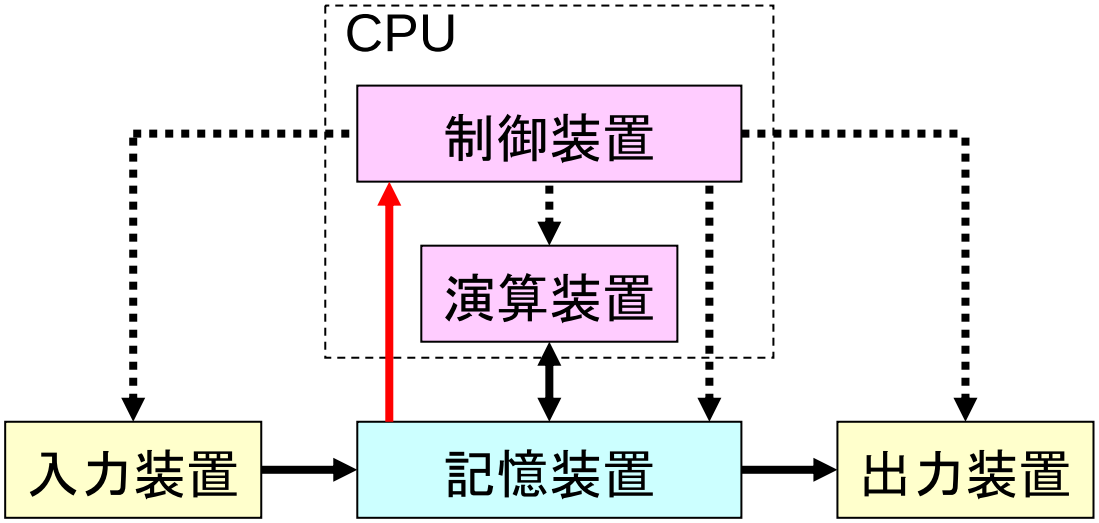
- パーソナルコンピュータ(パソコン)
  - 家庭やオフィスなどで多目的に利用されるコンピュータ
- 携帯情報端末(PDA: Personal Digital Assistant)
  - ノートパソコンよりも小型で携帯して持ち運ぶことが可能なコンピュータ
  - スマートフォンによって復権
- マイクロコンピュータ(マイコン)
  - 1つのチップに納められたコンピュータ
  - 家電製品などに組み込んで利用される
  - マイコンはマイクロコントローラの略にも

PDA(スマートフォン)

マイコン

3.1 コンピュータの構成

5大装置(機能): 入力、出力、記憶、制御、演算



- 実線はデータの流れ
- 破線は制御の流れ
- 赤線は命令の流れ

(1) コンピュータの構成

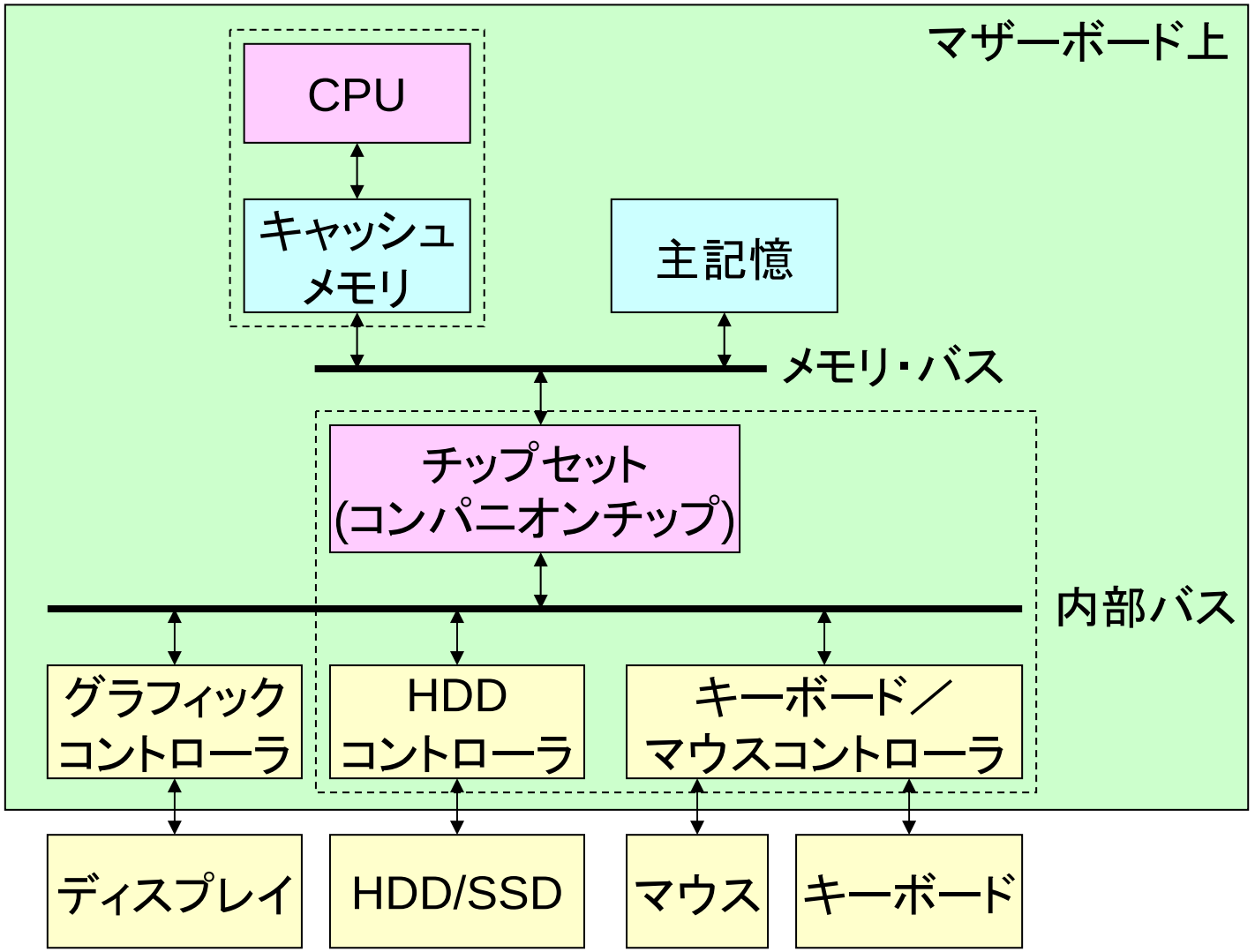
- 入力装置
  - データやプログラムを呼び込む
  - キーボード、マウス、タブレット
- 記憶装置
  - データやプログラムを記憶する。
  - 記憶装置には、主記憶装置と補助記憶装置がある。
    - 主記憶装置：コンピュータ内部にありCPUで実行するプログラムやデータを格納する。揮発性
      - DDR(2|3)-SDRAM
    - 補助記憶装置：実行前／後のプログラムやデータを格納。大容量低価格、不揮発性。
      - CD-ROM、DVD-ROM、BD-ROM、SSD、HDD、磁気テープ

(1) コンピュータの構成

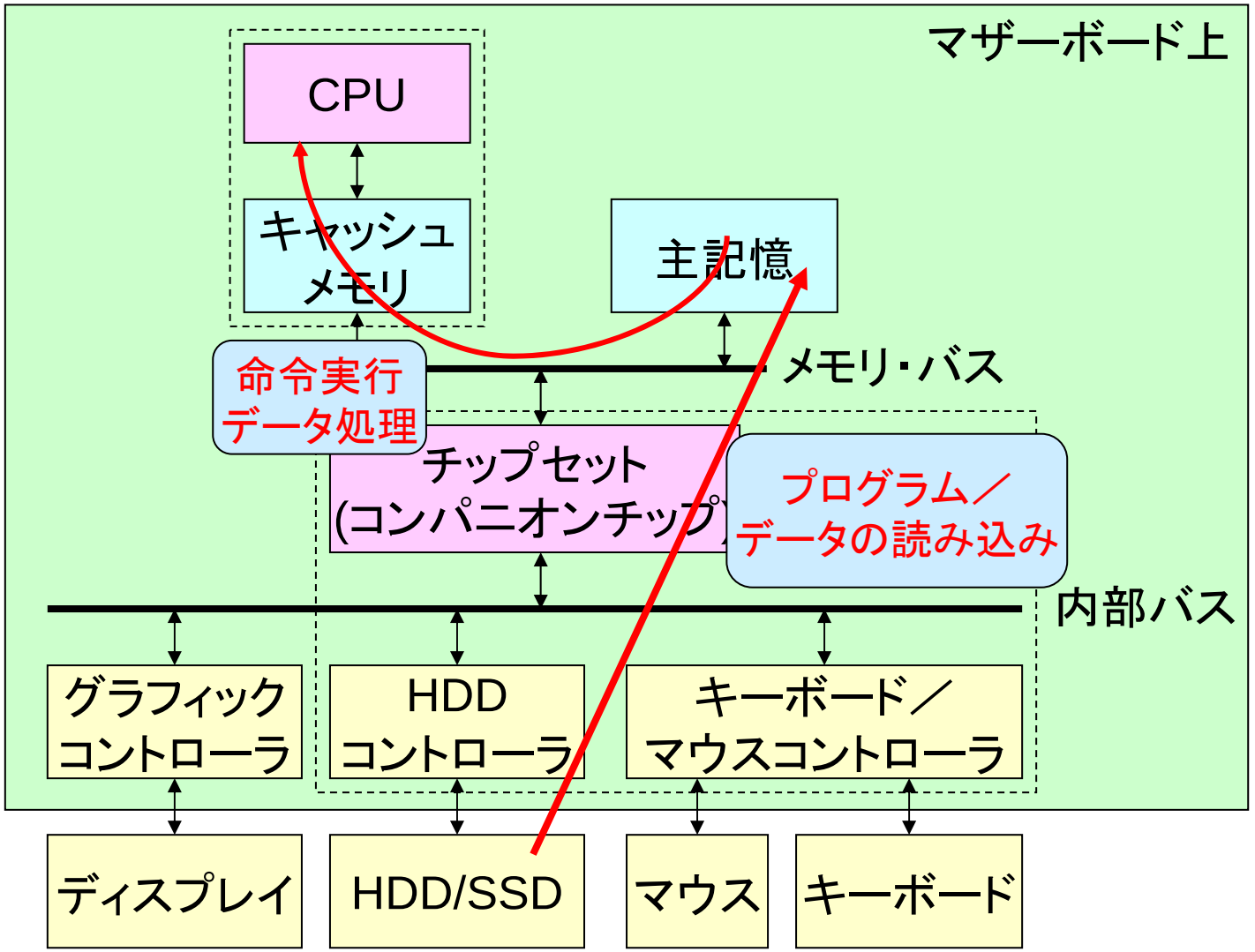
- 出力装置
  - 処理結果のデータを外部に取り出す
  - 例: ディスプレイ、プリンタ、スピーカー
- 演算装置
  - 計算、比較、判断などを行う
- 制御装置
  - 入力装置、記憶装置、出力装置、演算装置の制御を行う

CPU(Central Processing Unit): 演算装置と制御装置からなる

(2) コンピュータの基本的な構成



(2) コンピュータの基本的な構成



3.2 半導体素子(トランジスタ)とメモリ

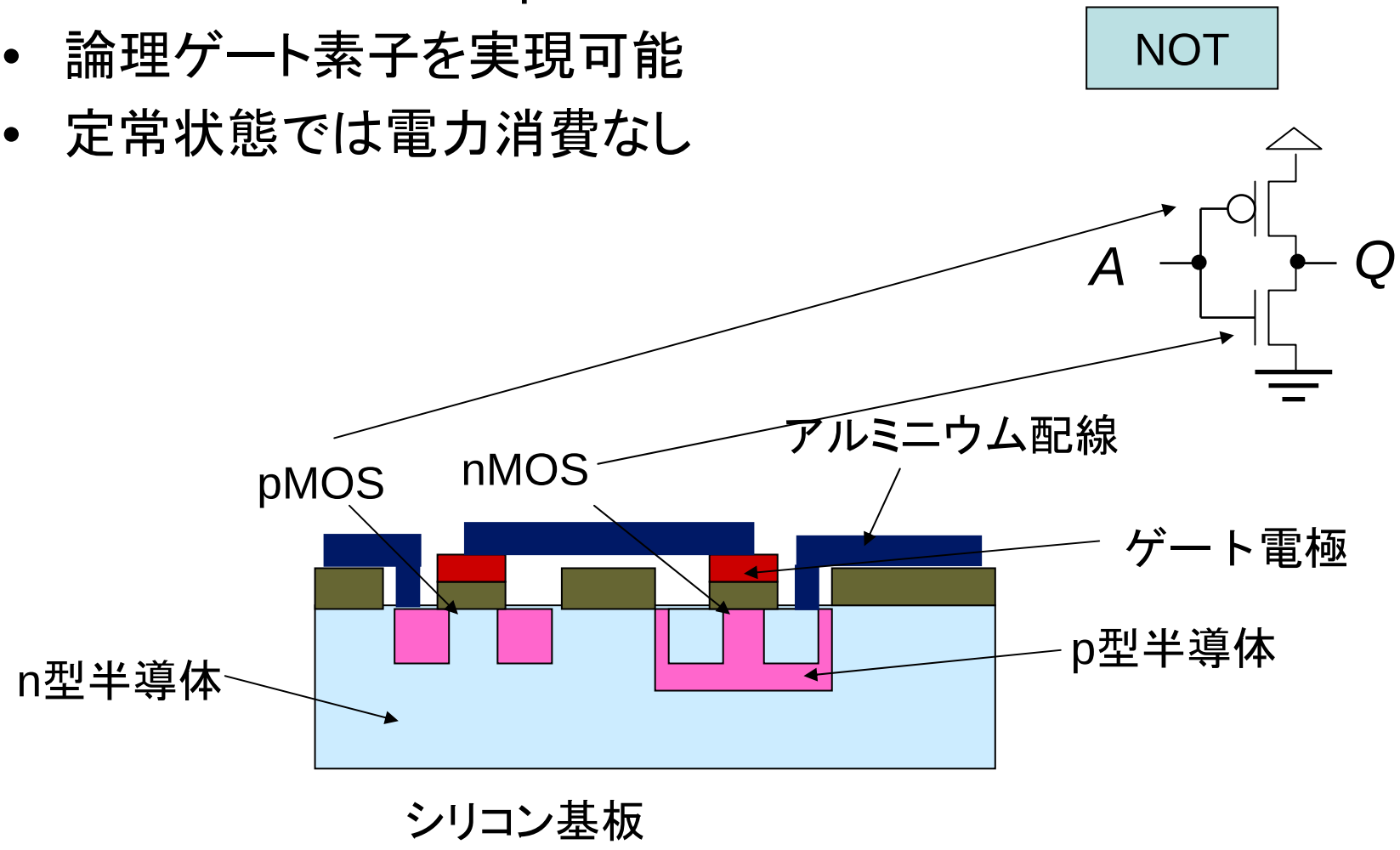
- 近年のコンピュータは半導体素子の中でもトランジスタを利用
  - かつては真空管とかリレーとか
- さらに、トランジスタを1つのシリコンチップに集積した集積回路(IC: Integrated Circuit)を利用

(1) 集積回路

- バイポーラ型
  - バイポーラ型トランジスタを組み合わせた集積回路
  - 回路の動作が高速
  - 消費電力が大きく、発熱量が多くなる
  - 90年代中盤に集積回路の主流から外れたが、強い電流を出力できるという利点があるので、使われている分野もある
- CMOS(Complementary Metal Oxide Semiconductor)型
  - 2種類のMOSFETを組み合わせた集積回路
  - 消費電力が少ない
  - バイポーラ型よりも集積できるので安価に製造可能
  - 現在のコンピュータの多くはCMOS型集積回路によって構成

シリコンチップ上のCMOS

- CMOSとはnMOSとpMOSの相補構成
- 論理ゲート素子を実現可能
- 定常状態では電力消費なし



(2) ICメモリ

- 大きくROMとRAMに分けられる
  - ROM(Read Only memory): 不揮発性
    - 電源を切ってもデータは残る
  - RAM(Random Access Memory): 揮発性
    - 電源を切ったらデータは消失する
- 上記のRAMとROMの両方の性能を満たすことを目的とした物も開発中
  - Magnetic RAM
  - Phase Change RAM
  - Registive RAM

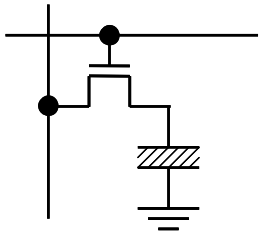
ROM(Read Only Memory)

- マスクROM
  - 製造時に記憶内容を書き込み、書き換えることはできない
- PROM(Programmable ROM)
  - 一度だけデータを書き込めるROM。書き換えることはできない
- EPROM(Erasable PROM)
  - 記憶内容を紫外線で一括消去して、内容を書き換えることができる
- EEPROM(Electrically EPROM)
  - 電気的にブロックまたはバイト単位で記憶内容を消去し、データを書き換えることができる
- フラッシュメモリ
  - 電気的に一括またはブロック単位でデータを消去して内容を書き換えることができる
  - SDカードなどの記録メディアに主に使われる
  - 近年では、SSDという形でHDDの代替に使われることもある

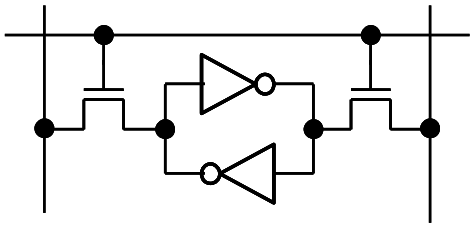
RAM(Random Access Memory)

- DRAM(Dynamic RAM)
  - コンデンサを使用してデータを記憶する
  - コンデンサ中の電荷は漏れるため、一定時間ごとに再書き込み（リフレッシュ）を行い、記憶内容を保持
  - 集積化しやすく、低コスト化、大容量化に対応

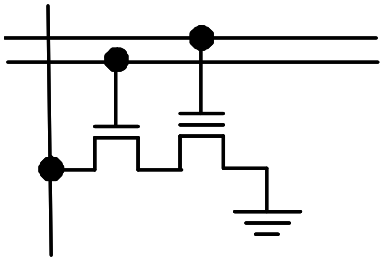
  ->主記憶装置に用いられる。
- SRAM(Static RAM)
  - フリップフロップ回路を用いてデータを記憶する
  - DRAMに比べて高速
  - キャッシュメモリに用いられる



DRAM



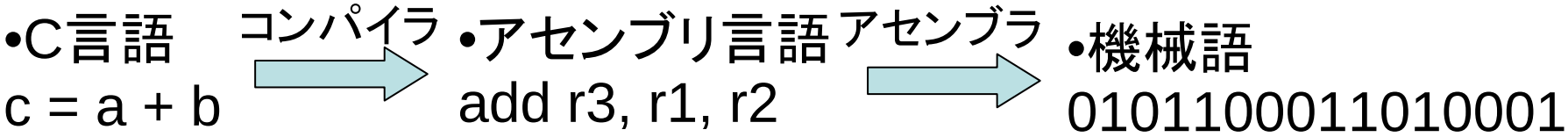
SRAM



フラッシュメモリ

3.3 プロセッサアーキテクチャ

- 機械語
  - 0/1の羅列 ->人間にはまずちんぷんかんぷん
  - CPUの命令セットが違えば異なる
- アセンブリ言語
  - 機械語と1対1に対応する表記
  - 人間にはだいぶ分かりやすい
  - アセンブラで機械語に変換可能
- 高級言語(例：C言語)
  - 人間に分かりやすい言語
  - コンパイラでアセンブリ言語に変換可能

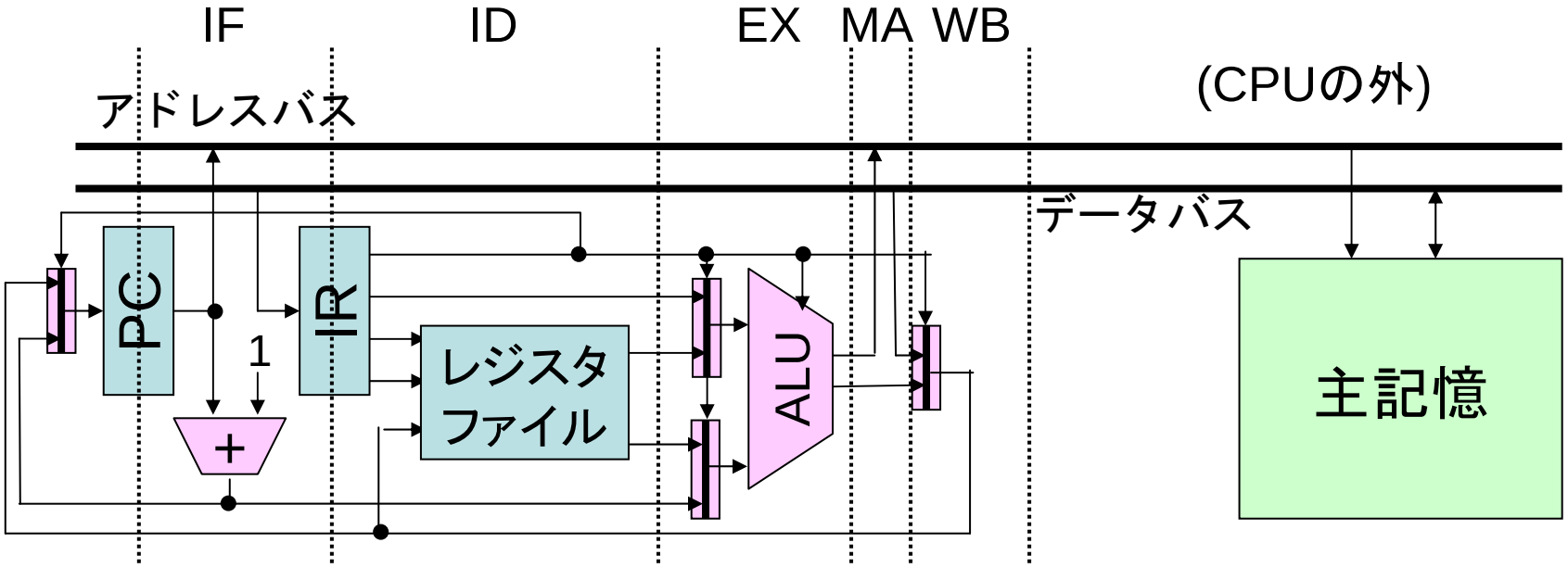


CPUに関するアーキテクチャ

- アーキテクチャ: 構造、建築術
- CPUに関するアーキテクチャ
  - 命令セットアーキテクチャ: 機械語の構造
    - ISA: Instruction Set Architecture
  - マイクロアーキテクチャ: 回路の構造
  - ロード／ストアアーキテクチャ: 主記憶の内容は一度レジスタに読み込まないと演算できない構造
- ISAが同じでマイクロアーキテクチャが違うものは多数ある
  - Intel社Core-i7とAMD社Phenom
  - ARM-v7 ISAのCPU(携帯電話等で多用)は非常に多数ある

(2) CPUにおける命令の実行

- 以下の5段階(フェーズ)で命令を実行
  - 命令フェッチ(IF: Instruction Fetch)
  - 命令デコード(ID: Instruction Decode)
  - 実行(EX: EXecution)
  - メモリ(主記憶)アクセス(MA: Memory Access)
  - ライトバック(WB: Write Back):

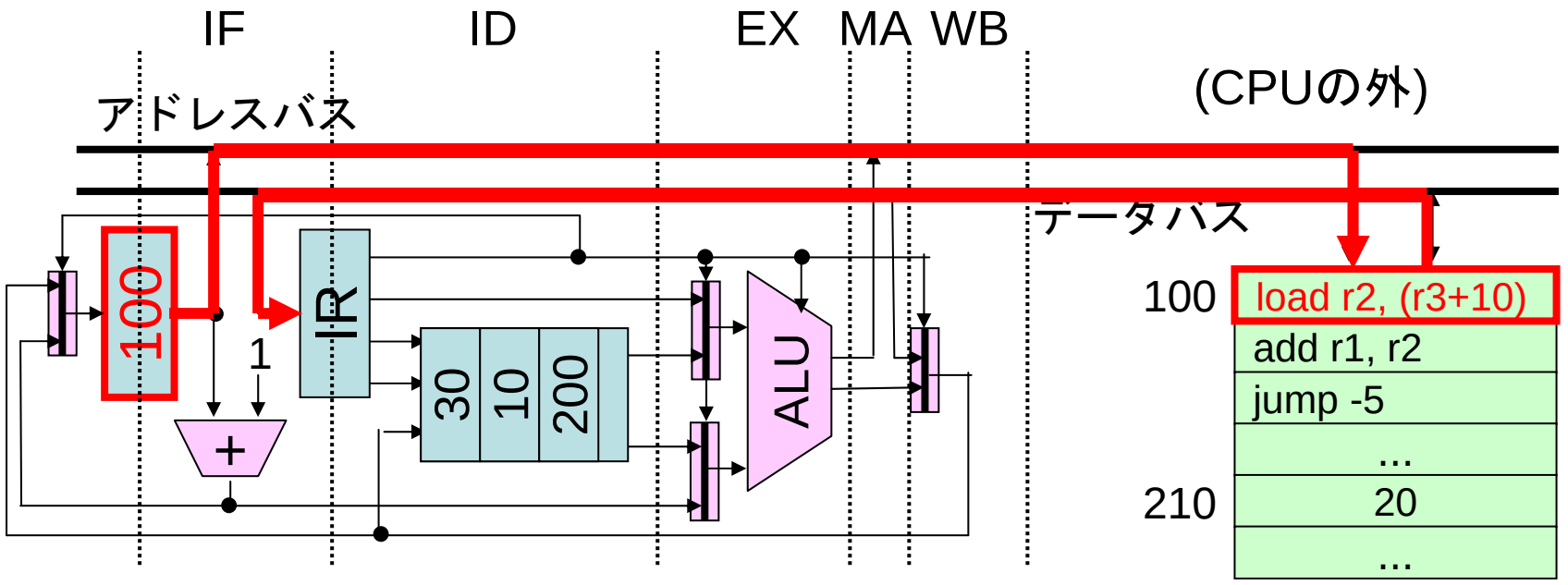


3つの命令による命令実行の例示

- メモリアクセス命令(ロード／ストア命令)
  - メモリからレジスタに値を読み込むロード命令の例示
  - load r2, (r3+10)
    - レジスタ2番に、メモリ上の(レジスタ3番の値＋10)番地の値を読み込む
- 演算命令
  - レジスタ値同士を加算する加算命令の例示
  - add r1, r2
    - レジスタ1番とレジスタ2番の値を加算し、結果をレジスタ1番に格納
- 制御命令(分岐命令)
  - 無条件に分岐するジャンプ命令の例示
  - jump -5
    - PCを-5し、5つ前の命令の実行に戻る

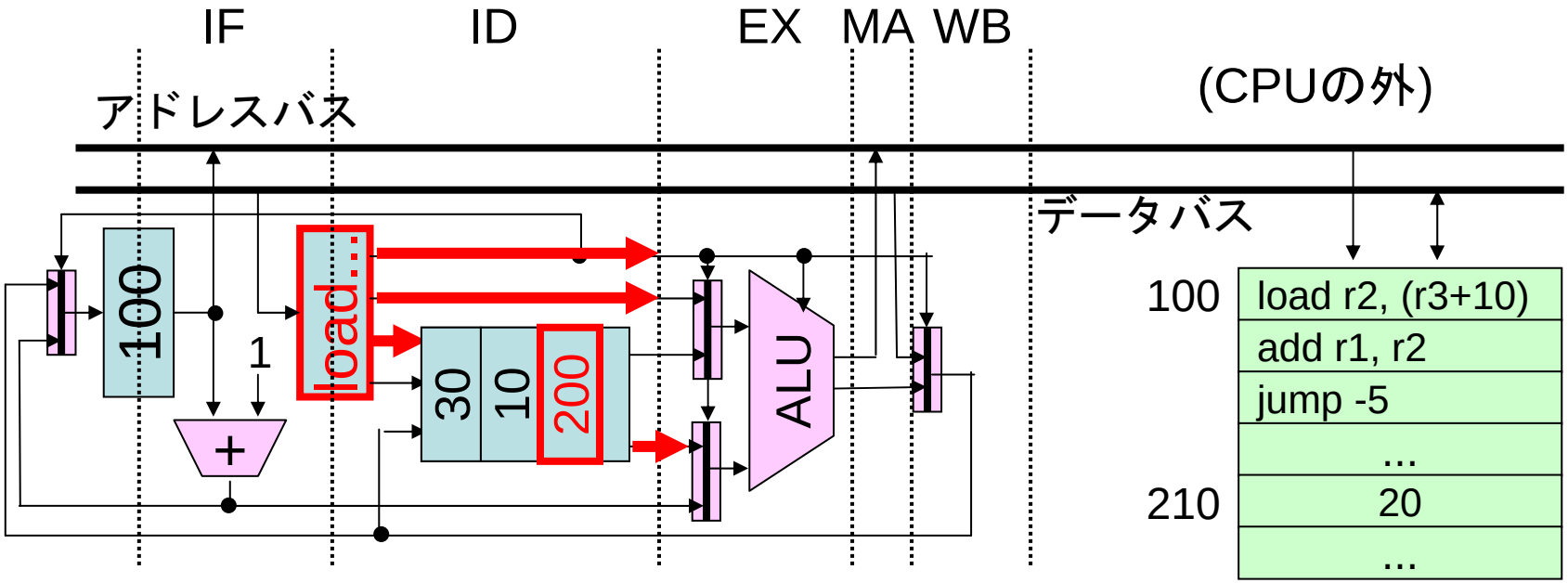
ロード命令の実行(IF)

- IF: Instruction Fetch
- PC(Program Counter)の値を主記憶に送り、対応する番地の命令語を読み出す
- 命令語はIR(Instruction Register)に保存



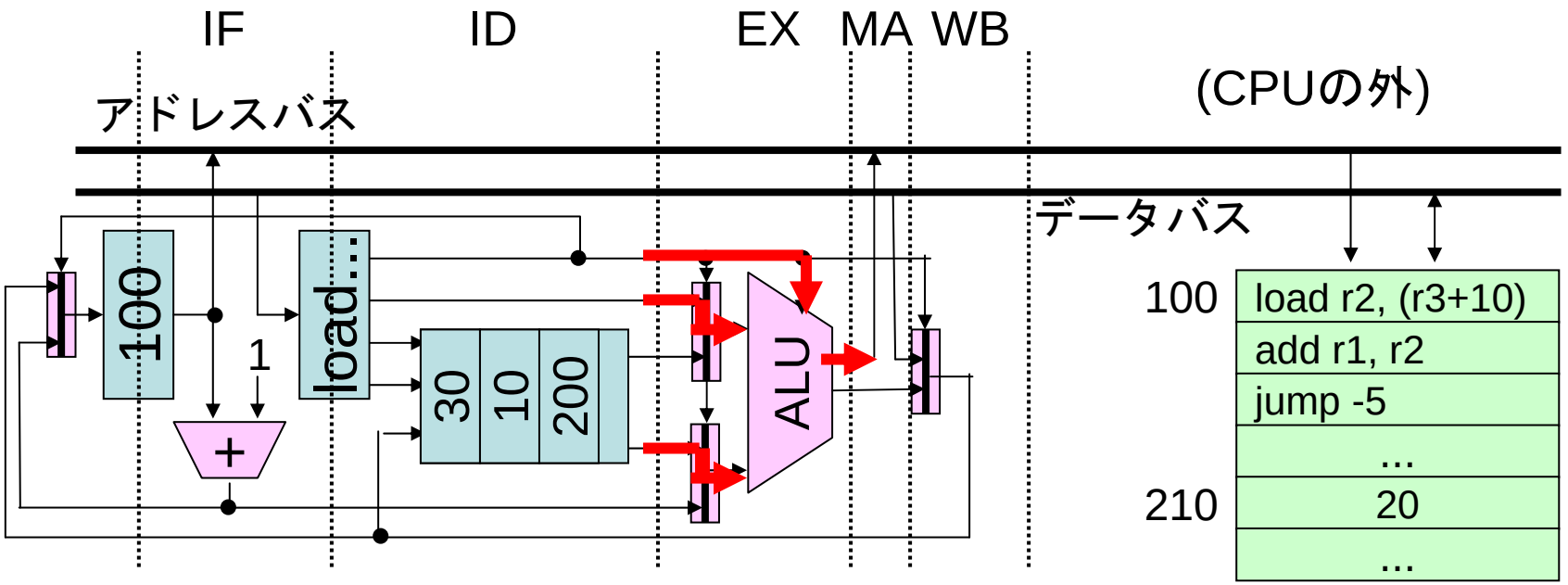
ロード命令の実行(ID)

- ID: Instruction Decode
- レジスタファイルからレジスタ値r3の読み出し
- 命令語から即値10の切り出し
- 命令語から演算の内容等を切り出し



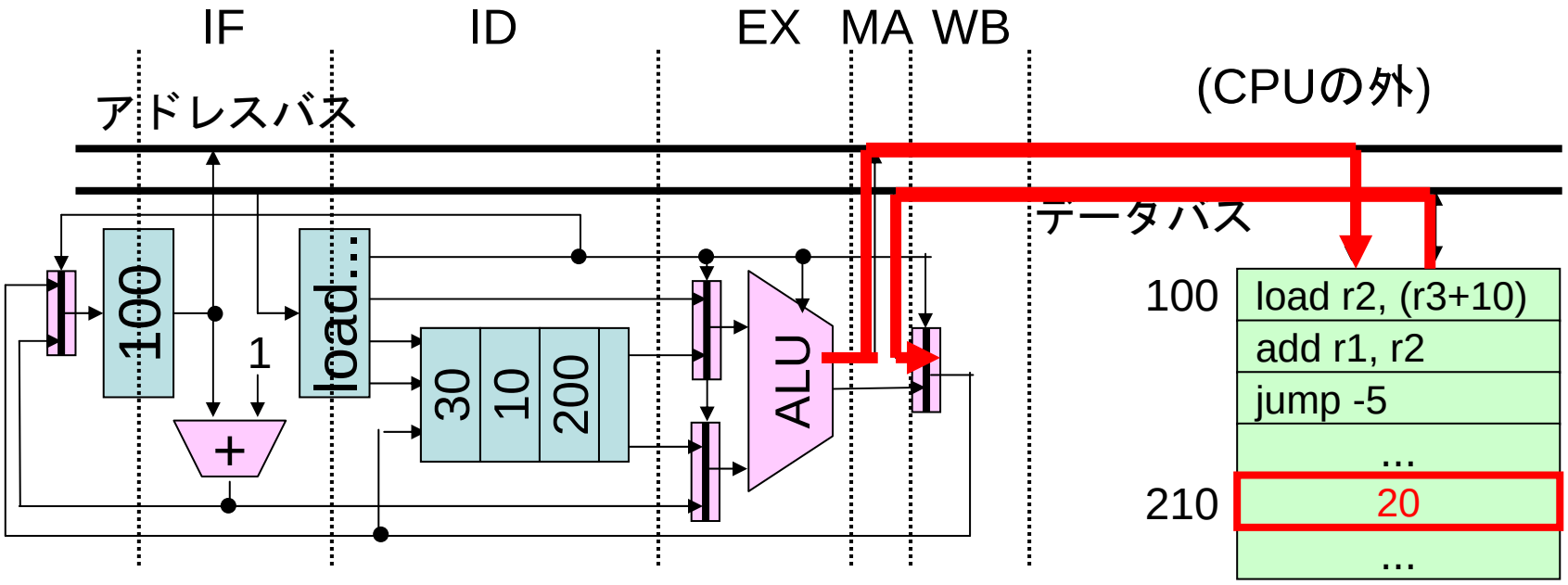
ロード命令の実行(EX)

- EX: EXecution (EXEとも略す)
- ALU(Arithmetic Logic Unit)で実効アドレスを生成
  - レジスタr3の200と即値の10を加算



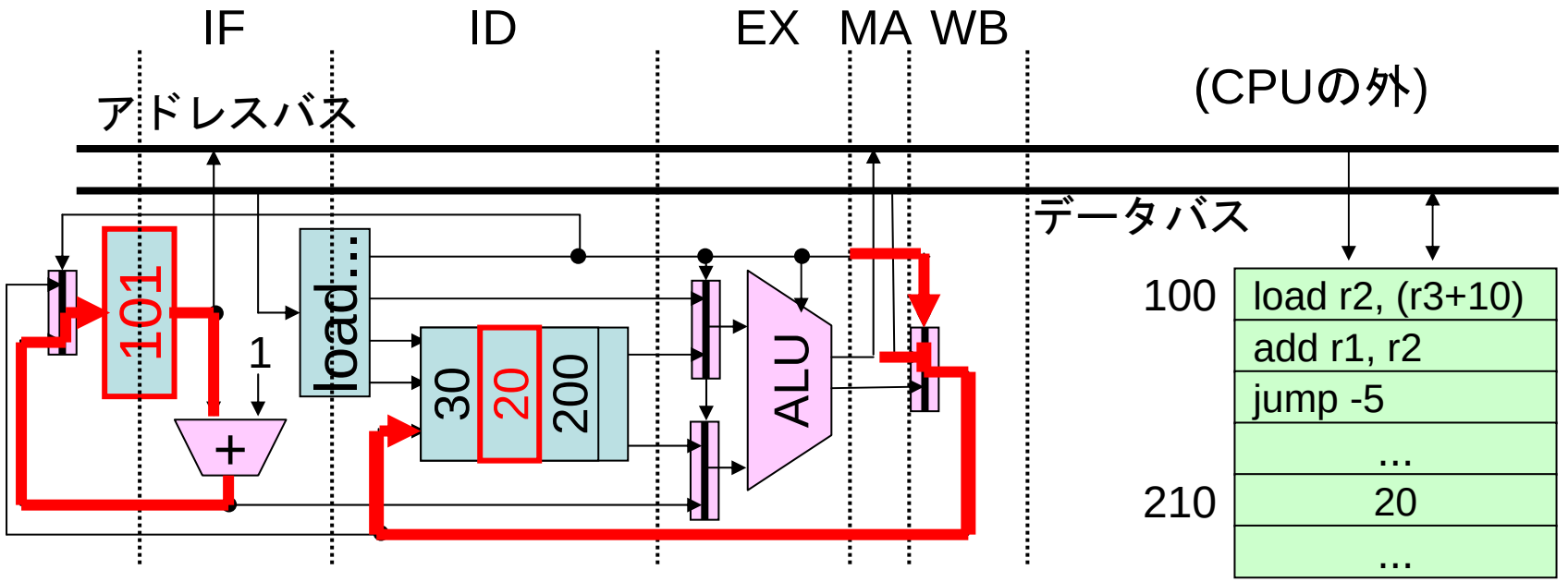
ロード命令の実行(MA)

- MA: Memory Access (MEMとも略す)
- 計算した実効アドレスを主記憶に送り、データを読み出す



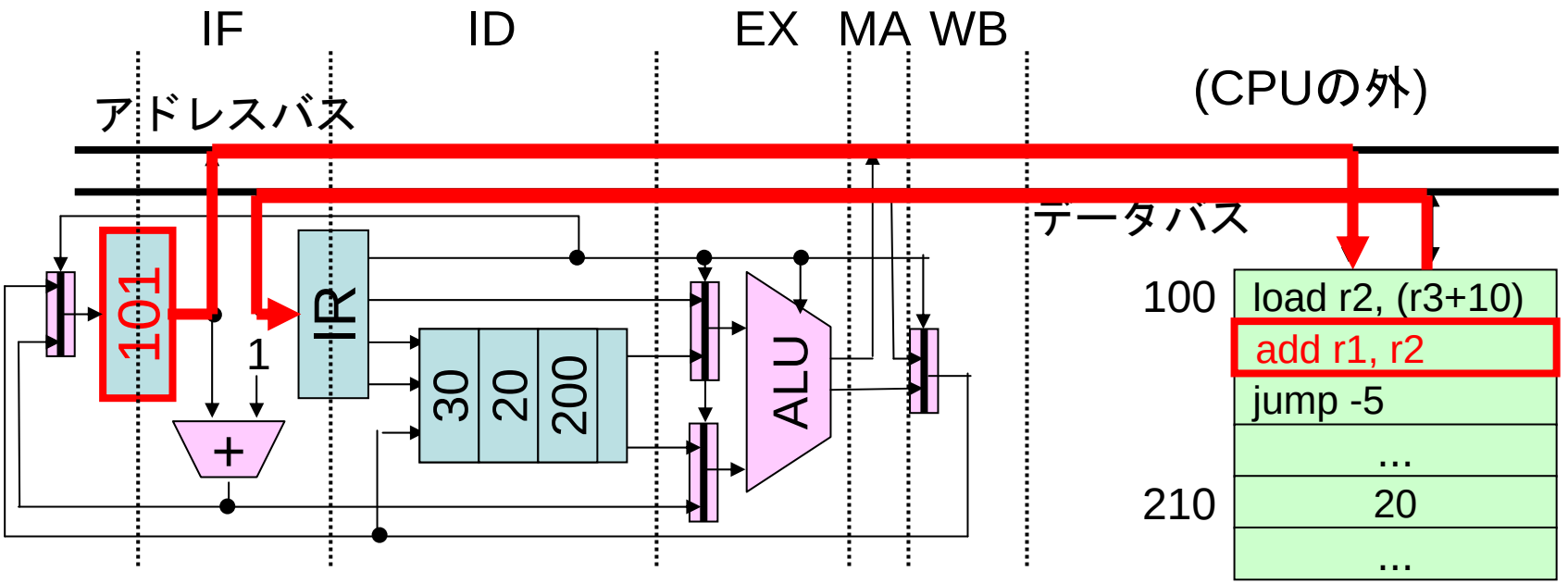
ロード命令の実行(WB)

- WB: Write Back
- ロードした結果をレジスタr2として保存
- 次の命令の実行に備え、PCを+1して次の命令を指し示す
  - IFの直後に+1するマイクロアーキテクチャも多い



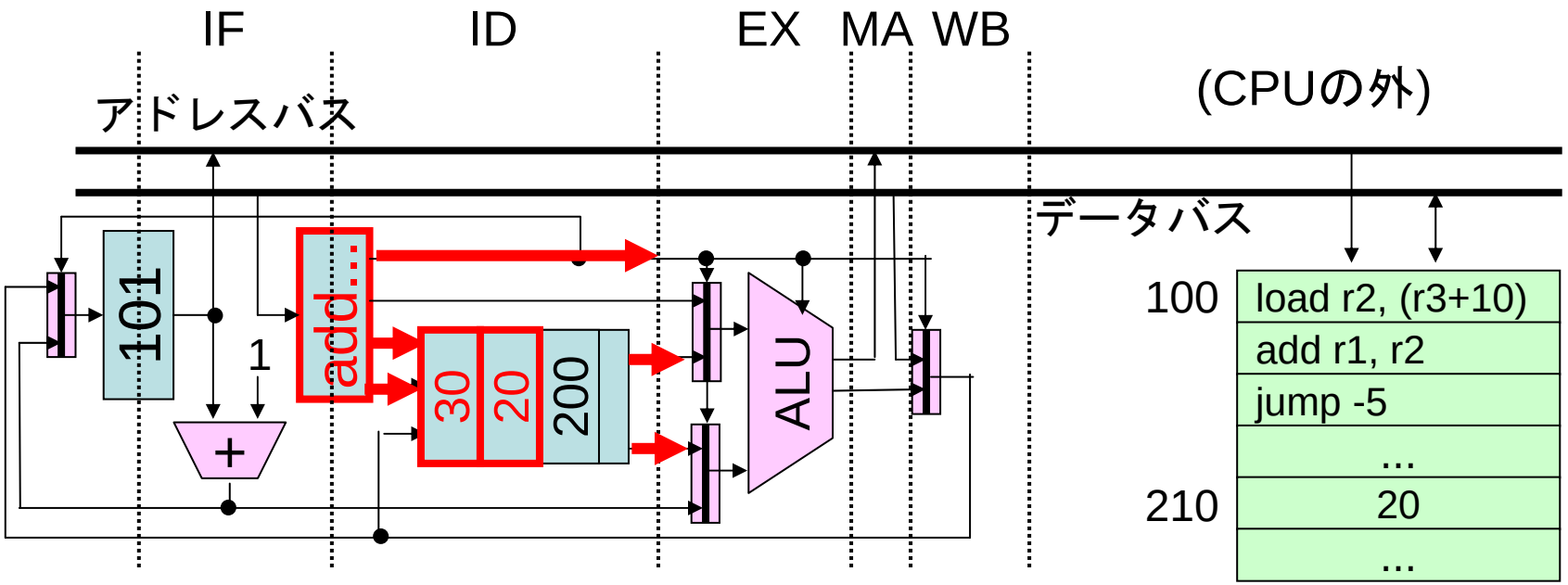
演算命令の実行(IF)

- ロード命令と変わらず



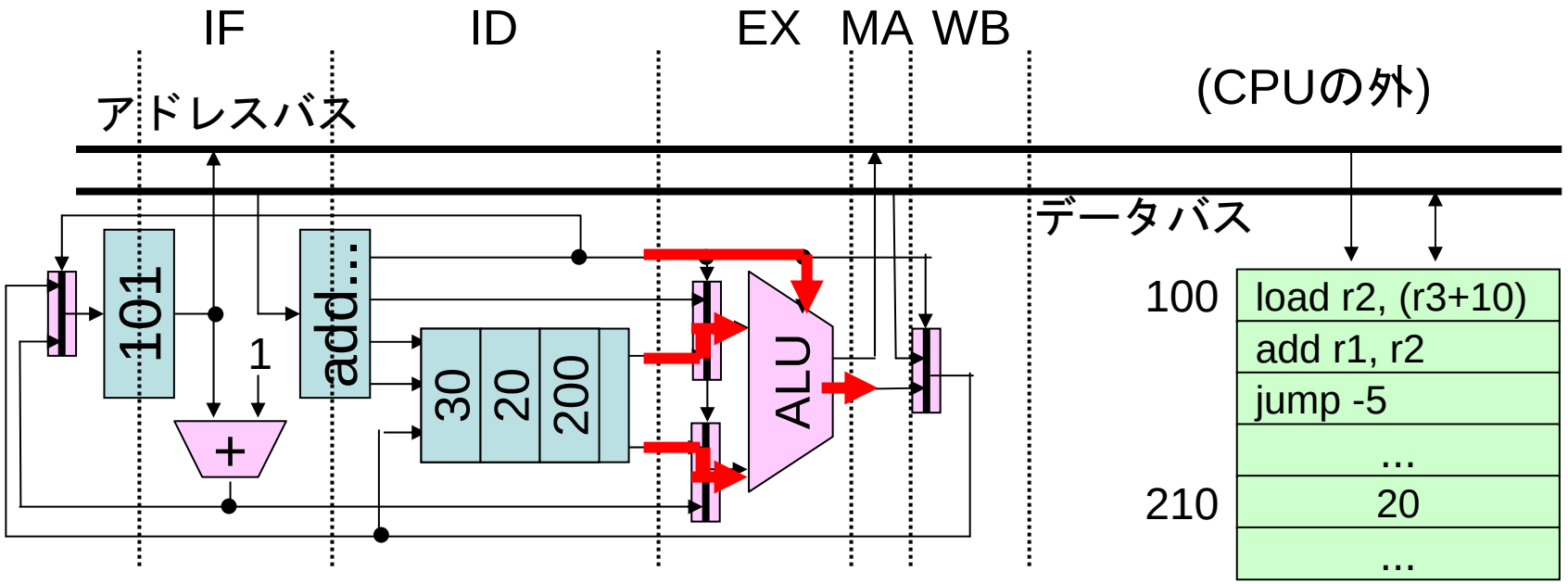
演算命令の実行(ID)

- レジスタファイルから2個のレジスタ値(r1, r2)を読み出す
- 命令語から演算の内容等を切り出し



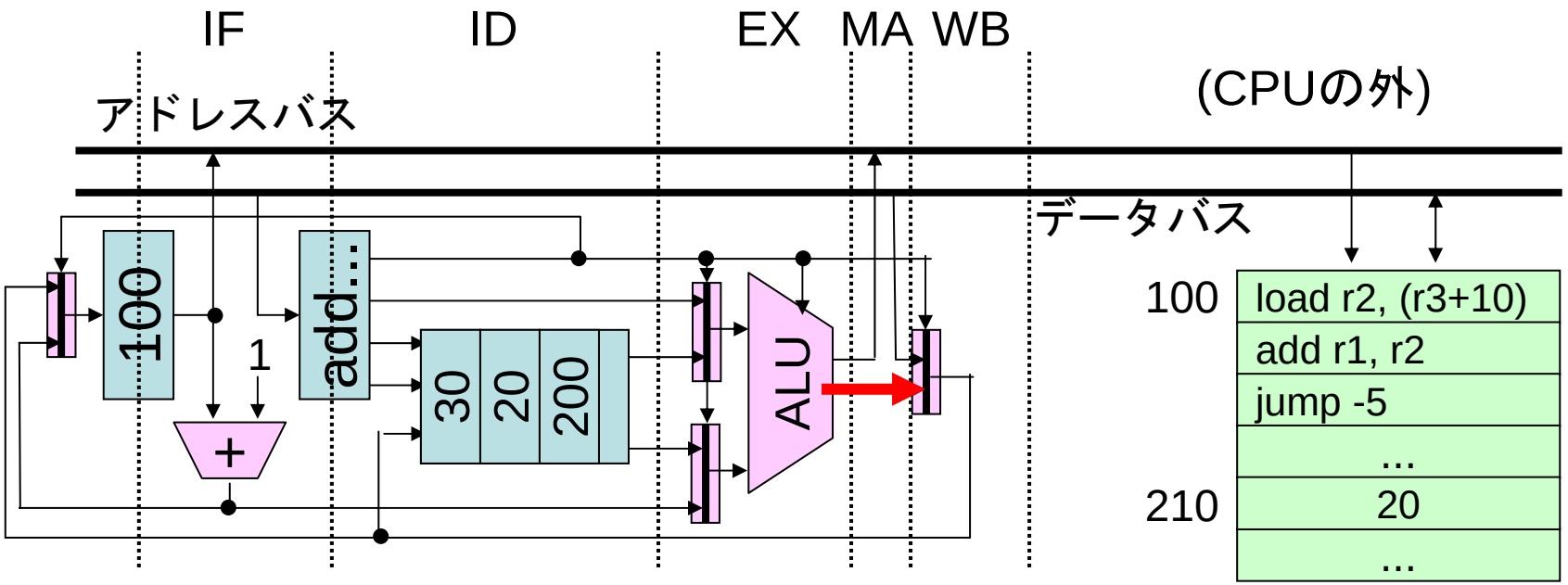
演算命令の実行(EX)

- ALUで2つのレジスタ値を加算
  - レジスタr1の30とレジスタr2の20を加算



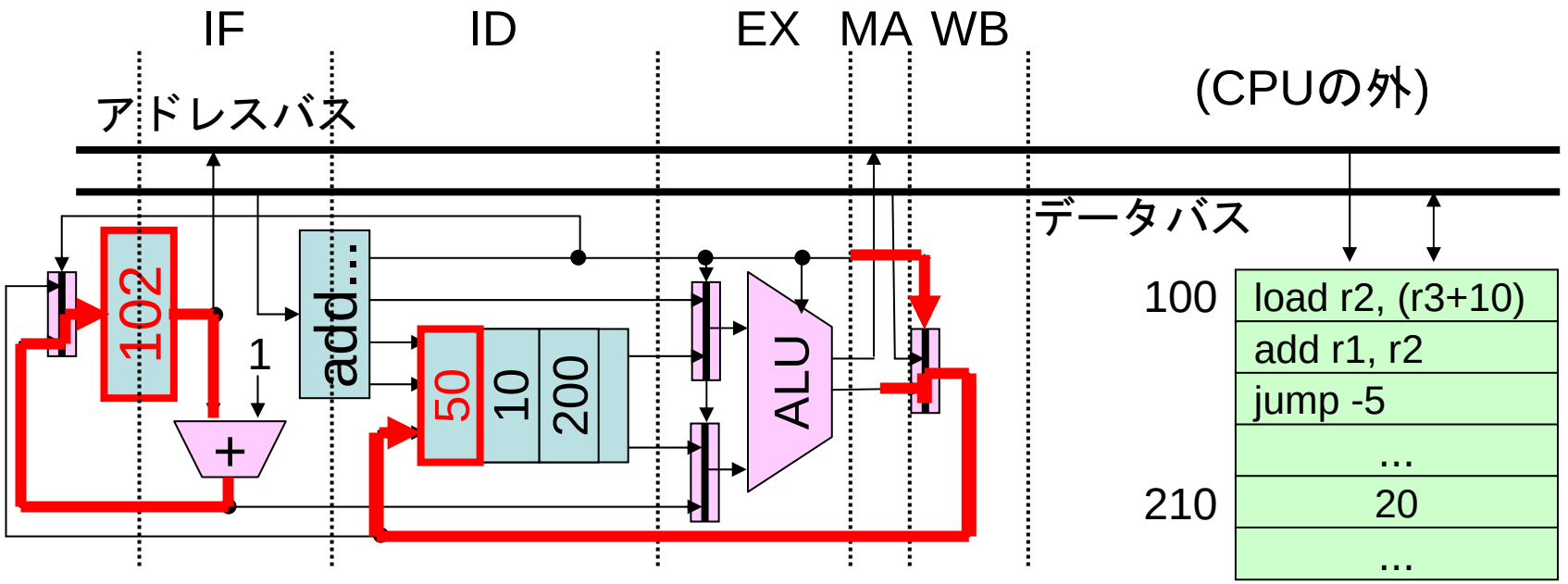
演算命令の実行(MA)

- 何も行わない
  - 主記憶アクセスがない命令は全て同じ



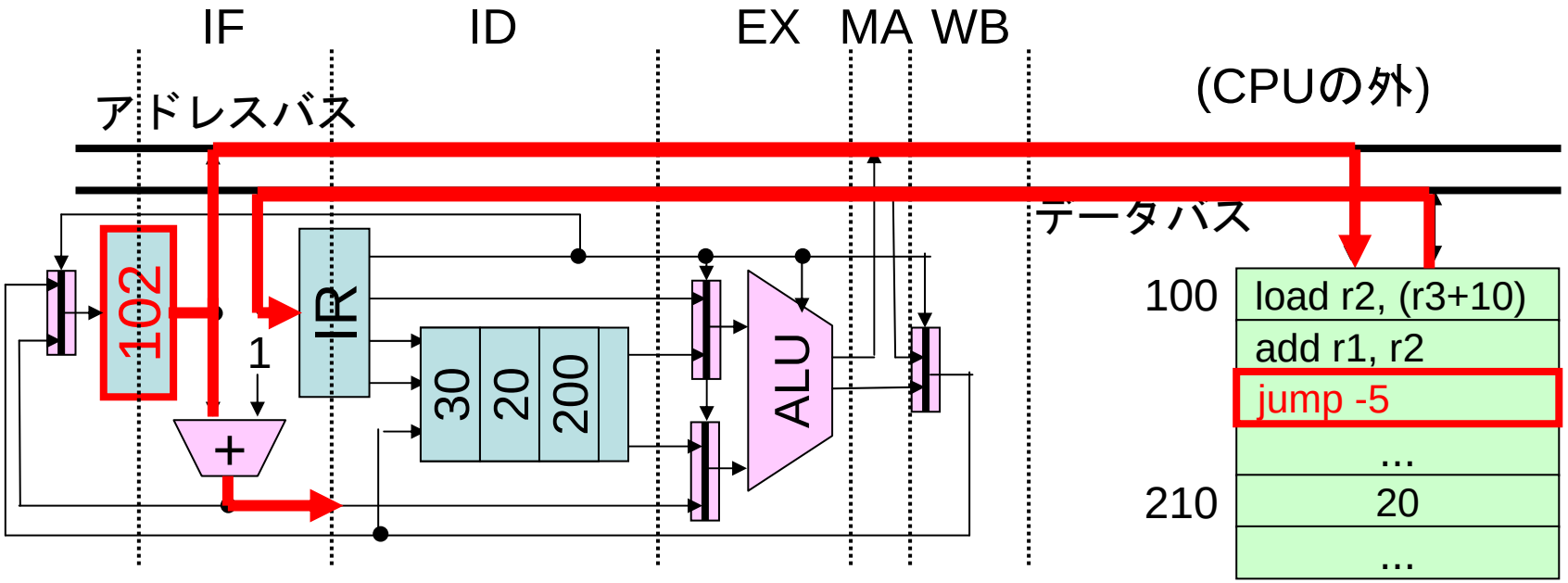
演算命令の実行(WB)

- 演算結果をレジスタr1として保存
- 次の命令の実行に備え、PCを+1して次の命令を指し示す



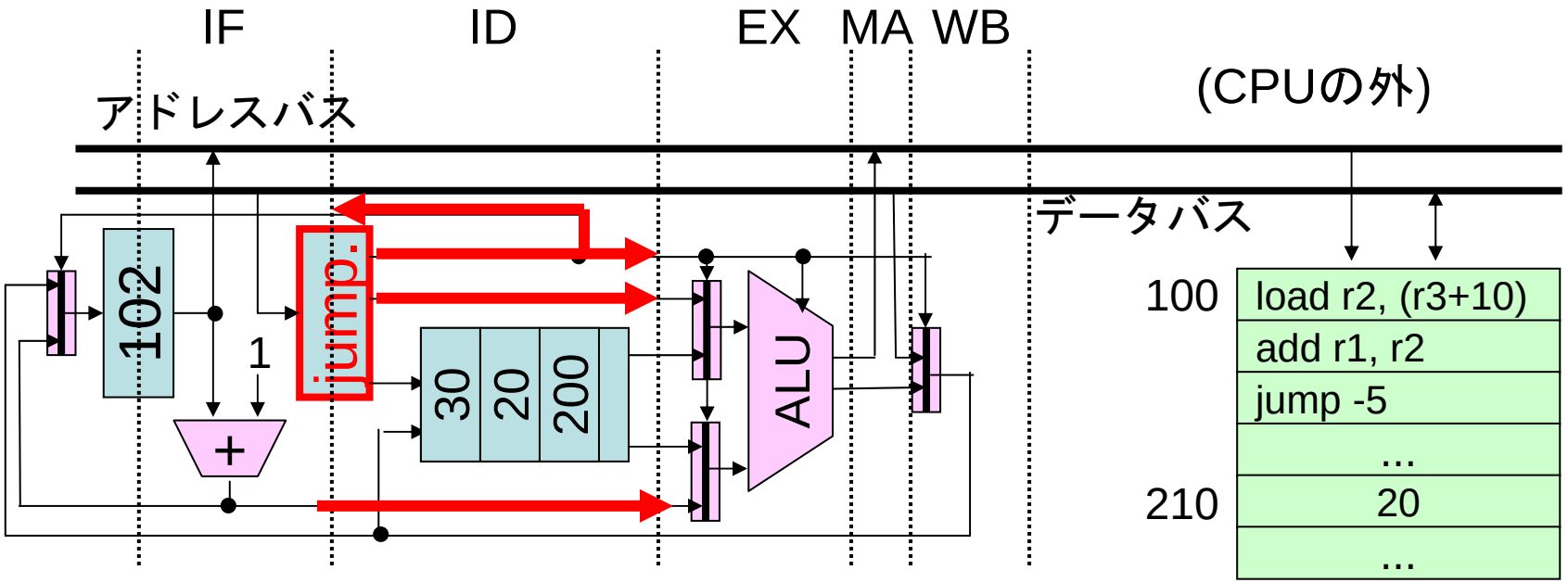
ジャンプ命令の実行(IF)

- ロード命令、加算命令に同じ
- あえて言うならば、PC+1の値が分岐先の計算の為にALUに送られる



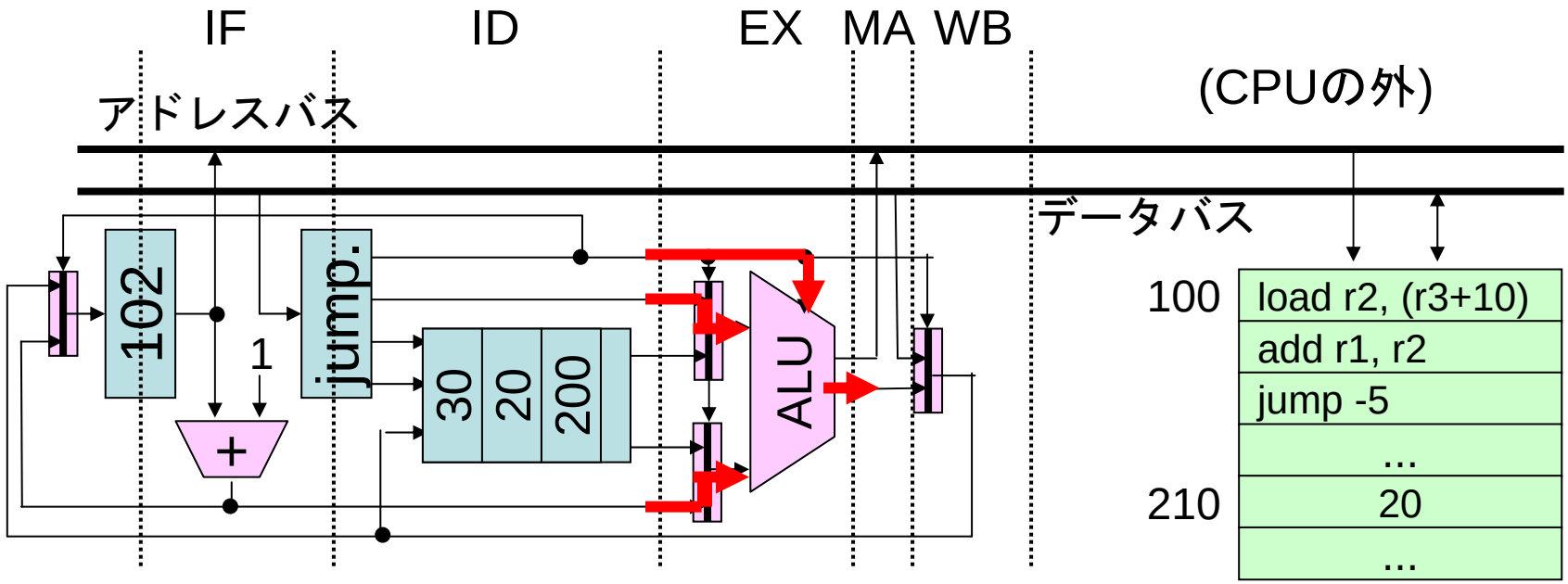
ジャンプ命令の実行(ID)

- PC+1の値をALUに送る
- 命令語から即値-5を切り出す
- PCの更新をALUの出力に切り替える準備



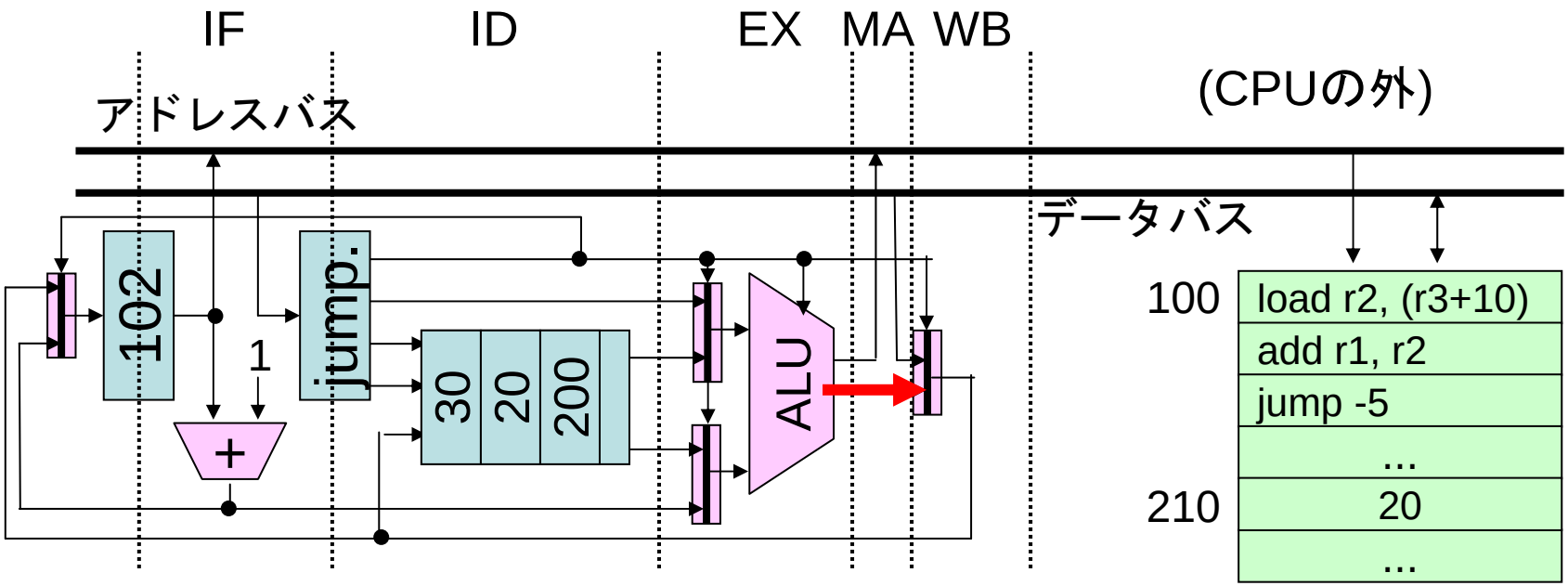
ジャンプ命令の実行(EX)

- PC+1(103)と即値-5を加算する



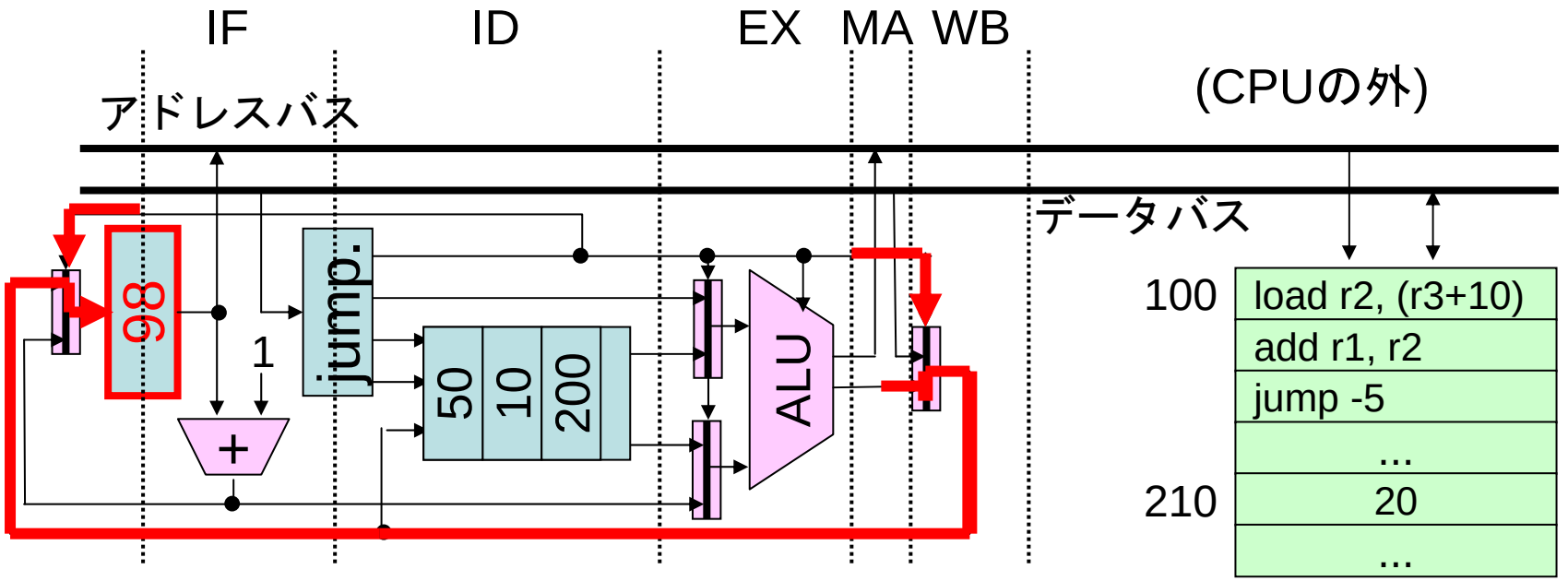
ジャンプ命令の実行(MA)

- 何もしない



ジャンプ命令の実行(WB)

- PCの値を演算結果である98に更新



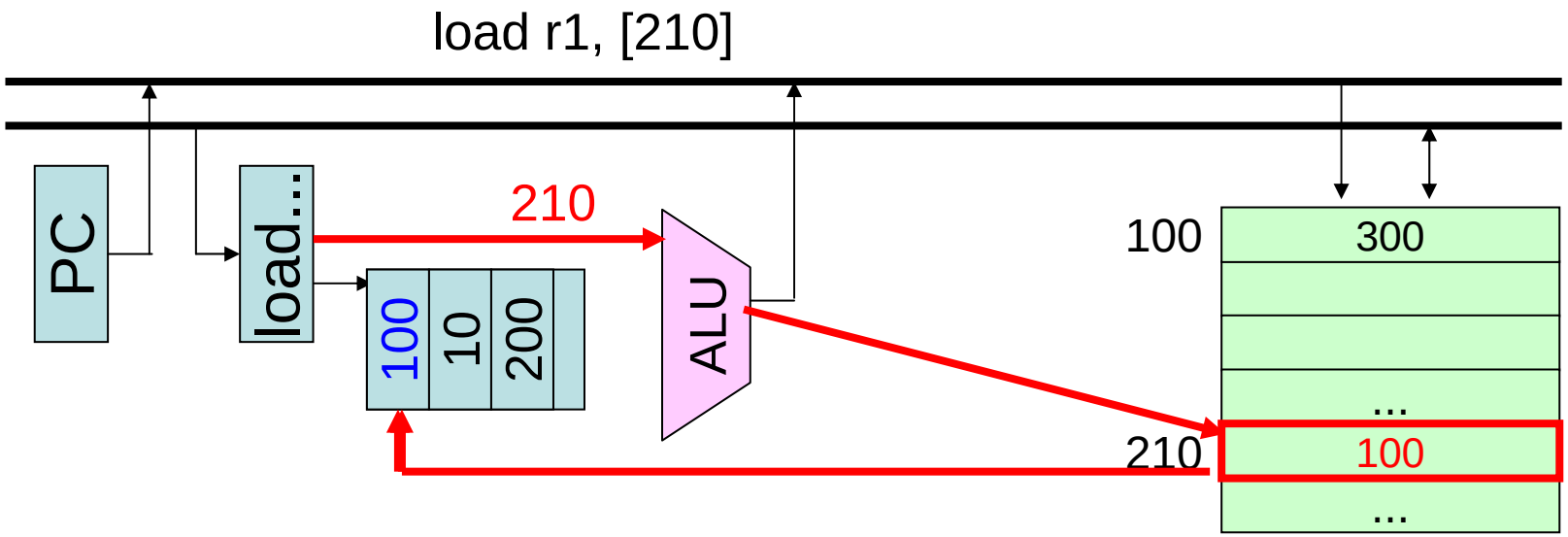
(3) アドレス指定方式

- メモリ中の命令語やデータは実効アドレスで示す
- 様々なアドレス指定方式(アドレッシングモード)が存在
  - 直接アドレス指定方式
  - オフセット付きレジスタ間接方式
  - メモリ間接指定方式
  - 即値オペランド指定方式

→うまく使い分けるとプログラムの記述が容易に

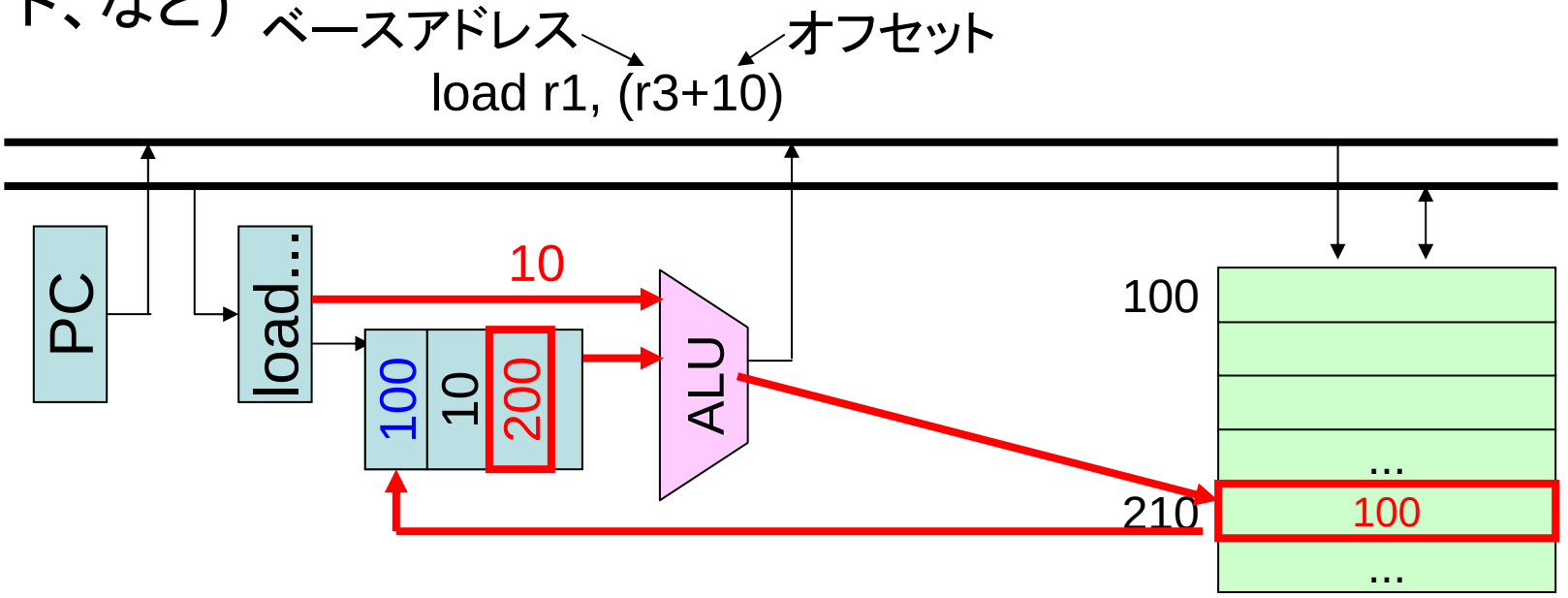
(a) 直接アドレス指定方式

- 命令語の一部をアドレスにする方法
- 命令語の長さに制約される



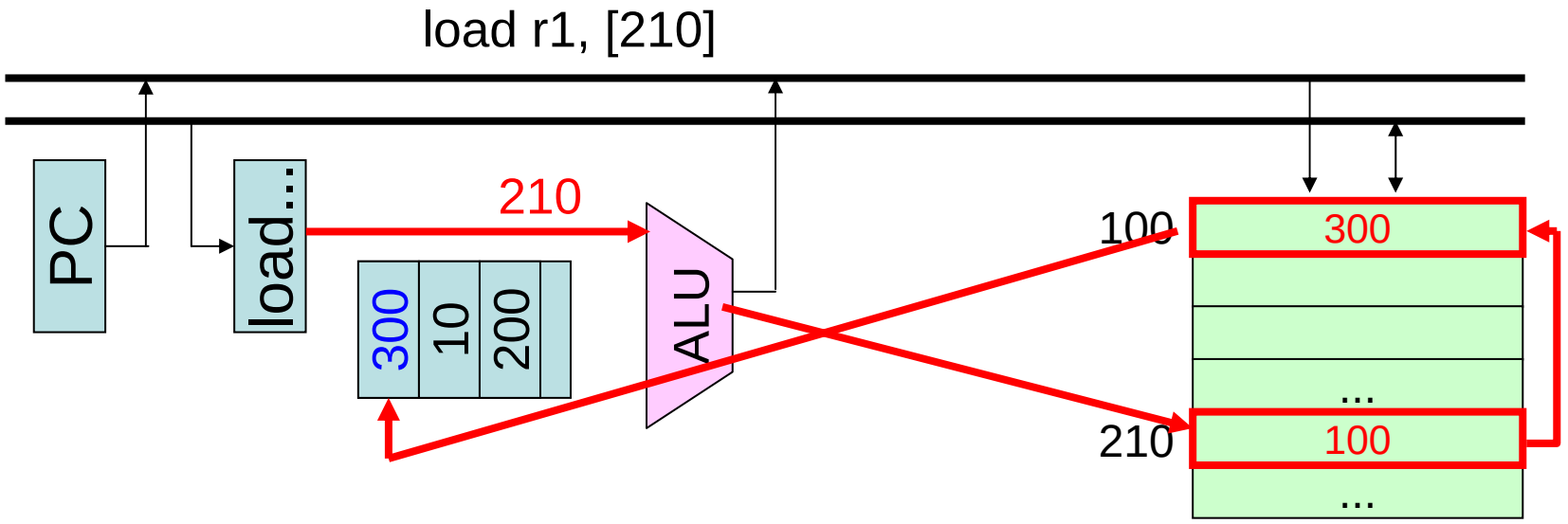
(b) オフセット付きレジスタ間接方式

- もっともよく使われる方式
- 配列のアクセス等で便利
  - 変数nをベースアドレスに
  - 変数”n+1”の+1をオフセットに
- 派生も多い(例：ベースのみ、ベース＋インデクス＋オフセット、など)



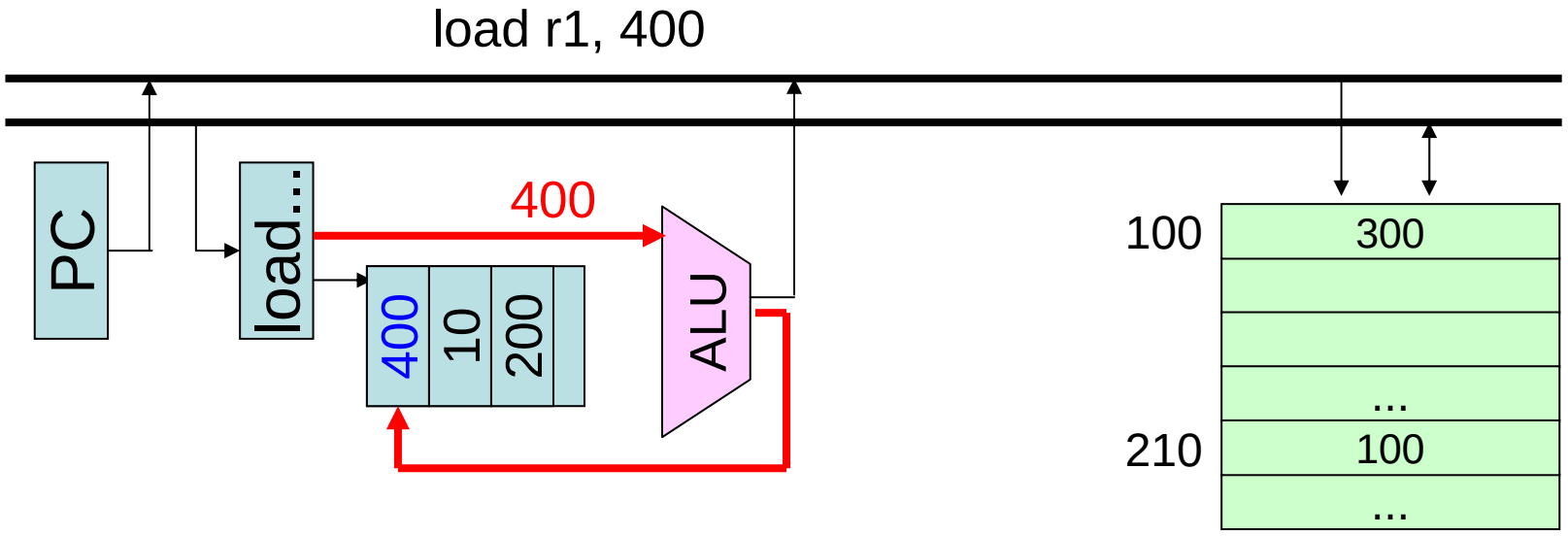
(c) メモリ間接指定方式

- 主記憶内のデータが実効アドレスとなる方式
  - 最初の実効アドレスは別途生成
- ライブラリ関数の呼び出し時のジャンプテーブルなどに便利
- 動作複雑なので使えないISAも多い



(d) 即値オペランド指定方式

- 命令語の一部を直接レジスタ値にする
- 主記憶アクセスは行われない



(4) CPUの性能指標

- クロック周波数
  - CPUは、コンピュータ内部にある発振器から発せられる信号に同期して動作
  - 発振器が「一秒間に何回発振しているか」と言う指標が、クロック周波数
    クロック周波数が1GHz＝1秒間に10億回の信号の振幅が発生
  - 同じ設計ならば、クロック周波数が高い方が性能が高くなる
- CPI (Clock cycles Per Instruction)
  - 1命令を実行するのに要するクロックサイクル数（クロックの振幅の数）
  - 例：Intel社のPentium 4の場合、1命令の実行に4～30クロックサイクル程度を必要
  - 命令によって、CPIは異なる
  - 後述するパイプライン化では、処理が並列に行われているため、平均CPIは単一命令の実行時よりも小さくなる

(4) 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100万単位で表したもの
  - 25MIPS = 25,000,000命令を１秒間に実行
- FLOPS(FLoting oint Operations Per Second)
  - １秒間に実行できる浮動小数点演算(倍精度浮動小数点)の数
  - スーパーコンピュータの性能指標によく使われる
    - 例：京コンピュータは10PFLOPSを目指して設計

ピーク性能(理論最大値)と
実効性能(プログラムを動作させて測定した値)に注意

処理性能の計算例

- 1.0×10^{15}個の行列演算を行う科学技術計算を1.0×10^{12}FLOPS(1TFLOPS)のスーパーコンピュータで実行する時の実行時間は？
  - ただし、演算に付随する処理は取るに足らないほど短時間で終了すると考える

$$1.0 \times 10^{15} \div 10 \times 10^{12} = 1000\text{秒}$$

処理性能の計算例

- 2GHzで動作しているCPUであるプログラムを動作させた時の平均CPIが0.5の場合、1秒あたりに実行される命令数は？

$$2.0 \times 10^9 \div 0.5 = 4 \times 10^9 \text{命令／秒}$$

処理性能の計算例

- 以下のような800MHzで動作するCPUの1秒あたりに実行される命令数は？
  - 平均CPIが演算命令で1、メモリアクセス命令で5、分岐命令で10である
  - 演算命令50%、メモリアクセス命令30%、分岐命令20%のプログラムを実行した時

平均CPIは$1 \times 0.5 + 5 \times 0.3 + 10 \times 0.2 = 4$
$800 \times 10^6 \div 4 = 200 \times 10^6$命令／秒

演習

- 3GHzで動作するCPUの平均CPIが浮動小数点演算命令で3、メモリアクセス命令で10、分岐命令で15である。浮動小数点演算命令50%、メモリアクセス命令40%、分岐命令10%のプログラムを実行した時の1秒あたりに実行される命令数は？