命令セットアーキテクチャ

コンピュータアーキテクチャ

命令セットアーキテクチャ

- 「命令セット」
  - そのコンピュータ用に用意されてる命令たちのこと
- 「命令セットアーキテクチャ」
  - その命令たちの動作と表現形式を定めたもの
- => 設計者に仕様を、ユーザにそのコンピュータで何ができるのかを教えてくれる

レジュメ

- 命令の表現形式とアセンブリ言語
- 命令セット
- アドレシング
- サブルーチンの実現
- まとめ

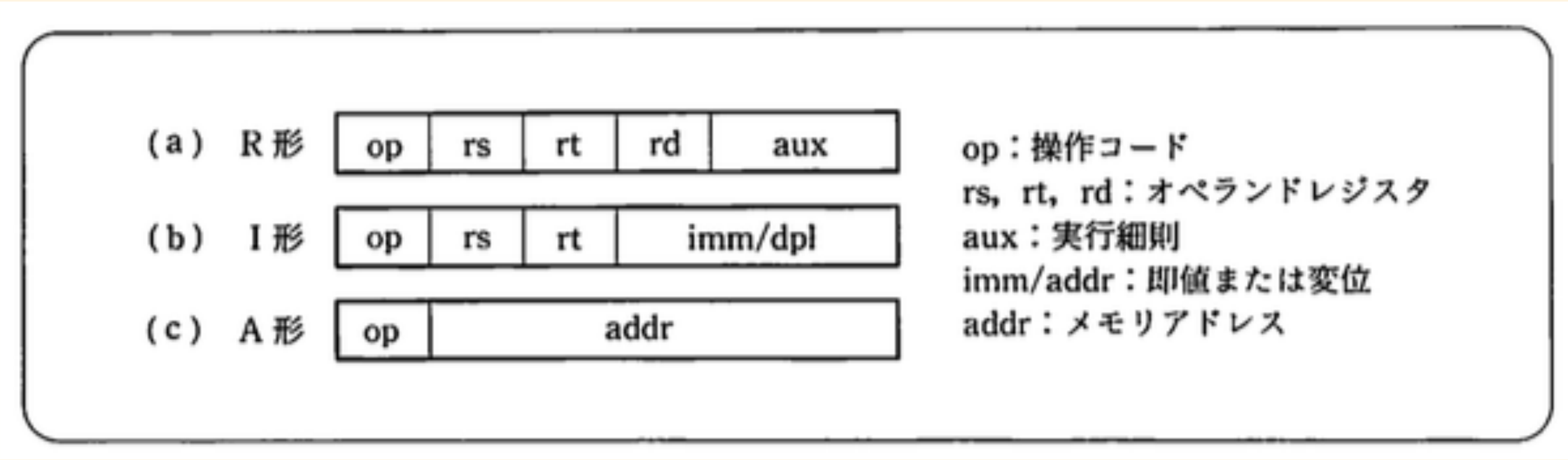
命令の表現形式とアセンブリ言語

操作とオペランド

- 一つの命令は「操作(operation)」と「対象(operand)」の組
- 命令形にはいくつかの形式がある
  - e.g. R形, I形, A形

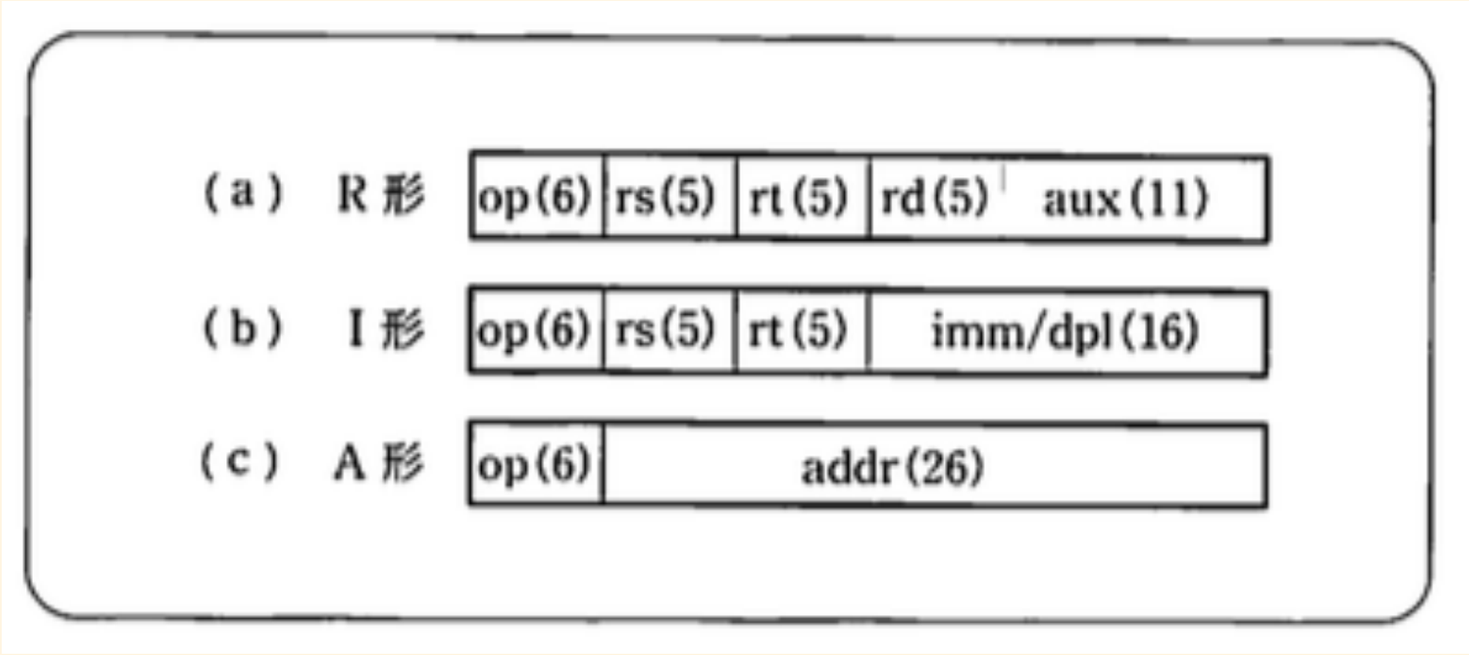
命令の表現形式

- 命令は一語で固定長。通常命令語32ビット
- 命令形式は、R, I, A形の3種類のみ
- 命令を極力単純にする => デコードにかかる時間を減らす



命令フィールド

- 命令語も、各フィールドも固定長
- 各フィールドの大きさ
    - op: log_2 (対象コンピュータの命令セットの大きさ)
    - rs, rt, rd: log_2 (レジスタファイルに含まれるレジスタの数)
    - aux, imm/dpl, addd: (命令長) - (他のフィールドのビット数の総和)



命令フィールド

- フィールドが固定長
- 各命令フィールドのビットのうち、operandの構成をoperationを読んだ時点で判断できる！
  - opが add (000000) なら、R形だ
  - opが subi (000001) なら、I形
  - opが j (110110) なら、 A形

アセンブリ言語とは

- より自然言語(英語)に近い記号で表される命令
- 機械語の命令と一対一で対応する！
- (後半で具体例とともに解説)

命令セット

いくつか具体的な
命令について説明します

算術論理演算命令

- レジスタ間の演算 or レジスタと即値との間の演算 => R形かI形

表 3.1　算術演算命令

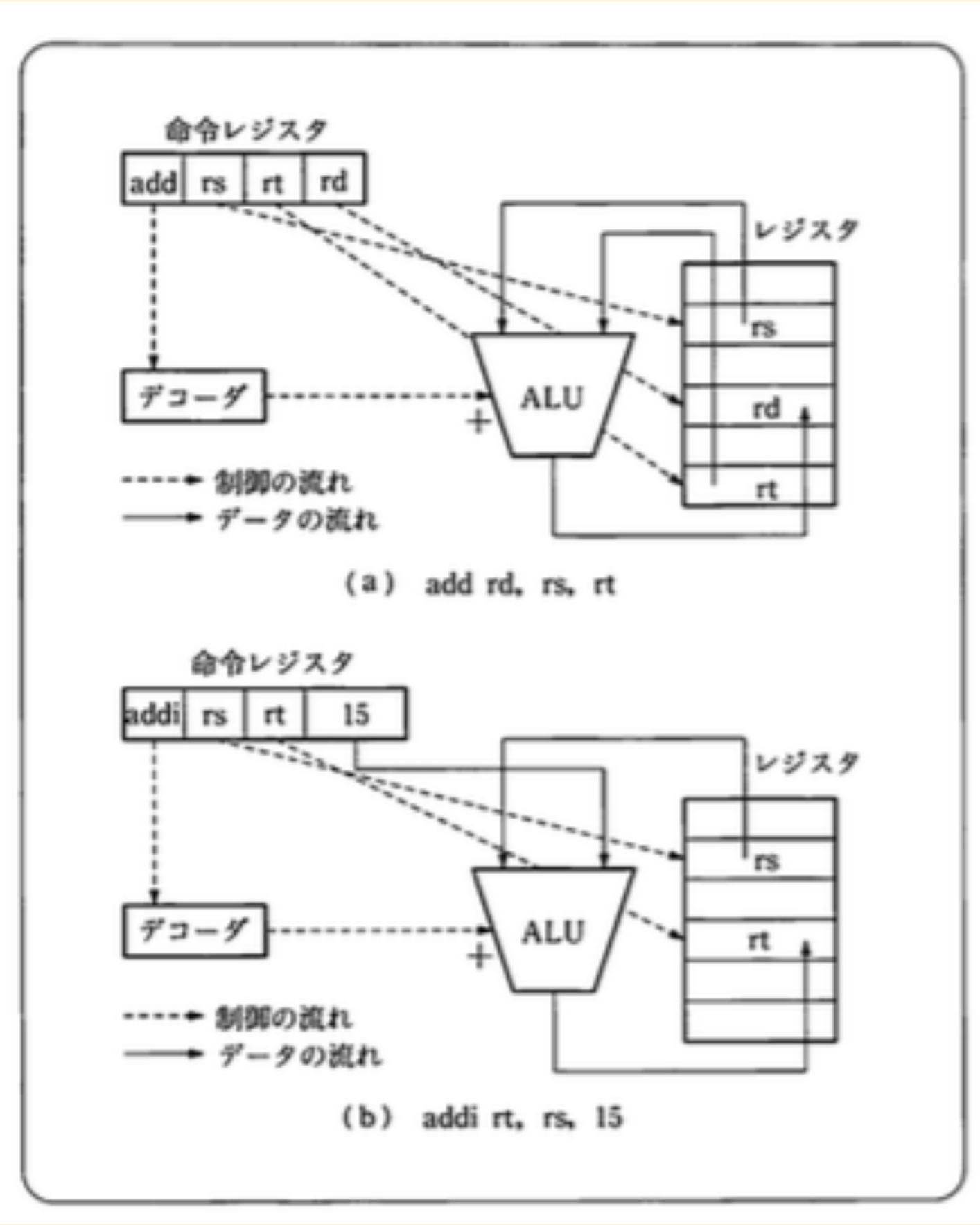
演算命令	整数演算命令		浮動小数点演算命令
	R形	I形	R形
加　算	add	addi	fadd
減　算	sub	subi	fsub
乗　算	mul	muli	fmul
除　算	div	divi	fdiv
剰　余	rem	remi	——
絶対値	abs	——	fabs
算術左シフト	sla	——	——
算術右シフト	sra	——	——

表 3.2　論理演算命令

演算命令	R形	I形
論理積	and	andi
論理和	or	ori
否　定	not	——
NOR	nor	nori
NAND	nand	nandi
排他的論理和	xor	xori
EQUIV	eq	eqi
論理左シフト	sll	——
論理右シフト	srl	——

算術論理演算命令の動作例

- 他の命令もALUへの制御命令を変えることで実現できる
- mul、div、シフト、小数点演算についてはALUと別に専用のユニットを設けるのが普通

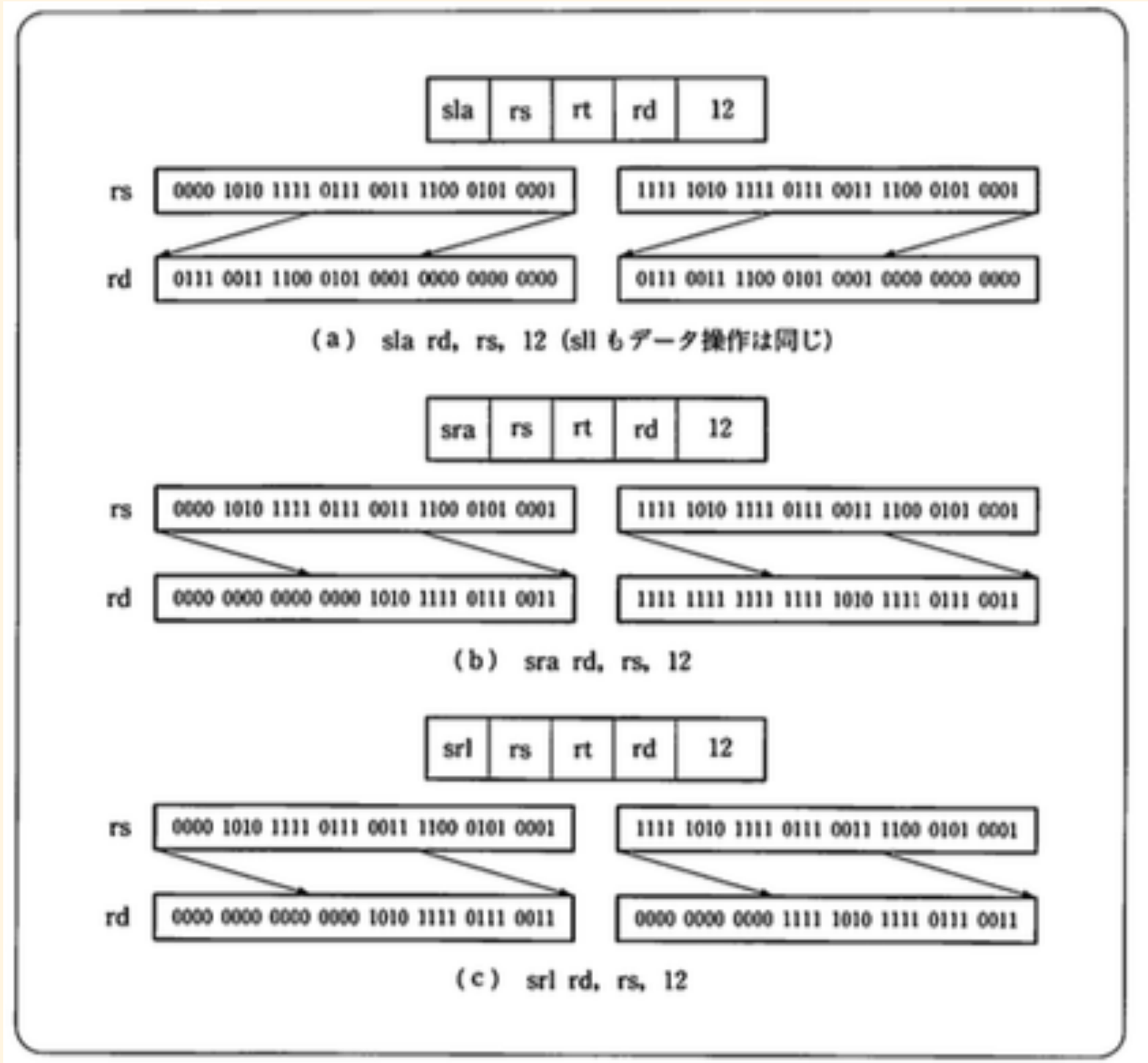


シフト命令

- 右シフト
    - 論理シフト: 空いた上位ビットに0を突っ込む
    - 算術シフト: もとのデータの符合を入れる
- 左シフト:

```
sla rs rt rd 12
```

シフト命令



(a) sla rd, rs, 12 (sll もデータ操作は同じ)

(b) sra rd, rs, 12

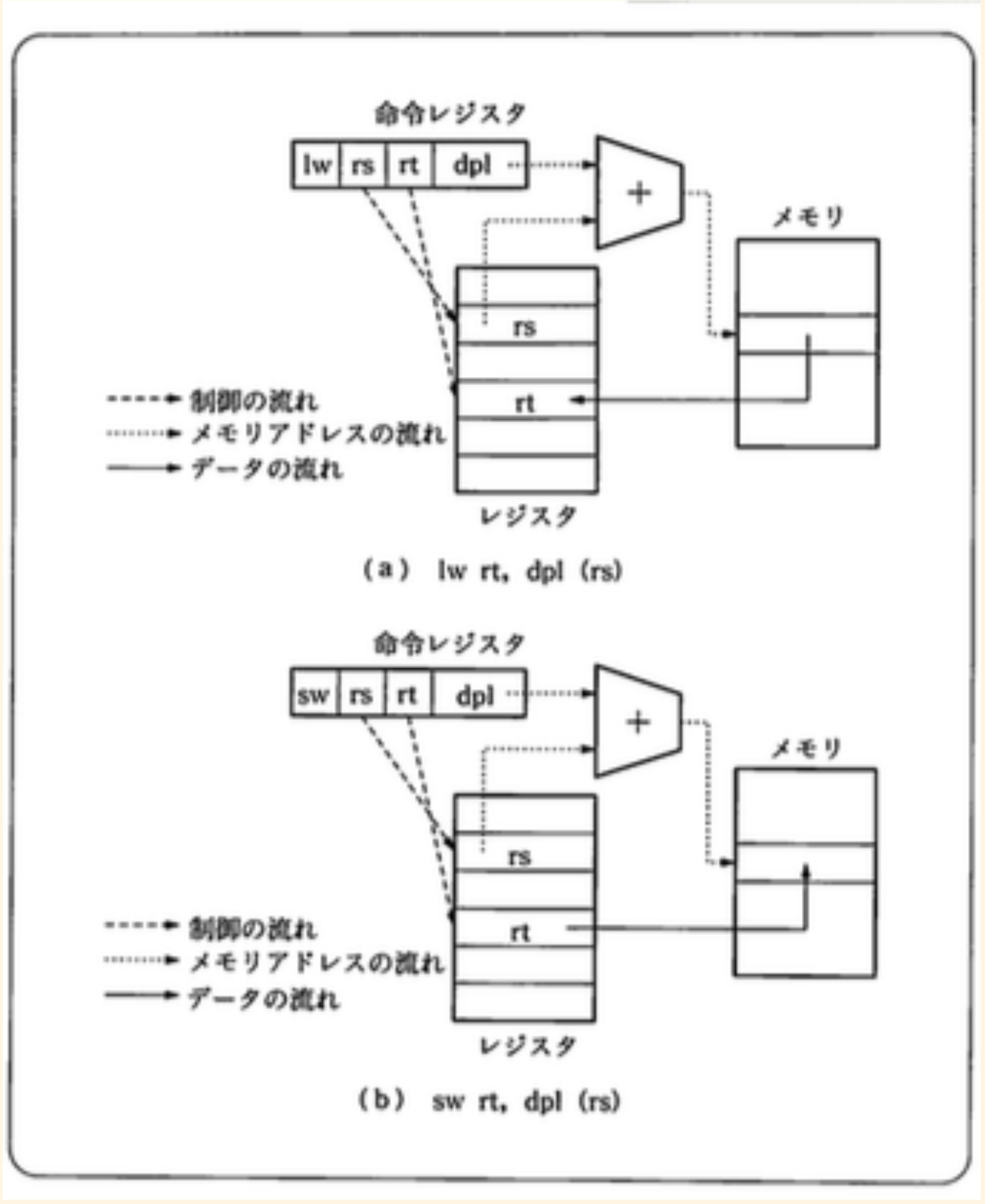
(c) srl rd, rs, 12

データ移動命令

- メモリとレジスタの間のデータ移動
  - メモリ => レジスタ
    - lw ... load word (32 bits)
  - レジスタ => メモリ
    - sw ... store word (32 bits)

```
sla rs rt rd 12
```

データ移動命令の動作


命令レジスタ
lw
rs
rt
dpl
+
メモリ
rs
rt
制御の流れ
メモリアドレスの流れ
データの流れ
レジスタ
(a) lw rt, dpl (rs)
命令レジスタ
sw
rs
rt
dpl
+
メモリ
rs
rt
制御の流れ
メモリアドレスの流れ
データの流れ
レジスタ
(b) sw rt, dpl (rs)

分岐命令 (無条件編)

- j ... jump: アドレスを指定してジャンプする (ラベルを指定する使い方がメインな気がする)
- jr ... jump register: 指定したレジスタの内容にジャンプする
- jal ... jump and link: ジャンプの前に特定レジスタに現在のプログラムカウンタの次の番地を入れておく => サブルーチンのときとか場所を保存しておく

```
sla rs rt rd 12
```

分岐命令 (条件編)

- beq ... branch on equal: rs == rt ならジャンプ
- bne ... branch on not equal
- blt ... branch on less than
- ble ... branch on less than or eq.

```
beq rs, rt, dpl
```

アドレシング

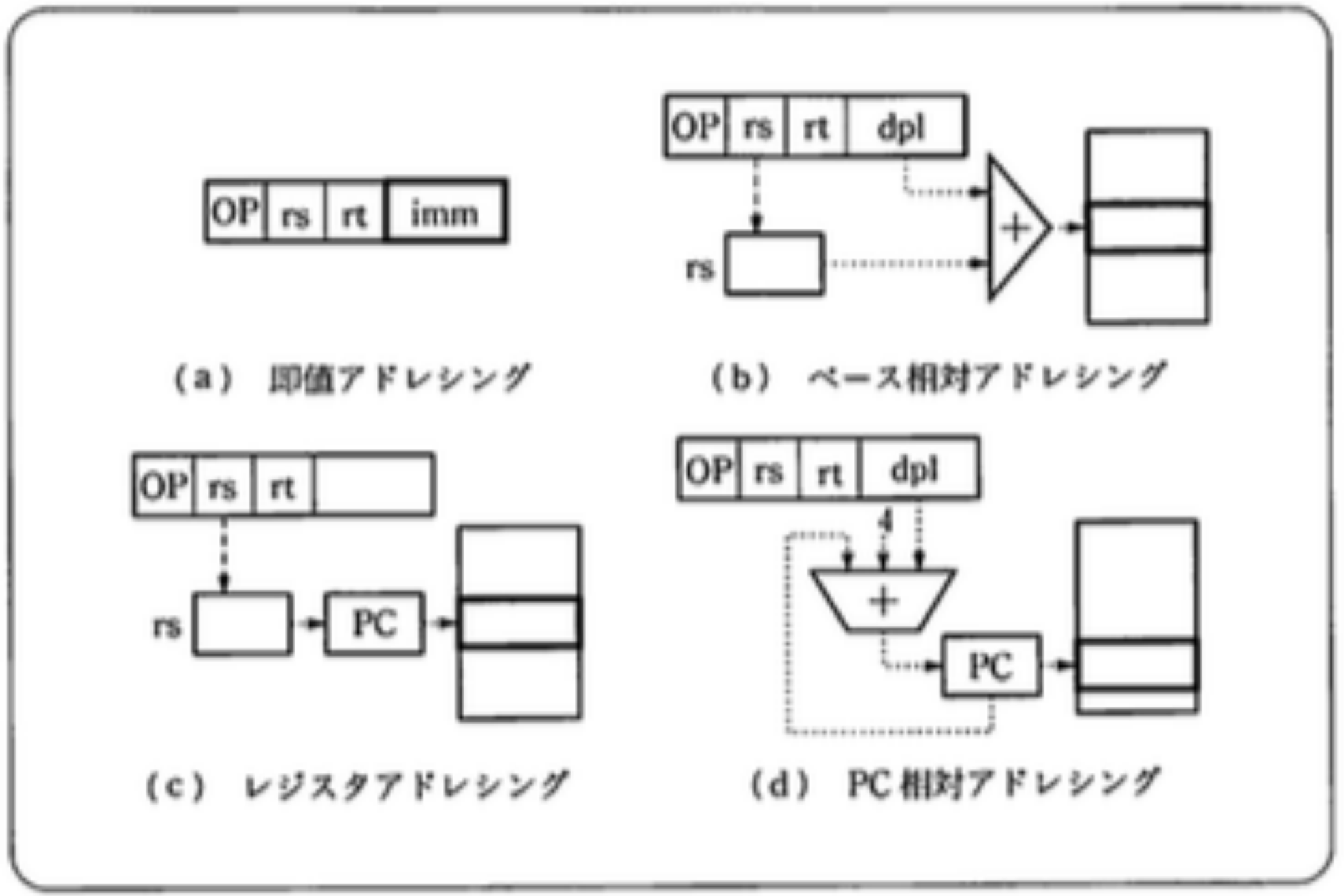
アドレシングとは

- データや命令の居場所を特定すること
- 命令からメモリの番地を生成したり

アドレシングの種類

- 即値アドレシング
  - addi rt, rs, imm
    - 直接値immを生成
- ベース早退アドレシング lw
  - lw rt, dpl (rs)
  - rs + dpl
- レジスタアドレシング j
  - j rs
    - rs
- PC相対アドレシング
  - beq rs, rt, dpl
  - (pc) + 4 + dpl (分岐するとき)

アドレシングの種類


OP
rs
rt
imm
(a) 即値アドレシング
OP
rs
rt
dpl
rs
+
(b) ベース相対アドレシング
OP
rs
rt
rs
PC
(c) レジスタアドレシング
OP
rs
rt
dpl
4
+
PC
(d) PC相対アドレシング

アドレシングの種類

- メモリアドレス 30ビットで、1GBのメモリのアドレシングができる
- ビッグエンディアン … データの上位バイトから並べる
- リトルエンディアン … データの下位バイトから並べる

ゼロレジスタと定数の生成

- ゼロレジスタ ... $zero とかr0で参照される
- 定数を生成するときに使われる

```
addi r1, r0, 28
```

サブルーチンの実現

サブルーチンとは

- よく使われるプログラムの部分をまとめて切り出したもの
- Cでいう関数とかのこと

サブルーチンの手順

1. レジスタ値の退避
2. 戻り番地 (次の命令番地) の退避
3. サブルーチンの先頭番地へジャンプ
4. サブルーチン本体の実行
5. 戻り番地へのジャンプ
6. レジスタ値の復帰
7. もとの命令形の実行再開

メモリの退避方法

- Caller Save Method … レジスタ値を呼出し側で退避する
- Calle Save Method … レジスタ値を呼び出された側が退避する方式

スタックで実現

- 退避領域は、データメモリの中にスタックの構造(LIFO)で保存されている
- スタックポインタからフレームのサイズを引いて、フレーム用にメモリを割り当てる
- $spはスタックポインタ専用のレジスタ (って規約)

スタックで実現

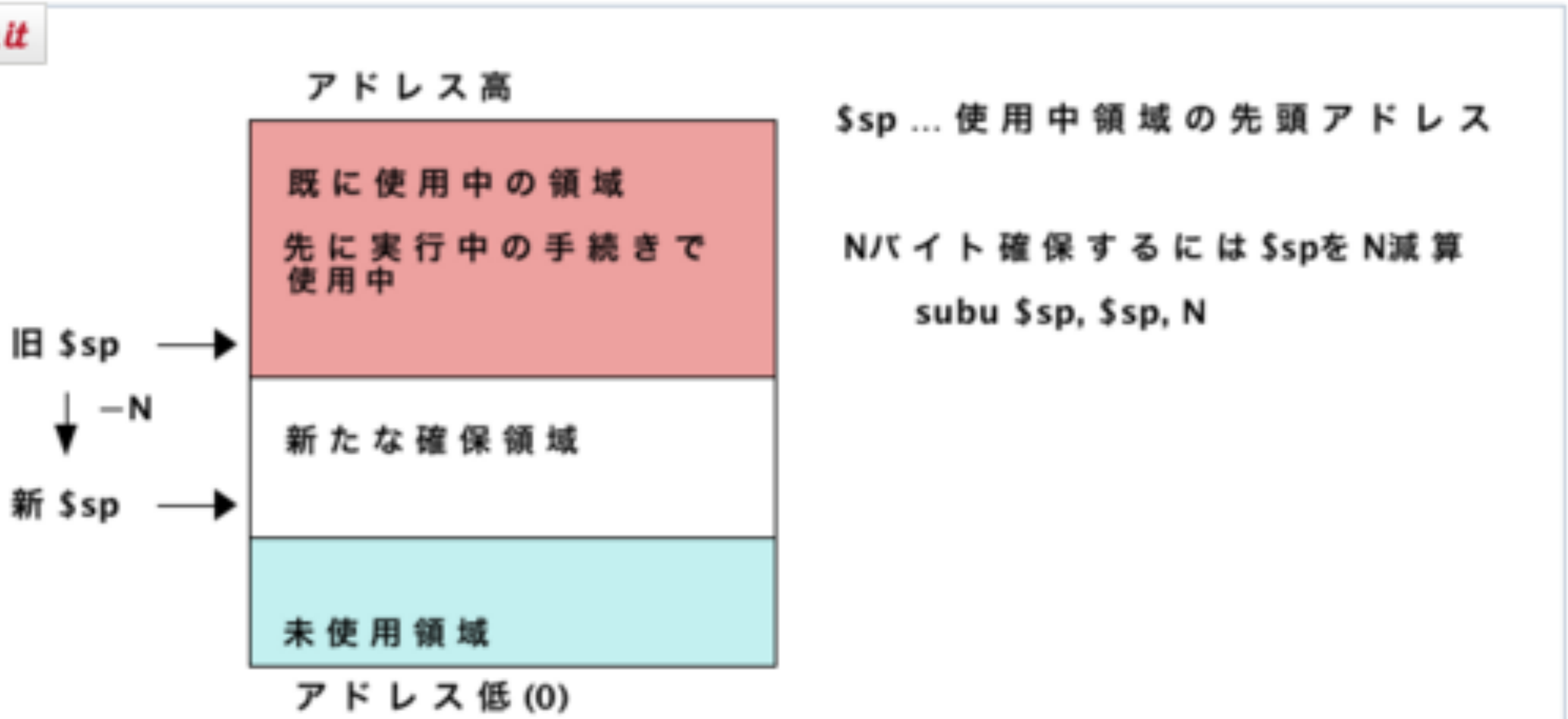


Figure 1: スタックを確保している様子

サブルーチンの例

3.F　サブルーチンのアセンブラプログラム

```
sw r 1，0(sp)     ：   レジスタ値の待避（必要なだけ）始め
sw r 2，4(sp)
……
sw rk，4k(sp)     ：   レジスタ値の待避終わり
add sp，4k＋4
jal address
sub sp，4k＋4
lw r 1，0(sp)     ：   レジスタ値の復帰始め
lw r 2，4(sp)
……
lw rk，4k(sp)     ：   レジスタ値の復帰終わり
元の仕事の続き
……
```

```
address：……      サブルーチン本体
         ……
         ……
         jr r 31
```

談話室

CISCとRISC

- CISC (complex instruction set computer)
  - **命令セットを大きくして、一つの命令でいろんなことを実現することで効率あげよう！**
    - **IBM汎用コンピュータ、VAX-11、Intel 80486**
- RISC (reduced instruction set computer)
  - **必要最小限に、シンプルな命令セットにして命令実行時間 (MIPS) を上げよう！**
    - Sparc, MIPS, Power PC, PA-RISC, Pentium

CISCとRISC

- RISCの勝利！
  - よく使う命令を重点的に高層化するという設計思想の正しさ
  - コンパイラとの連携に適していた
  - マイクロプロセッサの高集積化・高性能化との相性

まとめ

まとめ

- 命令セット ... コンピュータの命令の集まり
- 命令の表現形式 ... 命令の2進数表現の形式、フィールドで区切られる。R形、I形、A形に分類
- アセンブリ言語 ... 機械語をより自然言語に近い表現に
- 算術論理演算命令 ... レジスタ間またはレジスタと即値の間で演算がされる。結果はレジスタに格納される
- 分岐命令 ... 制御の流れを変更する。条件付 or 無条件
- アドレシング ... メモリアドレスの生成方法、即値アドレシングなど
- サブルーチン ... プログラムを再利用可能な形にしたもの。スタックを用いた規約手続きに従う必要あり