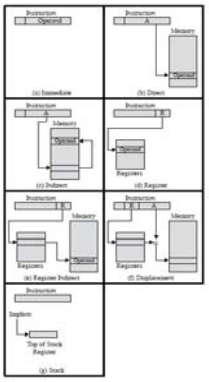
アドレス指定
Addressing Mode

櫻井 彰人

最初にまとめ



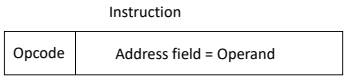
モード	mode	アドレス計算	主な長所	主な短所
即値	immediate	オペランド = A	メモリ参照なし	オペランドの大きさに制約
直接	direct	EA = A	簡単	アドレス空間に制約
間接	indirect	EA = (A)	アドレス空間大	複数回の目盛参照
レジスタ	register	EA = R	メモリ参照なし	アドレス空間に制約
レジスタ間接	register indeirect	EA = (R)	アドレス空間大	メモリ参照あり
オフセット	displacement	EA = A + (R)	融通性大	複雑
スタック	stack	EA = スタック先頭	メモリ参照なし	利用法に制約

アドレス指定方式

- 即値（イミディエイト）（Immediate）
- 直接（直接）
- 間接（Indirect）
- レジスタ（Register）
- レジスタ間接（Register Indirect）
- 変位・オフセット（Displacement）
- スタック

即値アドレス指定

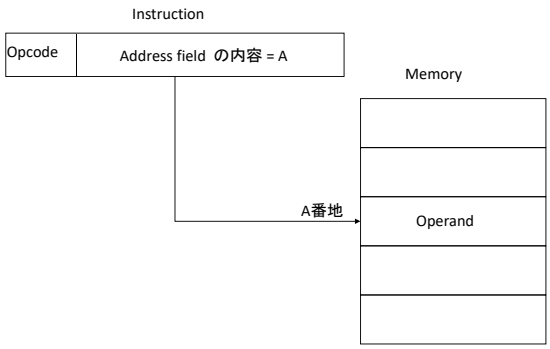
- Operand は命令の一部
- Operand が アドレスフィールド そのもの
- 例. ADD 5
  - アキュムレータの内容に 5 を加える
  - この 5 が operand
- データを取り出すためのメモリ参照はなし
- 速い
- アドレス範囲に強い制約



直接アドレス指定

- アドレスフィールドには、オペランドのアドレスが入っている
- 実効番地 Effective address EA = address field A
- e.g. ADD A
  - 番地 A のメモリ内容をアキュムレータに加える
  - オペランドを得るために、番地Aのメモリを見に行く
- データにアクセスするために、一回メモリ参照する
- 実行番地の計算は、それだけ
- アドレス空間が制限される

図 直接アドレス指定



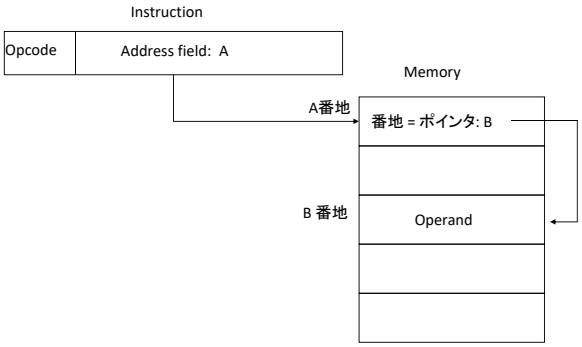
間接アドレス指定 (1)

- （命令の）アドレスフィールドが指し示すメモリセルに、オペランドのアドレスがある（オペランドへのポインタがある）
- EA = (A)
  - A 番地を見る, そこの内容「(A)」を番地だと思って、オペランドを探す
- e.g. ADD (A)
  - A番地の内容を番地とする（A番地によってポイントされる）メモリセルの内容をアキュミュレータに足す

間接アドレス指定 (2)

- アドレス空間が広い
- 2^n ただし n = 語長
- ネストさせる、レベルを深くする、多段階
  - 例. EA = (((A)))
    - どのような図になるか？
- オペランドを見出すのに、複数回メモリアクセスが必要
- そのため、遅くなる

図 間接アドレス指定



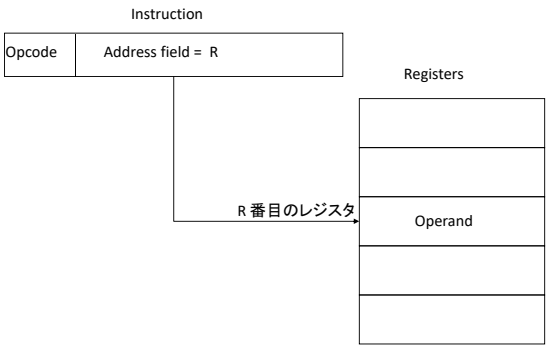
レジスタアドレス指定 (1)

- オペランドは、アドレスフィールドで指定したレジスタ内にある
- EA = R
- レジスタ数は限られている
- アドレスフィールドは非常に小さくてよい
  - 命令語長が短い
  - 命令のフェッチが速い

レジスタアドレス指定(2)

- メモリアクセスなし
- 実行が非常に速い
- アドレス空間が狭い（レジスタ数分しかない）
- レジスタ数が多いと性能が向上する
  - アセンブラプログラミング能力が必要。またはよいコンパイラ作成が必要
  - 注意: C 言語に次の表現あり
    - register int a;
  - PDP-11 はこれを活用
- 比較: 直接アドレス

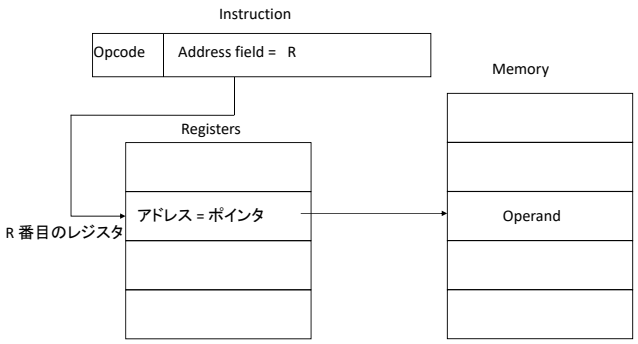
図 レジスタアドレス指定



レジスタ間接アドレス指定

- 比較: 間接アドレス
- EA = (R)
- レジスタ R の内容が番地であり、それが指し示す（ポイントする）アドレスに、オペランドがある
- アドレス空間は広い (2^n)
- 間接アドレスに比べると、一回メモリアクセスが少ない

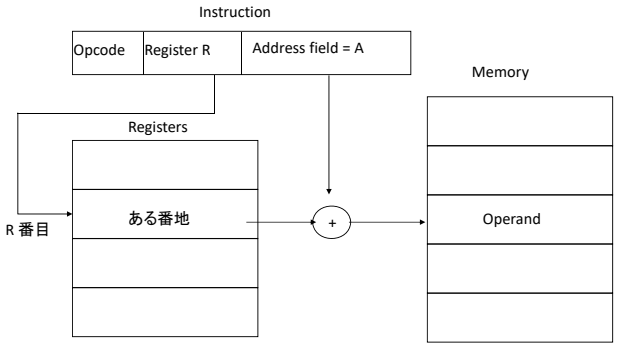
図 レジスタ間接アドレス指定



オフセットアドレス指定

- EA = A + (R)
- Address field には2つの値
  - A = 基底の値
  - R = オフセットを保持するレジスタの番号
  - または、その逆、すなわち
  - A = オフセットの値
  - R = 基底の値を保持するレジスタの番号

オフセットアドレス指定



相対アドレス指定（PC相対）

- 偏差アドレス指定の一変形
- R = Program counter, PC
- EA = A + (PC)
- i.e. 現在のプログラムカウンタPCの指し示す場所からA番地先のメモリセルに、オペランドがある
- 注意: 参照の局所性とキャッシュの利用
  - cache vs cash

インデックスアドレス指定

- A = 基底
- R = オフセット
- EA = A + R
- 配列のアドレスを指定するのに便利
  - EA = A + R
  - R++

組合せ

- Postindex
- EA = (A) + (R)

- Preindex
- EA = (A+(R))

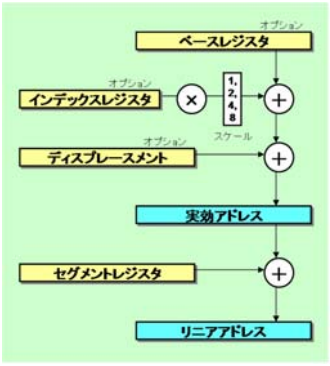
- (図は省略)

スタックアドレス指定

- Operand は (暗黙的に) スタックの先頭
- 例.
  - ADD　スタックの先頭から2項取り出し、足し算をし、スタックの先頭におく

X86アーキテクチャのアドレス指定

大昔の負の遺産
を引きずっている



日本語Wikipediaより