コンピュータの仕組み
(5)

柴山　潔

コンピュータの仕組み

1　コンピュータシステム
2　ハードウェア
3　内部装置
4　プロセッサ(2)
5　メモリ
6　外部装置
7　論理回路
8　オペレーティングシステム

4 プロセッサ(2)

4.1 プロセッサの動作とその制御(2)

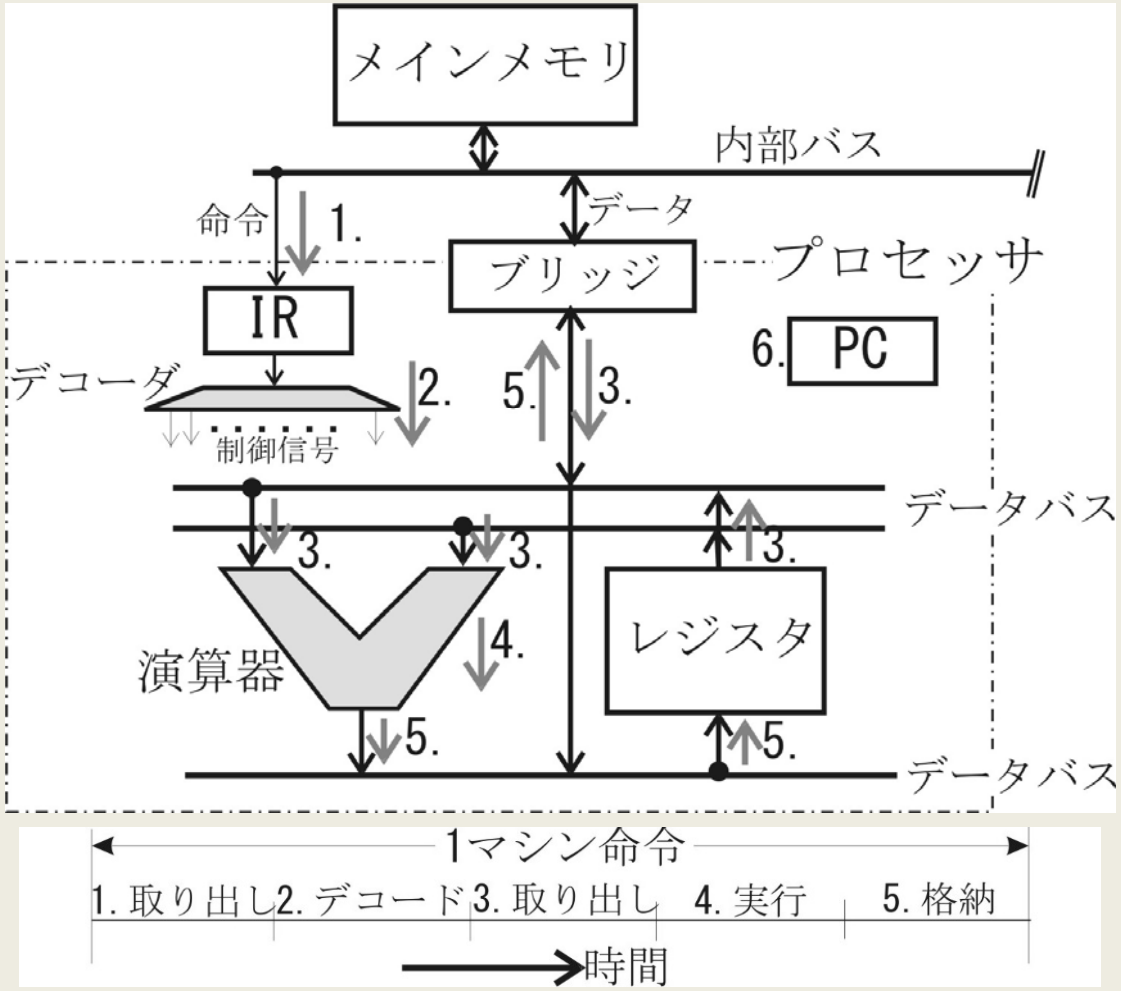
4.2 プロセッサの機能 ―演算と制御―

4.3 演算器

4.4 順序制御機構

命令実行サイクルとプロセッサの動作（図）

（再掲）



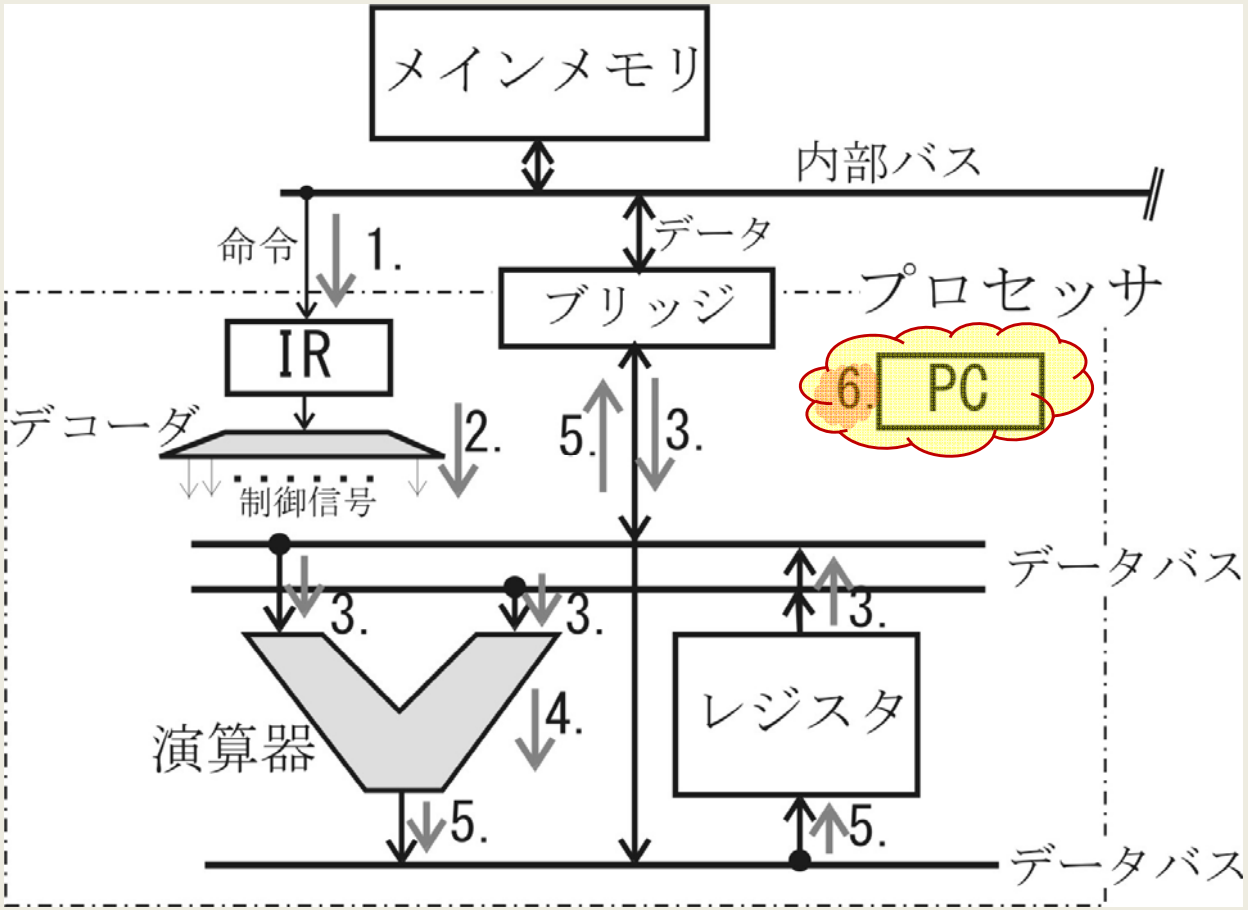
命令実行サイクルのステージ動作順

- 1～5のステージはこの順序でしかできない一連の操作
  - 1～5の各ステージは，それぞれ相異なるハードウェア機構によって実現
  - 各ステージのハードウェア機構は順に実行されるマシン命令列によって共用することが可
- 6のステージは，1～5のステージのいずれかと並行させることが可

命令実行サイクルのステージ動作(6)

6. 次命令アドレスの決定：次に実行すべき命令のアドレスをPCに設定
   - 今実行している命令が演算命令なら，単にPCをカウントアップ(count up; 増加順での計数)するだけでよいが，順序制御命令なら，次命令アドレスを計算/決定してPCに設定する必要

命令実行サイクルとプロセッサの動作（図）(6)



（再掲）

命令実行サイクルのステージ動作順

- 1～5のステージはこの順序でしかできない一連の操作
  - 1～5の各ステージは，それぞれ相異なるハードウェア機構によって実現
  - 各ステージのハードウェア機構は順に実行されるマシン命令列によって共用することが可
- 6のステージは，1～5のステージのいずれかと並行させることが可

命令実行サイクルのステージ動作順（実例）

（例）

- 先行して実行サイクルに入った命令Aが3のオペランド取り出しから4の実行ステージに到達すれば，Aに後続する命令Bが2の命令デコードから3のオペランド取り出しのステージに入ることが可

➢ 最新のプロセッサ（マイクロプロセッサ）では，1ステージを数ナノ秒で実行

マシン命令形式

- 命令コードやオペランドそれぞれどれくらいの長さにして，それらを1個のマシン命令中にどのように納める（エンコードする，符号化する）のか？
  - プロセッサ内の種々のハードウェア機構の規模や複雑さを左右

マシン命令長（例）

（代表例）2項演算のマシン命令

- その命令で，1個の命令コードと3個のオペランド（2個のソースオペランドと1個の結果オペランド）を指定する必要

➢ 1マシン命令中に都合4個の情報が必要

↓

決められた長さのマシン命令を効率よく使用するため，「4個」という数を減らせないだろうか？

アドレス形式

- 2項演算で指定すべきとされる3個のオペランドはすべて独立に指定可能でなければならないのだろうか？

- マシン命令中のオペランド数の違いによるマシン命令形式の分類方法（＝マシン命令中のオペランド数を減らす方法）

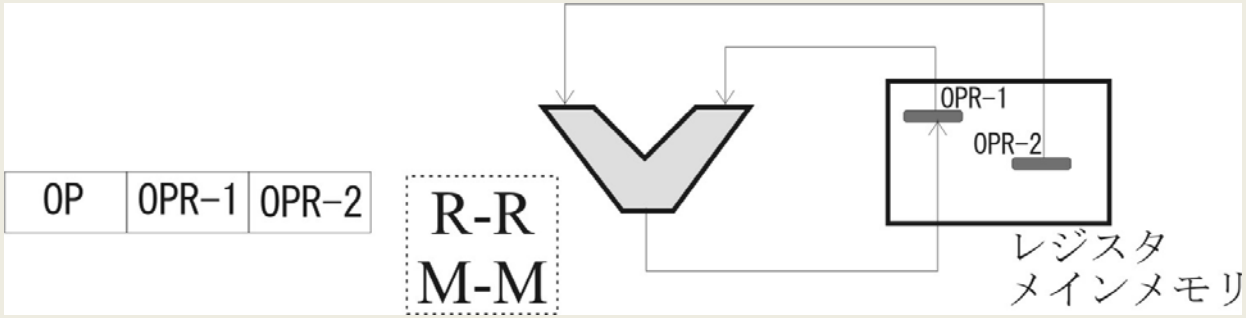
アドレス形式の分類

(1) 2アドレス形式（R-R形式，M-M形式）
(2) 1・1/2アドレス形式（R-M形式）

（凡例）　OP：命令コード　OPR：オペランド

2アドレス形式(1)

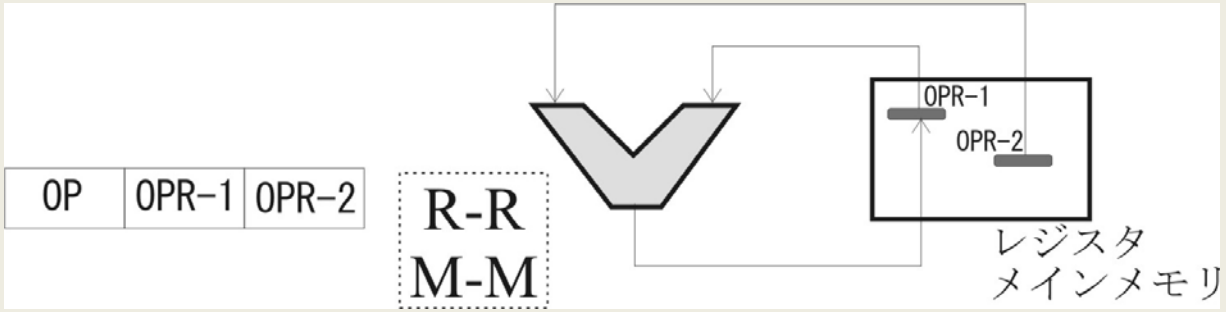
- 2個のうち1個のソースオペランドと結果オペランドの指定を同一のオペランドで
  - ○1個のオペランド指定が省略可
  - ×ソースオペランドの片一方は結果オペランド→ソースオペランドとして使われたデータは演算結果によって上書きされる（書き込まれて消される）



2アドレス形式(2)

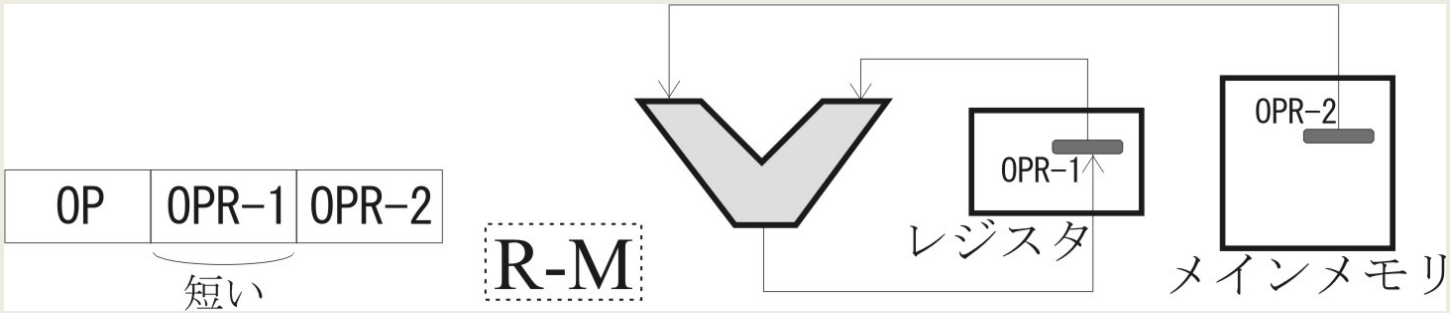
◆レジスターレジスタ(R-R)形式：2個のオペランドがいずれも汎用レジスタ番号である場合

◆メモリーメモリ(M-M)形式：2個のオペランドがいずれもメインメモリアドレス



1・1/2アドレス形式(1)

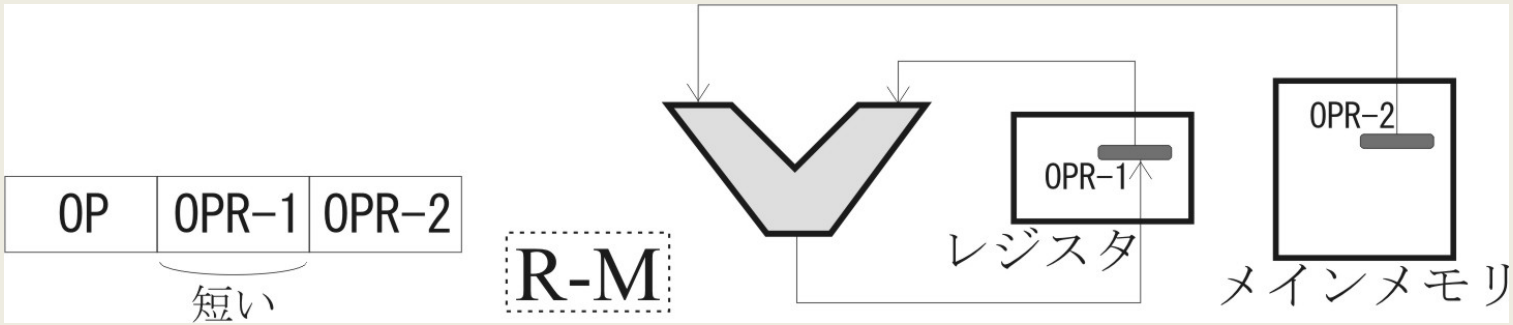
- 2アドレス形式の場合の，ソースオペランドと結果オペランドとを兼ねるオペランドをレジスタ番号と決めておく
  - ○限られたマシン命令長を活用 ← メインメモリアドレスを対象とするオペランドよりもレジスタ番号を指定するオペランド長は短くてすむという特徴を活用



1・1/2アドレス形式(2)

◆レジスターメモリ(R-M)形式：2個のオペランドとしてレジスタとメインメモリとを指定

- レジスタを活用する現代のコンピュータにおいて代表的なマシン命令形式



最新マイクロプロセッサのマシン命令形式

- 2アドレス形式と1・1/2アドレス形式とをR-R，M-M，R-M形式として使い分けるものが多い

オペランド指定

＝アドレス指定，アドレス修飾

- 演算対象のデータをオペランドで指定する方式

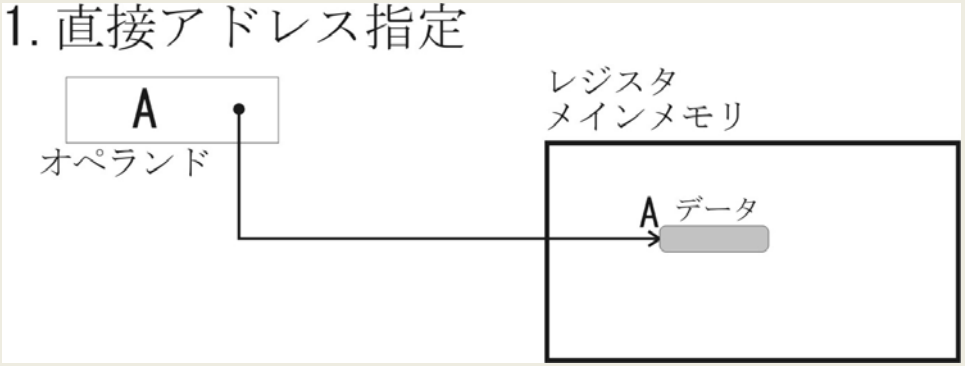
オペランド指定方式の分類(A)

■「マシン命令中のオペランドの使い方」による分類

1. 直接アドレス指定：オペランドがレジスタ番号かメインメモリアドレスかのいずれか
2. 間接アドレス指定：オペランドで指定された格納場所に入っているアドレスでもう一度データの格納場所を決定
3. 即値指定：オペランドそのものが演算対象データ

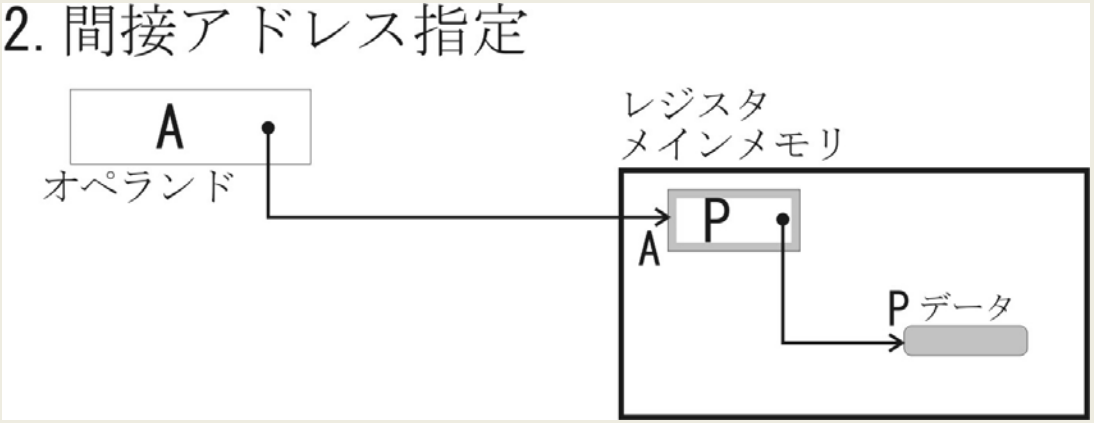
直接アドレス指定

- オペランドによって，「どのレジスタか，またはメインメモリ内のどのアドレスか」を決定，そこへアクセス(access)（読み出しか書き込み操作）
  - いったん決定されたアドレス（図では"A"）はオペランドとして命令中に書き込まれているので，実行中に書き換えは不可
  - そのアドレスに格納されているデータは実行中に更新可



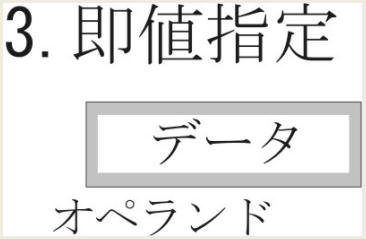
間接アドレス指定

- レジスタかメインメモリに2回アクセスする必要があり
  - 最初に読み出すアドレス（図では“P”）も，そのアドレスでアクセスするデータも，いずれもデータとして扱う（実行中に書き直したりする）ことが可



即値指定

- マシン命令中にデータを直接埋め込み
  - 命令中にデータが埋め込まれているので，実行中の書き直しは不可
  - 読み出しのソースオペランド指定でのみ使用



オペランドとして指定するデータの格納場所

- レジスタ：少数（実際には，十数～百数十個）
  - これらを識別するには，4～8ビットのレジスタ番号をオペランドとして指定
- メインメモリ （レジスタと比べると，次の特徴）
  - (A) 大容量性：格段に容量が大（数メガ*バイト以上が普通，バイトごとにアドレス付けすると，1メガバイトでも20ビットが必要）
  - (B) 拡張性：プロセッサ外にあるので増設が容易 → 増設によるアドレス増加（変更）に対処する必要

（* mega-, M; $\times 10^6$）

オペランド指定方式の分類(B)

■オペランドで指定するデータ格納場所がレジスタかメインメモリかによる分類

●レジスタ指定：レジスタ番号をオペランドとして指定

➢プロセッサ内に実装される汎用レジスタは少数 → その番号を直接指定するのが普通

●メモリ指定：メインメモリアドレスをオペランドとして指定

➢メインメモリの大容量性(1)や拡張性(2)に対処できる柔軟なアドレス指定方式が必要

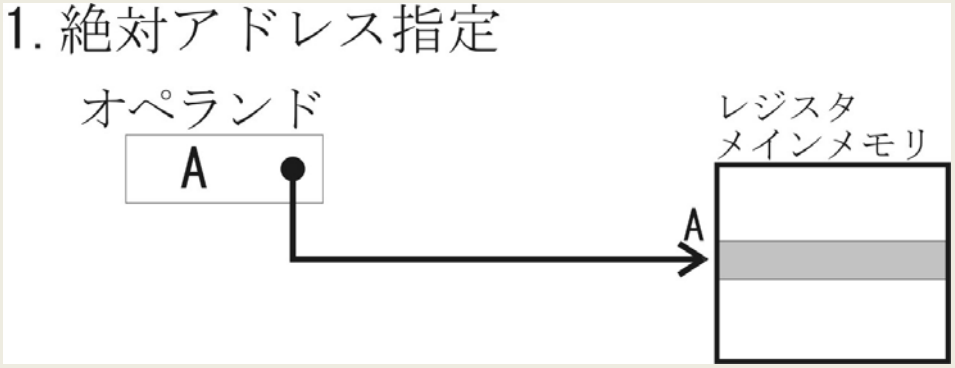
オペランド指定方式の分類(C)

■ 1個のオペランドを構成するアドレス情報が1個か2個かによる分類

1. 絶対アドレス指定：オペランドとして指定するアドレス情報は1個
2. 相対アドレス指定：オペランドはベース(base)とオフセット(offset)の2個のアドレス情報で構成・生成

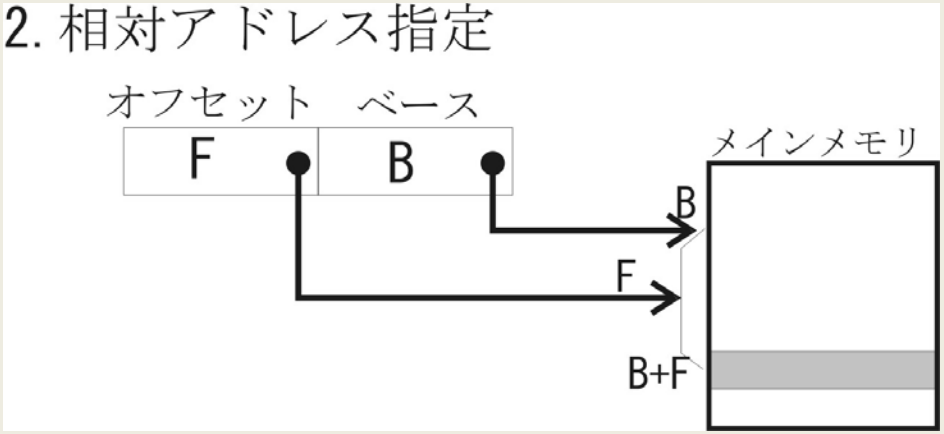
絶対アドレス指定

- オペランドとして指定するアドレス情報は1個で，それだけがレジスタ番号やメインメモリアドレスを生成する情報



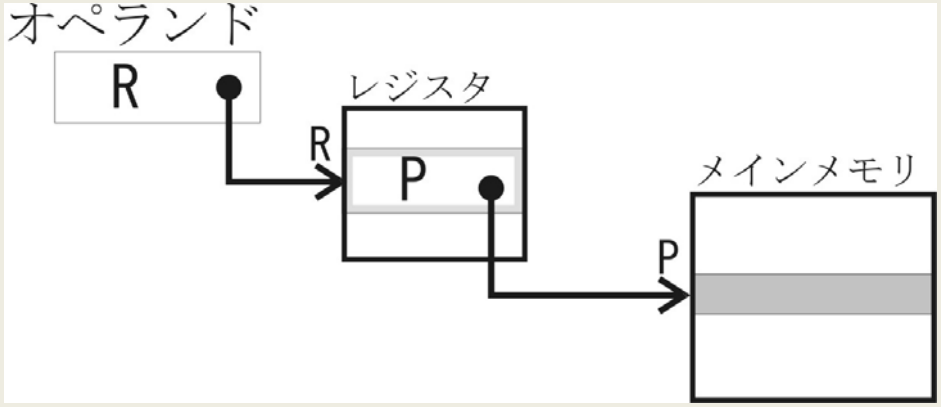
相対アドレス指定

- オペランドは，拠点アドレスとしてのベース(B)とそこからの相対的なずれを指すオフセット(F)で構成，それらを足し算(B+F)することによって，最終的なアドレスに
  - メインメモリの大容量性(1)や拡張性(2)にも対処可



レジスタ間接アドレス指定(1)

- 間接アドレス指定の「アドレス指定に時間がかかる」という短所の出現を防いで,「アドレス指定の柔軟性が確保可」という長所だけを活用するアドレス指定



レジスタ間接アドレス指定(2)

- 間接アドレスをレジスタに置く，レジスタ指定を活用した間接アドレス指定
  - 間接アドレス指定であるので，まず，アドレス（P）を読み出すが，それはレジスタ（R）へのアクセスで，メインメモリよりは速い
  - レジスタとメインメモリへの2回のアクセスを併せても，ほぼ直接アドレス指定のメインメモリへの1回のアクセスと同じ程度の時間でオペランド取り出しを行うことが可能

レジスタ間接アドレス指定 ＝ レジスタ指定 ＋ 間接アドレス指定