第3章　命令セットアーキテクチャ

3.1 命令

基本命令セット

- 命令セット
  - あるCPUが備えている命令の集まりを，そのCPUの命令セットという．
- 基本命令セット
  - どのコンピュータも備えているような命令の集まりを，基本命令セットという．

基本命令セット

- 基本命令セットの分類

分類	命令の内容
データ操作命令 （演算命令）	データを，処理・操作（演算）する命令． （ロード，ストア，算術演算，論理演算，比較演算，シフト演算など）
プログラム制御命令 （順序制御命令）	命令の実行順序を，明示的に制御・処理する命令． （分岐命令，コール，リターンなど）
OSだけが使用する命令	入出力装置を制御する命令，割り込みを行う命令など． 本講義では詳述しない

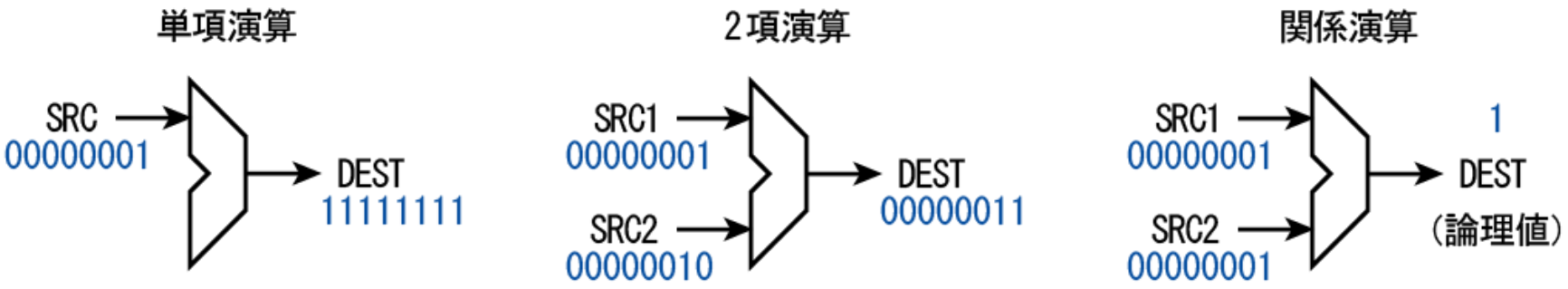
データ操作命令

- データ操作命令
  - データを，処理・操作（演算）する命令．
  - 以下の3種類に大別できる．
    - 算術演算命令
      - 数値データに対する演算命令．
    - 論理演算命令
      - 論理値データに対する演算命令．
    - ビット列操作命令
      - 各種のデータをビット列として操作する命令．

データ操作命令（算術演算命令）

- 算術演算命令の分類

	演算対象	結果	例
単項算術演算	1つの数値データ	1つの数値データ	符号反転など
2項算術演算	2つの数値データ	1つの数値データ	+, - など
関係演算	2つの数値データ	1つの論理値データ	=, <, > など

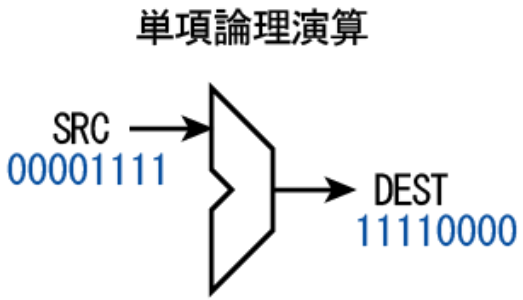


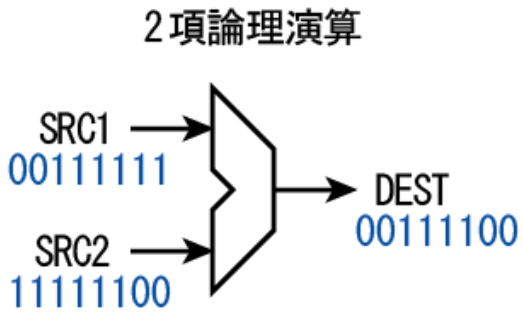
データ操作命令（論理演算命令）

- 論理演算命令の分類

	演算対象	結果	例
単項論理演算	1つの論理値データ	1つの論理値データ	否定
2項論理演算	2つの論理値データ	1つの論理値データ	論理和，論理積など

（注）論理演算命令は，ビットごとの独立した演算である．





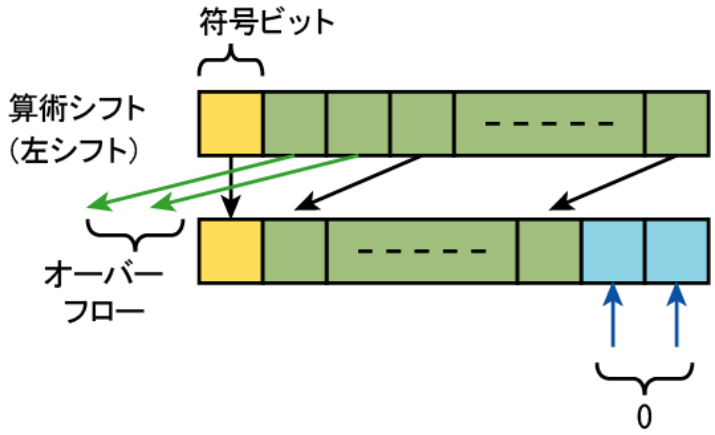
データ操作命令（ビット列操作命令）

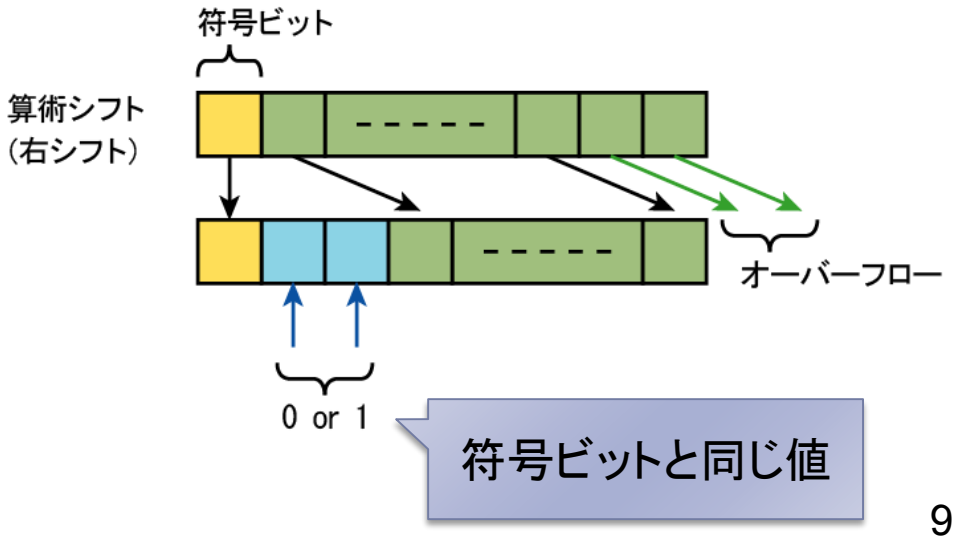
- ビット列操作命令の分類

	概要	例
コード変換	2進数や2進コードで表現する意味を保存しつつ，コードを変換する.	数値-2進コードの変換， 固定小数点数表現-浮動小数点数表現の変換， シフトJIS-EUCの変換など.
ビット列変換	ビット列そのものを操作する.	算術シフト，論理シフト，ビット列操作など.
転送	レジスタやメインメモリにあるデータの格納場所を移動する.	ロード，ストアなど.

データ操作命令（ビット列操作命令）［算術シフト］

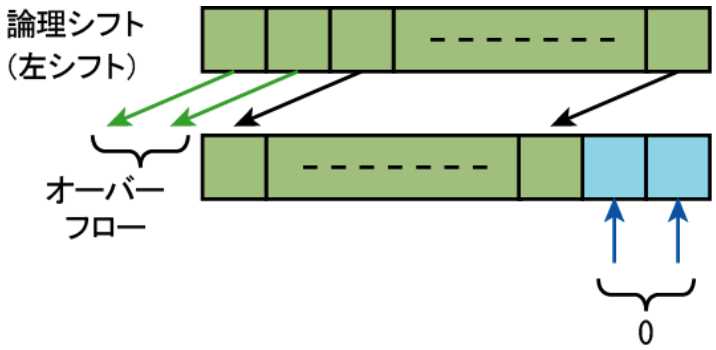
- 算術シフト
  - 数値データを対象とし，符号ビットを保存しつつシフトする．
  - 結果データも数値である．
  - nビット左シフトは，2^nを乗ずる演算と同じ結果を，nビット右シフトは，2^nで除する演算と同じ結果を与える．

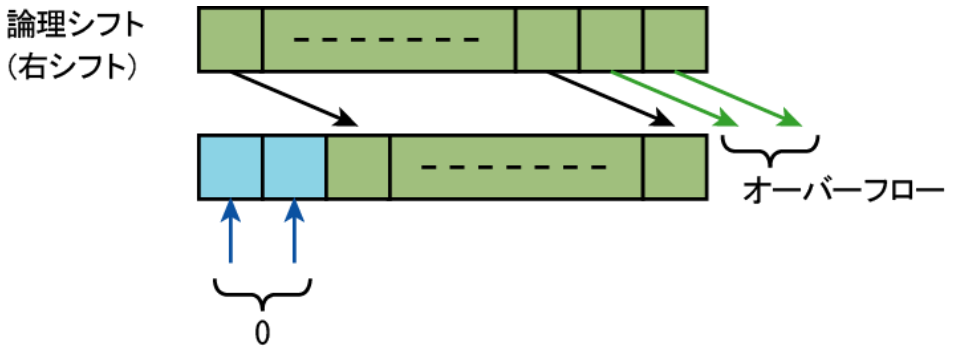




データ操作命令（ビット列操作命令）［論理シフト］

- 論理シフト
  - 論理値データを対象とし，対象データをすべて独立した論理値からなるビット列データとして操作する.
  - 結果データも論理値である.





データ操作命令（ビット列操作命令）［転送］

- 転送
  - レジスタやメインメモリにあるデータの格納場所を移動（コピー）する命令.
  - 転送先にあったデータは上書きされ，転送元のデータは変化しない.
  - レジスタ間転送，メインメモリ内転送，レジスタ・メインメモリ間転送がある.
  - レジスタ・メインメモリ間転送については，
    - ロード　：　メインメモリからレジスタ（プロセッサ内）への転送
    - ストア　：　レジスタ（プロセッサ内）からメインメモリへの転送

    と呼ぶ.

演習問題

- 問題1
  - $(24)_{10}$を2の補数表現を用いた1バイトの符号付き二進数で表せ.
  - また，この二進数を，2ビット右に算術シフトしたときに得られる二進数の十進数値と，2ビット左に算術シフトしたときに得られる二進数の十進数値を求めよ.

- 問題2
  - $(-4)_{10}$を2の補数表現を用いた1バイトの符号付き二進数で表せ.
  - また，この二進数を，2ビット右に算術シフトしたときに得られる二進数の十進数値と，2ビット左に算術シフトしたときに得られる二進数の十進数値を求めよ.

プログラム制御命令

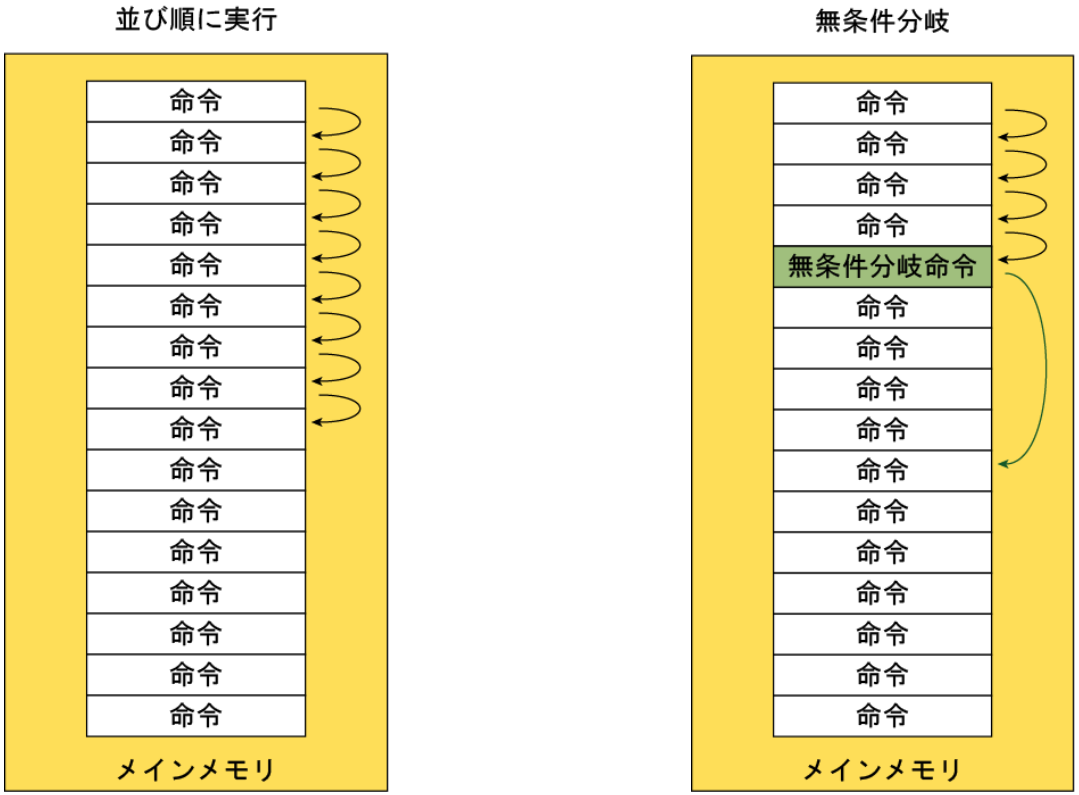
- プログラム制御命令
  - 命令は，原則として，メインメモリにおける格納順（並び順）に実行される．
  - 命令を並び順で実行したくないときには，オペランドとして次の命令のアドレスを明示的に指定する．
  - 「命令の実行順序を直接制御する命令」をプログラム制御命令あるいは分岐命令という．

- 主なプログラム制御命令
  - 無条件分岐
  - 条件分岐
  - サブルーチン分岐

  （次ページ以降で説明）

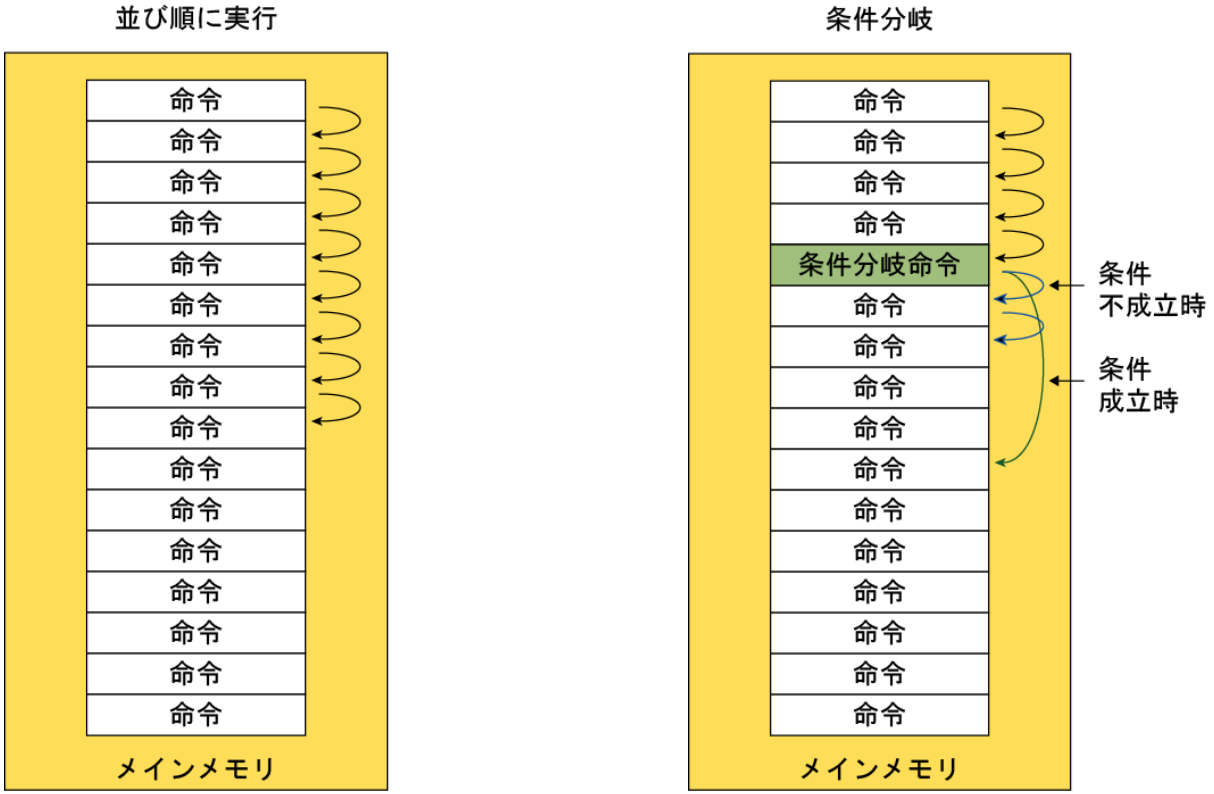
プログラム制御命令 ［無条件分岐］

- 無条件分岐
  - 無条件に，並び順ではなく，オペランドに対応するアドレスにある命令の実行に移る．



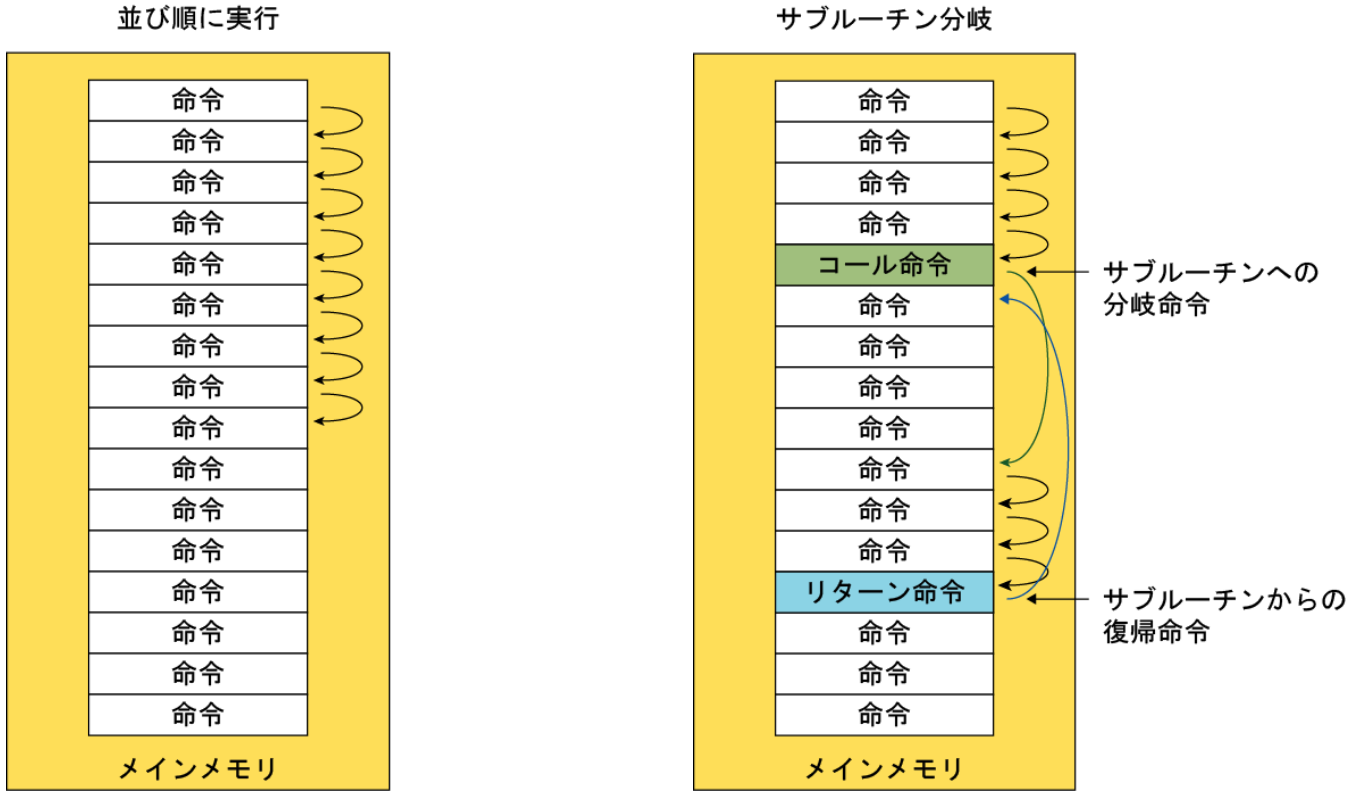
プログラム制御命令 ［条件分岐］

- 条件分岐
  - 条件成立の場合には，オペランドで指定するアドレスを分岐先とする．
  - 条件不成立の場合には，暗黙の指定によって，引き続く（並び順での）命令を次に実行する．



プログラム制御命令 ［サブルーチン分岐］

- サブルーチン分岐
  - あるプログラムの途中から，他のプログラム部分（サブルーチンという）へ分岐する．そのサブルーチンの実行が完了すると，元のプログラム（分岐元）へ戻る．



演習問題

- 問題3
  - データ操作命令とはどのような命令かを概略説明せよ.
- 問題4
  - プログラム制御命令とはどのような命令かを概略説明せよ.

CISCとRISC（初版P38，第2版P36）

- CISC (Complex Instruction Set Computer)
  - 複雑で高度な機能を，できるだけハードウェアで実現したもの．
  - 1個の命令で，複雑な処理を行うことができる．
  - 一方で，ハードウェアが複雑になる．

- RISC (Reduced Instruction Set Computer)
  - ハードウェアで分担する機能を，できるだけ整理して単純化したもの．
  - 簡単なハードウェアで実現することができる．
  - 一方で，複雑な処理を行うためには，複数の命令を組み合わせる必要がある．

命令機能の評価

- 命令機能の評価指標
  - TPI=TPC×CPI
    - TPI　（平均命令実行時間；time per instruction）
      - 1命令の平均実行時間（秒）
    - TPC　（マシンサイクル時間）
      - クロックの周期（クロック周波数の逆数）
    - CPI　（平均命令実行サイクル数；clock cycle per instruction）
      - 1命令を実行するのに必要な平均のクロック数

- TPCは，主として実装技術によって決まる．
- CPIは，主として命令セットアーキテクチャの複雑度によって決まる．
  したがって，CPIは，コンピュータアーキテクチャを定量的に評価する指標となる．
- CISCは，RISCに比べてCPIは大きくなる．

演習問題

- 問題5
  - マシンAのクロック周波数を1GHzとし，マシンBのクロック周波数を500MHzとする.
  - また，マシンAとマシンBは，同じ命令セットアーキテクチャを実現しており，同じプログラムに対して必要となる命令数は等しいものとする.
  - いま，あるプログラムに対して，マシンAのCPIが2.0であり，マシンBのCPIが1.2である.
  - どちらのマシンが，このプログラムを速く実行することができるか？

3.2　アドレッシング

アドレッシングとは

- 命令中のオペランドには，命令で使用するデータやデータの格納場所（アドレス）が埋め込まれる．

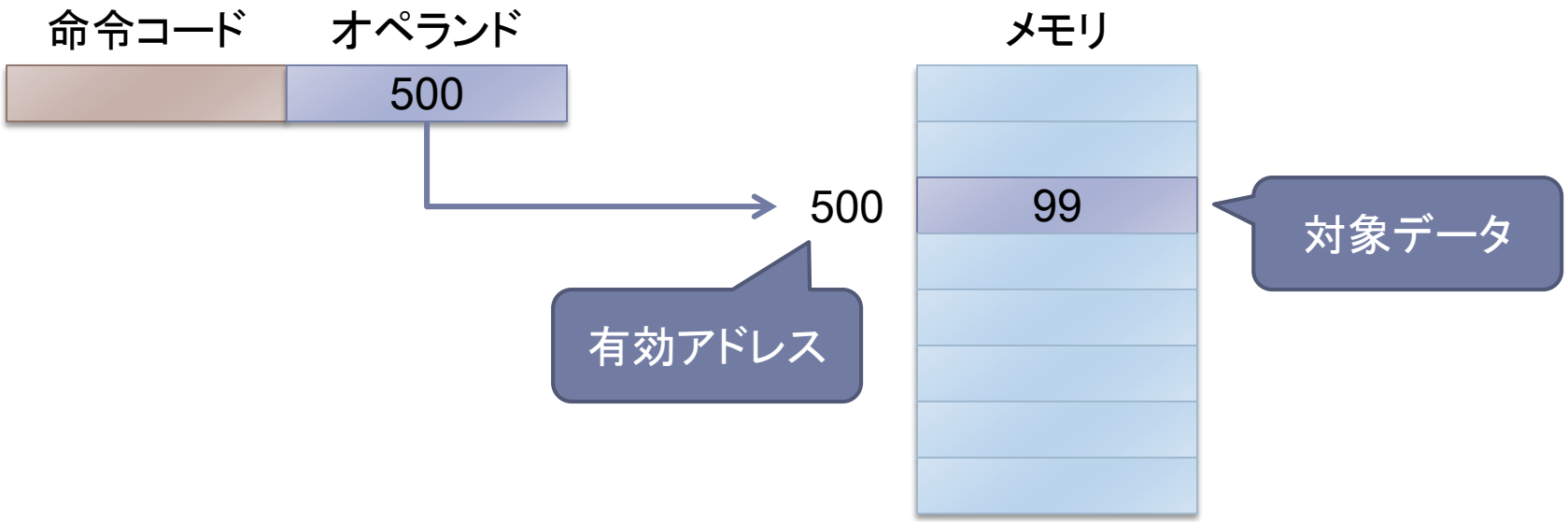
一般に，個々のコンピュータに実装されるメインメモリ容量は様々である．
また，効率的に，メインメモリにアクセスしたい．

- そのため，命令中のオペランドには，メモリアドレスそのものではなく，「メモリアドレスを生成するための情報」を埋め込む．
- 現在，コンピュータでは，各種のアドレスの表し方が用いられている．

最終的に参照されるアドレスを有効アドレス（実効アドレス）といい，有効アドレス（実効アドレス）を決めるための操作や，処理対象となるデータを決める操作を，アドレッシングという．

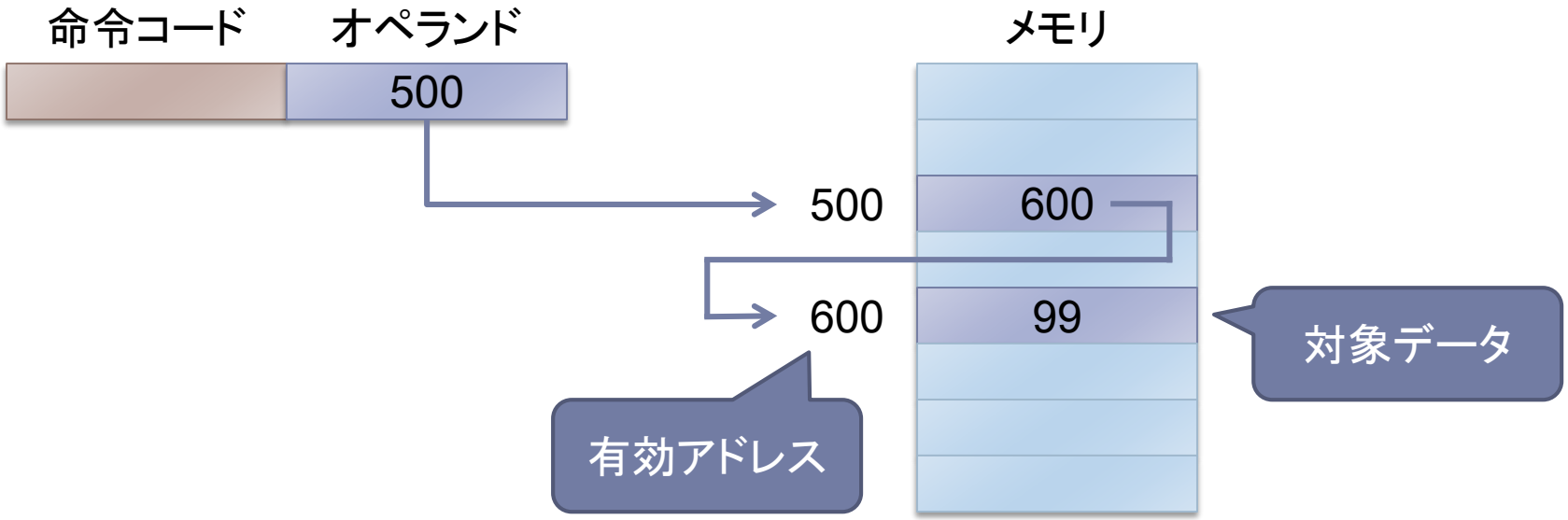
直接アドレッシング

- 直接アドレッシング
  - 「命令のオペランドに記述した値が示すアドレス」に格納されている値を「処理対象」とする方式.



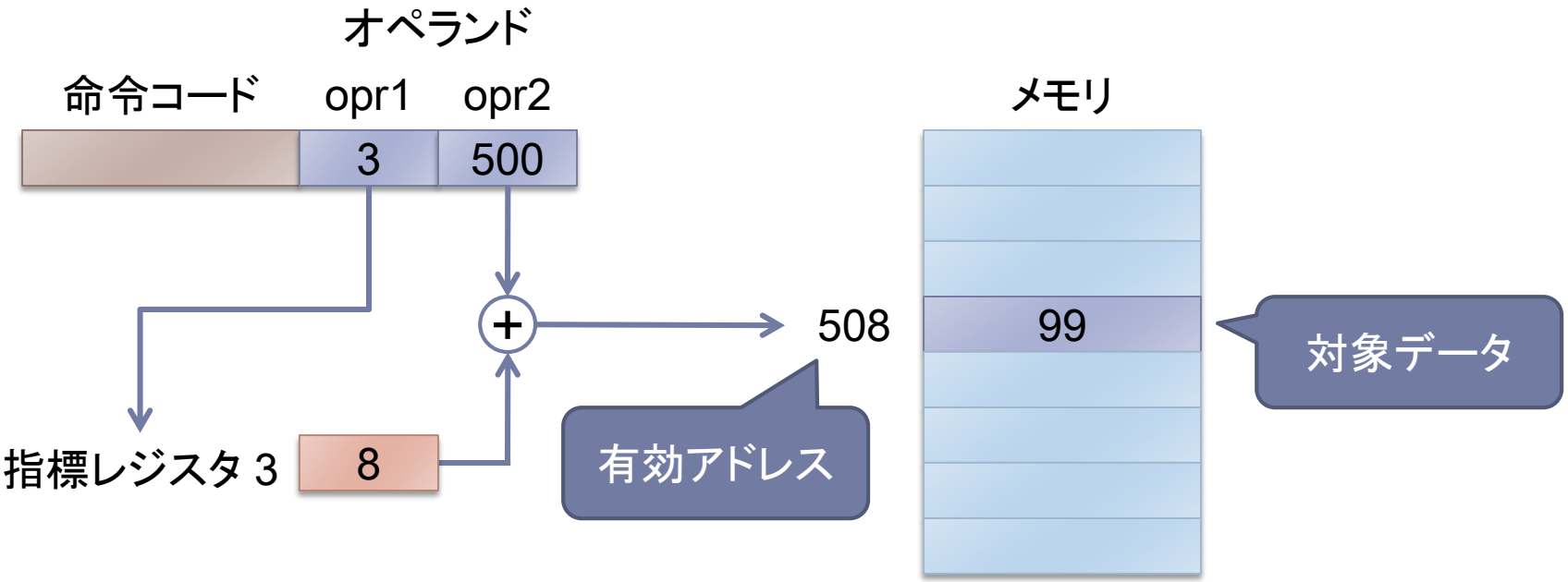
間接アドレッシング

- 間接アドレッシング
  - 「命令のオペランドに記述した値が示すアドレス」に格納されている値を「有効アドレス」とする方式.
  - 始めに参照するメモリの内容を書き換えることで，処理対象とするデータを変更することができる.



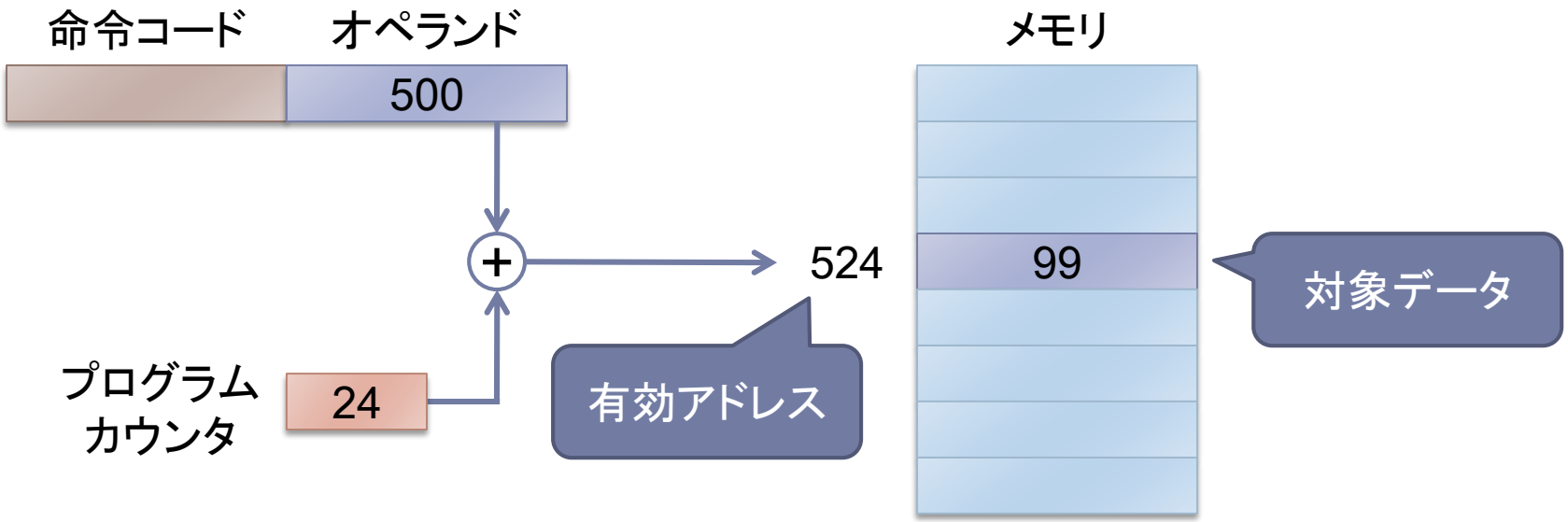
指標（インデックス）アドレッシング

- 指標（インデックス）アドレッシング
  - 「オペランドopr1で指定した指標（インデックス）レジスタに格納されている値」と「オペランドopr2に記述した値」を加算した結果を「有効アドレス」とする方式.
  - 指標（インデックス）レジスタの値を1加算（減算）していけば，メモリの連続した領域のデータを処理対象とすることができる.



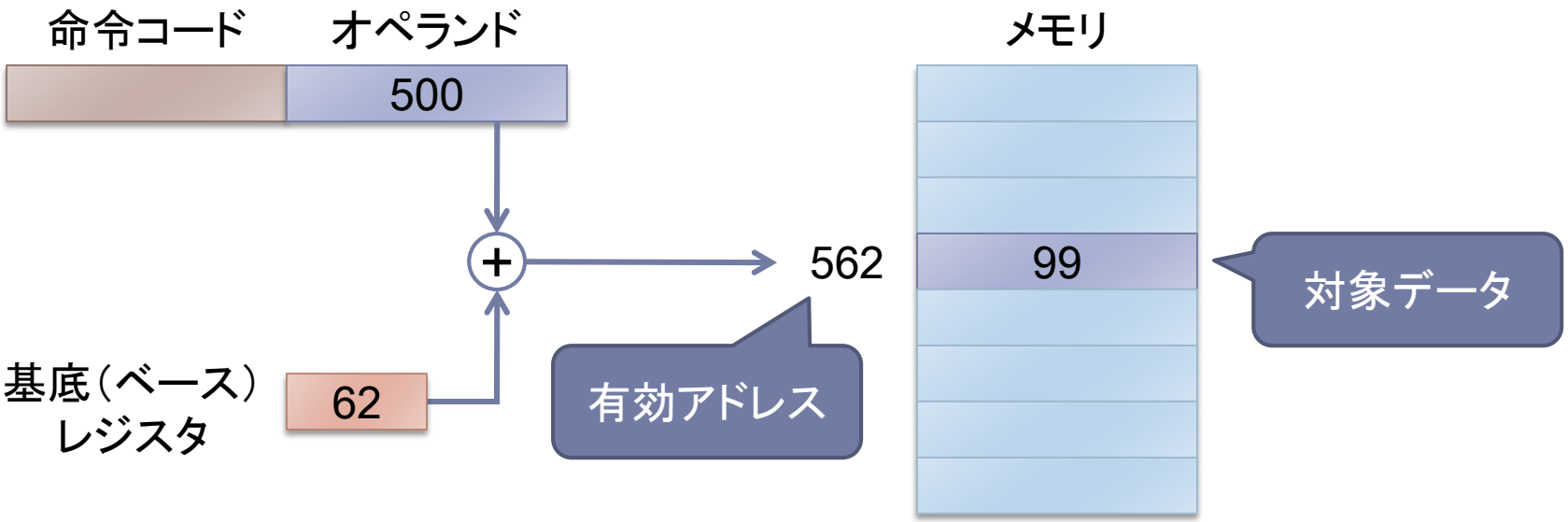
相対アドレッシング

- 相対アドレッシング
  - 「プログラムカウンタに格納されている値」と「オペランドに記述した値」を加算した結果を「有効アドレス」とする方式.
  - プログラムカウンタには，現在実行中の命令のアドレスが格納されているため，実行位置からの相対的なアドレスを指定することができる.



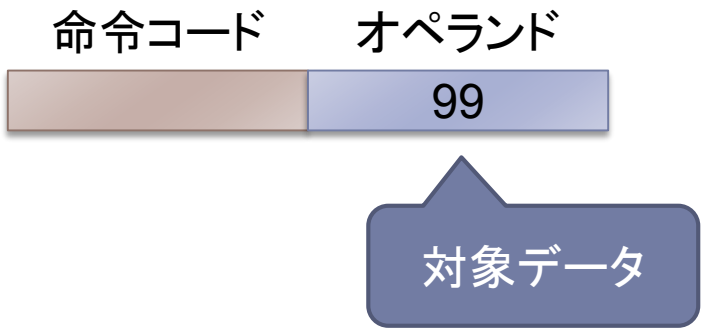
基底（ベース）アドレッシング

- 基底（ベース）アドレッシング
  - 「基底（ベース）レジスタに格納されている値」と「オペランドに記述した値」を加算した結果を「有効アドレス」とする方式.
  - 基底（ベース）レジスタには，プログラムの先頭アドレスが格納されており，この値はOSが管理している．プログラムをメモリの他の領域に移動するのに使用できる.



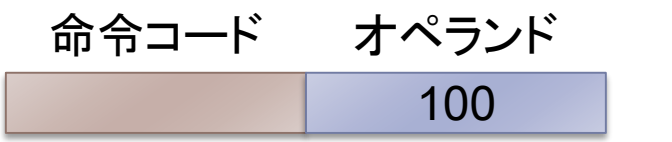
即値（イミーディエイト）アドレッシング

- 即値（イミーディエイト）アドレッシング
  - オペランドに記述した値をそのまま処理対象データとする方式.
  - メモリを参照する必要がないので，データ格納領域を節約でき，高速な処理が可能.
  - データが命令に埋め込まれてしまっているため，データの変更が容易ではない.



演習問題

- 問題6
  - 命令とメモリの状態を，概略的に，以下に示す．以下の各アドレッシング方式を用いた場合の，対象データの値を示せ．
    - ① 直接アドレッシング
    - ② 間接アドレッシング
    - ③ 相対アドレッシング
    - ④ 即値アドレッシング



プログラムカウンタ　3

基底（ベース）レジスタ　2

	メモリ
97	98
98	107
99	3
100	102
101	103
102	105
103	99
104	0
105	108
106	4
107	0