# 算術論理演算II
# ～乗除算・浮動小数点演算～

(株)日立製作所　マイクロデバイス事業部
Micro Device Division, Hitachi Ltd.

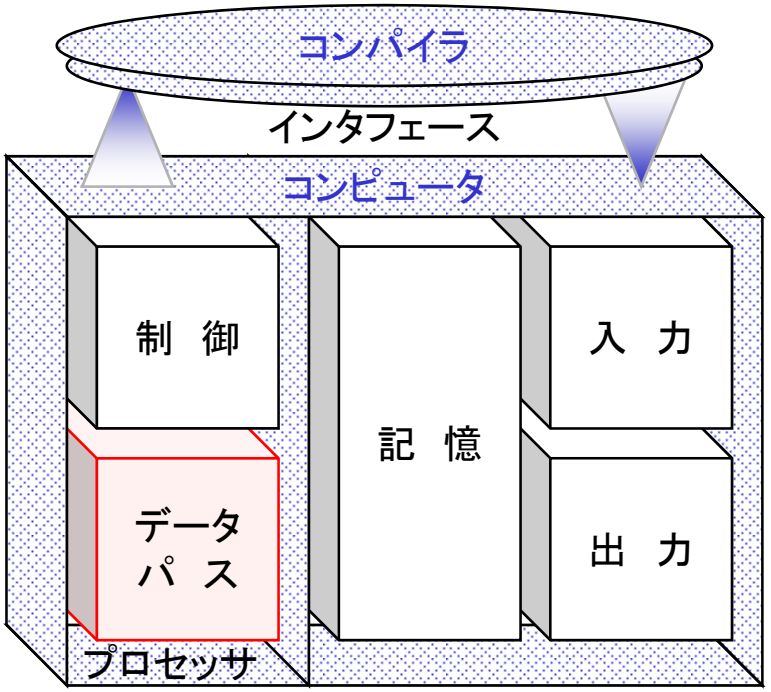
川下　　達也
Tatsuya Kawashimo

# 本日の授業の内容

① 乗算

② 除算

③ 浮動小数点とその演算



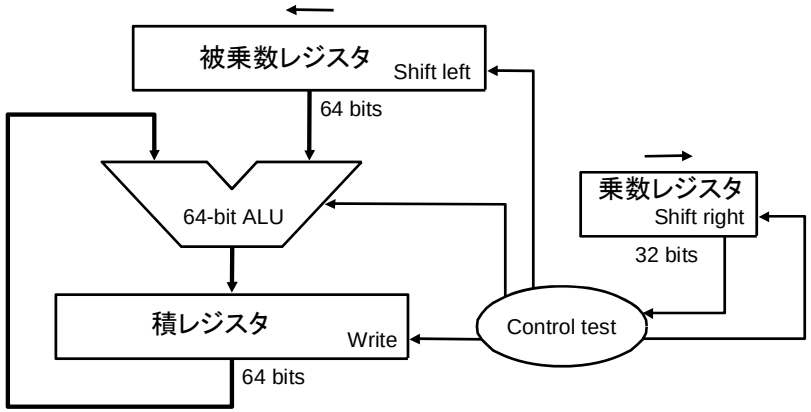
# 乗算

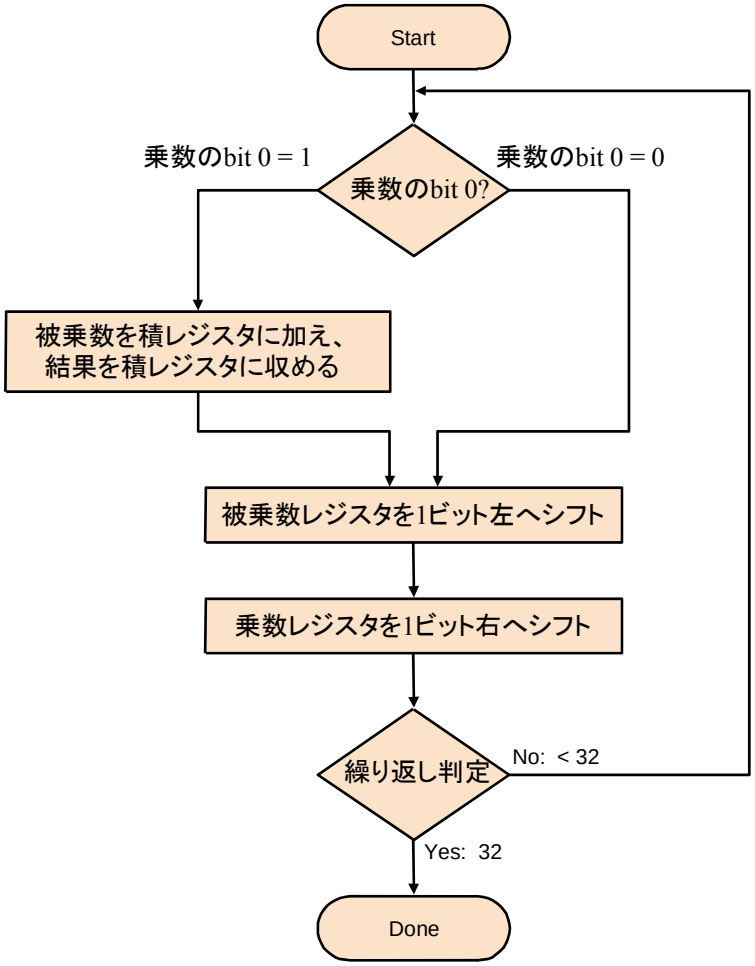
- 加算より複雑
  シフトと加算を使って乗算を実現

```
      0010  ・・被乗数 (multiplicand)
×)    0011  ・・乗数    (multiplier)
-------------
      0010
     0010
    0000
   0000
-------------
   0000110  ・・積 (product)
```

- 取り上げた乗数の数字が1の場合は、被乗数の値を適切な位置に記入
- 取り上げた乗数の数字が0の場合は、0を記入

# 乗算アルゴリズムとハードウエアの第1のバージョン

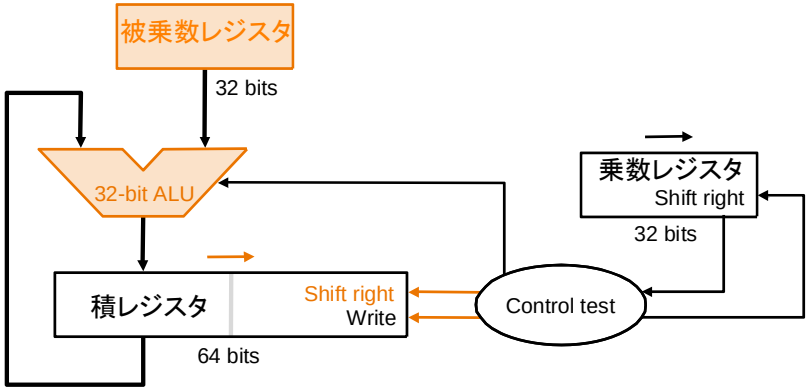


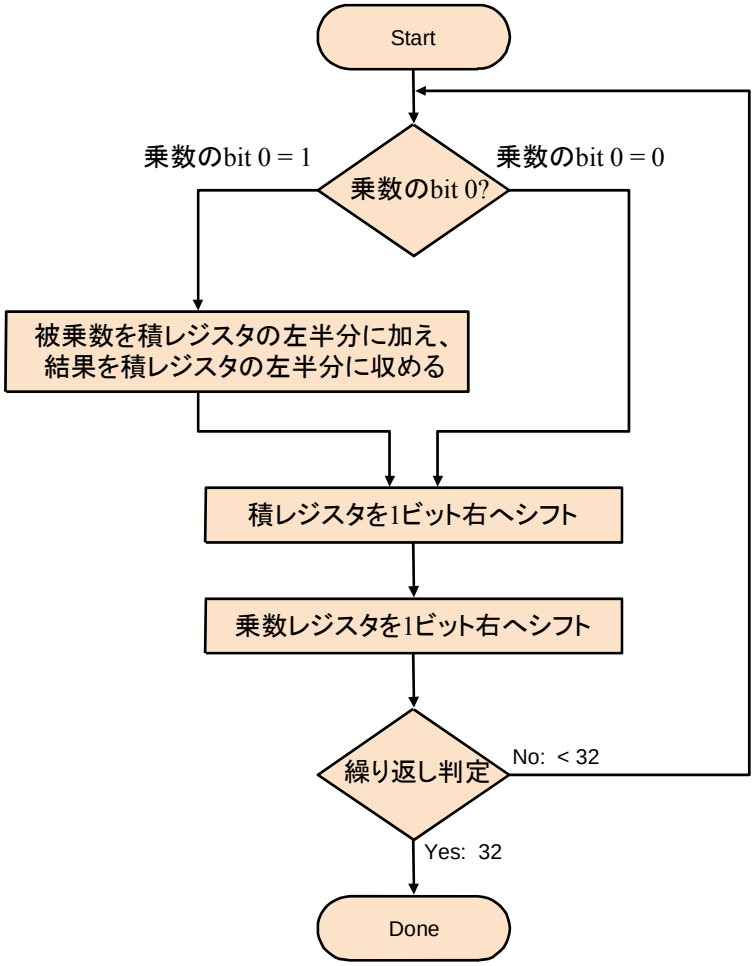




例題） 0010 x 0011

| Step | 処理 | 乗数 | 被乗数 | 積 |
|---|---|---|---|---|
| 0 | 初期値 | 0011 | 0000 0010 | 0000 0000 |
| 1 | 加算 | 0011 | 0000 0010 | 0000 0010 |
| | シフト | 0001 | 0000 0100 | 0000 0010 |
| 2 | 加算 | 0001 | 0000 0100 | 0000 0110 |
| | シフト | 0000 | 0000 1000 | 0000 0110 |
| 3 | シフト | 0000 | 0001 0000 | 0000 0110 |
| 4 | シフト | 0000 | 0010 0000 | 0000 0110 |

# 第2のバージョン



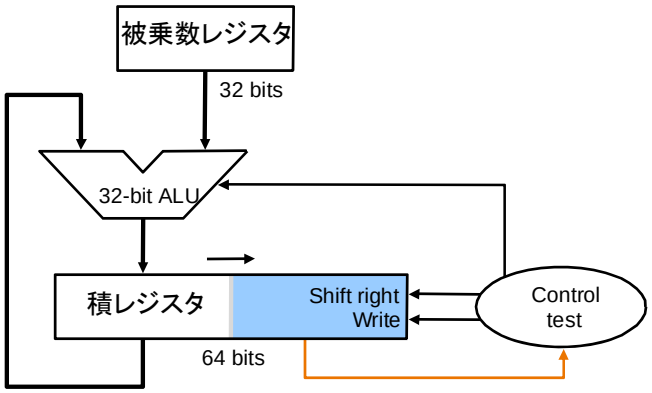



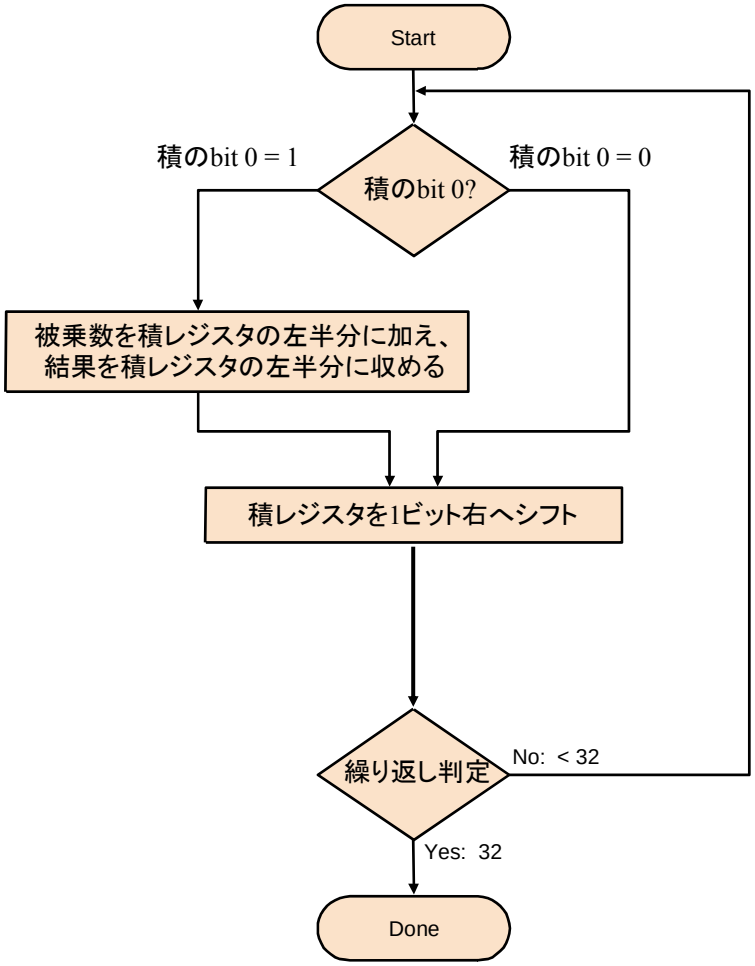

例題） 0010 x 0011

| Step | 処理 | 乗数 | 被乗数 | 積 |
|---|---|---|---|---|
| 0 | 初期値 | 0011 | 0010 | 0000 0000 |
| 1 | 加算 | 0011 | 0010 | 0010 0000 |
| | シフト | 0001 | 0010 | 0001 0000 |
| 2 | 加算 | 0001 | 0010 | 0011 0000 |
| | シフト | 0000 | 0010 | 0001 1000 |
| 3 | シフト | 0000 | 0010 | 0000 1100 |
| 4 | シフト | 0000 | 0010 | 0000 0110 |

# 最終バージョン

積レジスタの右半分を乗数レジスタとする




例題） 0010 x 0011

| Step | 処理 | 被乗数 | 積 |
|---|---|---|---|
| 0 | 初期値 | 0010 | 0000 0011 |
| 1 | 加算 | 0010 | 0010 0011 |
| | シフト | 0010 | 0001 0001 |
| 2 | 加算 | 0010 | 0011 0001 |
| | シフト | 0010 | 0001 1000 |
| 3 | シフト | 0010 | 0000 1100 |
| 4 | シフト | 0010 | 0000 0110 |

# Boothのアルゴリズム

Andrew D. Booth, “A signed binary multiplication technique” (1951)

Boothのアルゴリズム
乗数の1が連続するビット列で
開始ビット”1”で減算
途中ビット”1”で加減算せず
終了ビット “1”の次のビット “0”で加算

0010 x 0110 = 0010 x (−0010 + 1000)

連続した1の場合に高速
0110を$2^n$の和に変換

他にも
Wallace Treeなどの
方式あり

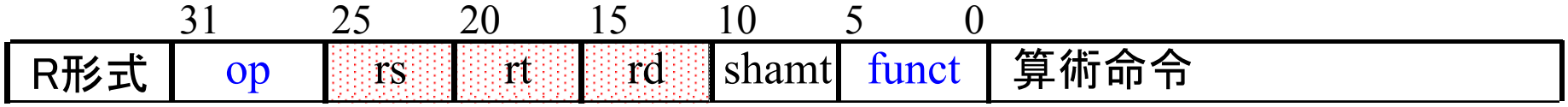
```
      0010 ‥被乗数
×)    0110 ‥乗数
------------
      0000
−     0010
      0000
+     0010
------------
  00001100 ‥積
```

一般には、
乗数の1のビットで加算

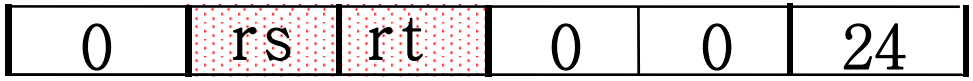
```
      0010 ‥被乗数
×)    0110 ‥乗数
------------
      0000
      0010
      0010
      0000
------------
  00001100 ‥積
```

# MIPSにおける乗算

## R (Register)形式

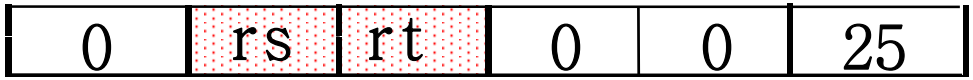



(1) mult rs, rt




Hi, Lo レジスタに符号付き64bitの積が格納

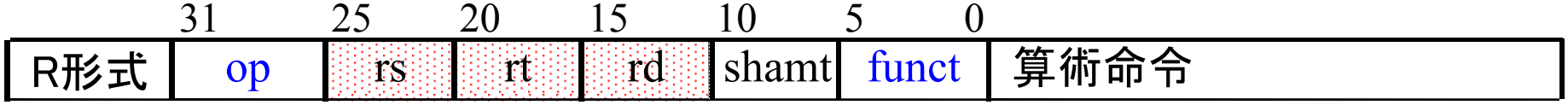
(2) multu rs, rt



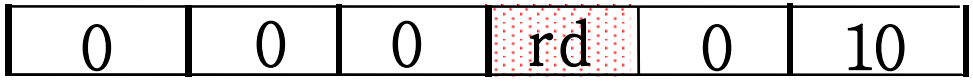

Hi, Lo レジスタに符号なし64bitの積が格納

# Hi, Loレジスタ

**Hi, Loレジスタ：乗算結果(64bit)を格納。特殊レジスタ**

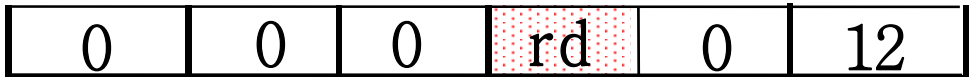



(1) mfhi rd (move from hi)




**Hiレジスタの値をrdレジスタへ移送**

(2) mflo rd (move from lo)




**Loレジスタの値をrdレジスタへ移送**

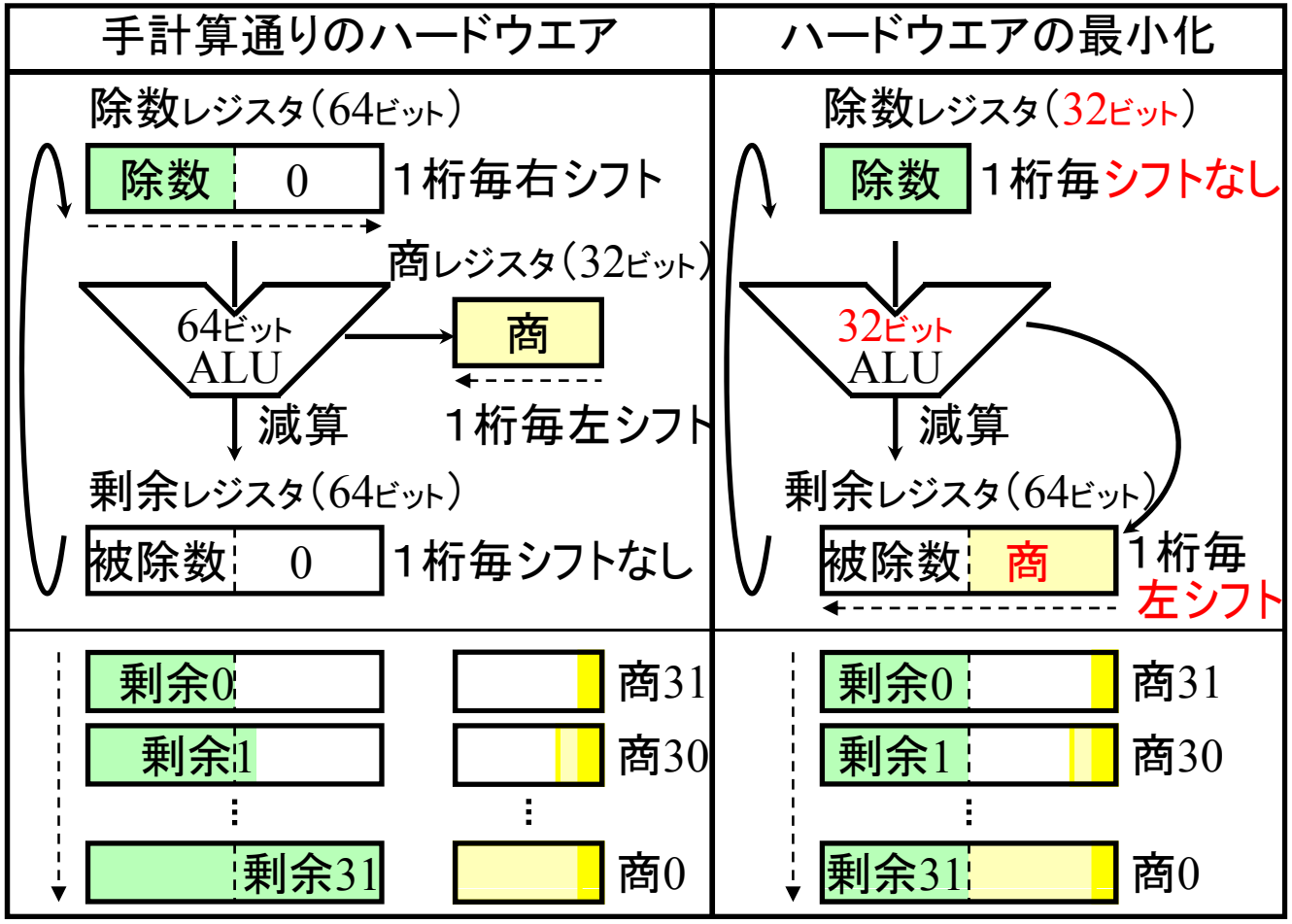
(3) mthi rd (move to hi)

(4) mtlo rd (move to lo)

# 除算

## コンピュータでの除算

被除数から除数を減算し、剰余＞0なら商の第0ビット=1




## 符号付き除算

剰余の符号＝被除数の符号

商の符号＝被除数と除数が

同符号なら正

異符号なら負

(+7) ÷ (+2)=+3 余り+1

(+7) ÷ (-2)=-3 余り+1

(-7) ÷ (+2)=-3 余り-1

(-7) ÷ (-2)=+3 余り-1

高速化手法

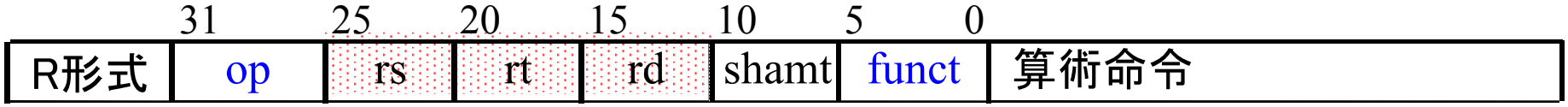
・SRT法

・Newton-Raphson

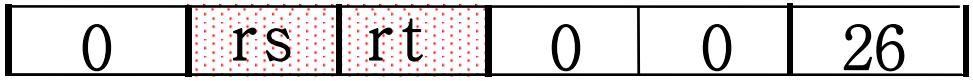
など

# MIPSにおける除算

## R (Register)形式

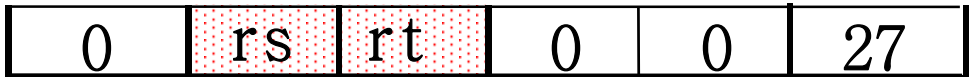



(1) div rs, rt




Hiに剰余 Lo レジスタに商を格納。符号付き演算

(2) divu rs, rt




符号なしの除算

# 浮動小数点形式

- 符号付き整数以外の数を表現する方法？
- 浮動小数点形式

  符号(sign)、指数部(exponent)、仮数部(significant)

  $(-1)^S \times$ 仮数部 $\times 2^{\text{指数部}}$

# IEEE 754浮動小数点規格

単精度

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 | F19 | F20 | F21 | F22 | F23 |

倍精度

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 | F19 | F20 |
| F21 | F22 | F23 | F24 | F25 | F26 | F27 | F28 | F29 | F30 | F31 | F32 | F33 | F34 | F35 | F36 | F37 | F38 | F39 | F40 | F41 | F42 | F43 | F44 | F45 | F46 | F47 | F48 | F49 | F50 | F51 | F52 |

- 単精度(single precision)　浮動小数点形式 -> 32bit
- 倍精度(double precision)　浮動小数点形式 -> 64bit

符号 指数部 仮数部

- 仮数部の先頭の1は、Fnに表現されない（暗黙の1）
- 仮数を1.xxxxxx (2進数)で表現 -> 正規化数
- 指数部にゲタ(bias)、ゲタは単精度で127, 倍精度で1023

指数部はunsigned number!!

$$( -1 )^{S} \times (1 + F1 \times 2^{-1} + F2 \times 2^{-2} + F3 \times 2^{-3} + \ldots\ldots) \times 2^{(\text{指数部} - \text{bias})}$$

# 浮動小数点で扱う数

※ 単精度の場合 (bias=127)

| 項目 | 符号 | 指数 (exp) | 仮数 (frc) | 意味 |
|---|---|---|---|---|
| 正規化数 | S | 00…01～<br>11…10<br>(1 ～ 254) | frc | $(-1)^S \times (1.frc) \times 2^{(exp-bias)}$ |
| 非正規化数 | S | 00…00 (0) | frc (≠0) | $(-1)^S \times (0.frc) \times 2^{(1-bias)}$ |
| 0 | S | 00…00 (0) | All 0 | +0, -0 の2種類 |
| 無限大 | S | 11…11 (255) | All 0 | +∞, -∞の2種類 |
| NaN<br>(Not a Number) | S | 11…11 (255) | frc (≠0) | 無効演算を行った時の結果<br>意図的に割込を入れたい場合 |

◎例外

- オーバーフロー：指数部が上限を超えた場合
- アンダーフロー：指数部が正規化数の下限を超えた場合
- ゼロ除算：非0の数を0で割った場合
- 無効演算：∞×0、∞－∞、∞÷∞、0÷0、負数の平方根など

# 下図の単精度浮動小数点を10進数に変換

| S | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 | F19 | F20 | F21 | F22 | F23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$(-1)^1 \times (1 + 0.25) \times 2^{(129-127)}$$

$$= -1.25 \times 2^2$$

$$= -1.25 \times 4$$

$$= -5.0$$

# 数値$-0.75_{10}$を単精度浮動小数点で表現

$$
\begin{aligned}
-0.75_{10} &= -(0.5 + 0.25)_{10} \\
&= -0.11_2 \times 2^0 \\
&= -1.1_2 \times 2^{-1} \\
&= (-1)^1 \times 1.1_2 \times 2^{(126-127)}
\end{aligned}
$$

| S | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 | F19 | F20 | F21 | F22 | F23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

126

# 浮動小数点加算

| | | | |
|---|---|---|---|
| F1： | S1 | EXP1 | FRC1 |
| F2： | S2 | EXP2 | FRC2 |

(1)桁合わせ：指数部が同じになるように仮数部をシフト

$F1=(-1)^{S1} \times (1.FRC1) \times 2^{(EXP1 - bias)}$

$F2=(-1)^{S2} \times (1.FRC2) \times 2^{(EXP2 - bias)}$

$=(-1)^{S2} \times (0.0..01FRC2) \times 2^{(EXP1 - bias)}$

→ (EXP1−EXP2)桁シフト（EXP1>EXP2の場合）

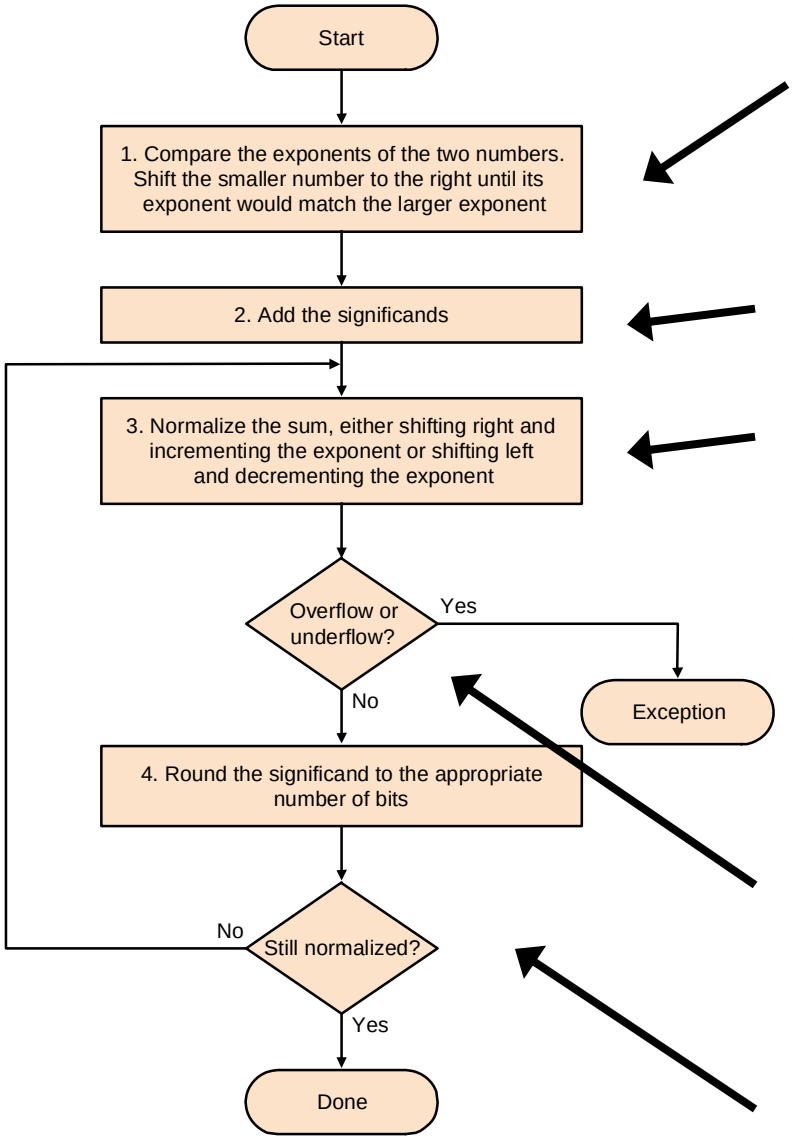
(2)仮数部の加算（符号が異なる場合は減算）

(3)正規化・まるめ

$F3=(-1)^{S3} \times (0.0..01FRC3) \times 2^{(EXP3 - bias)}$

$=(-1)^{S3} \times (1.FRC3) \times 2^{(EXP3 - SFT - bias)}$

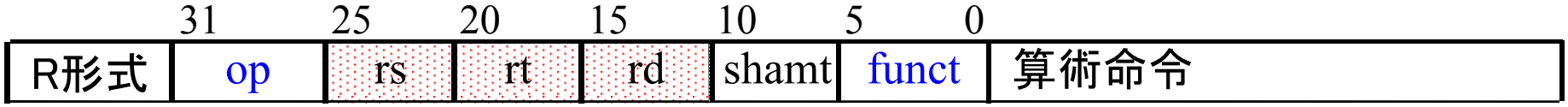
# 浮動小数点加算



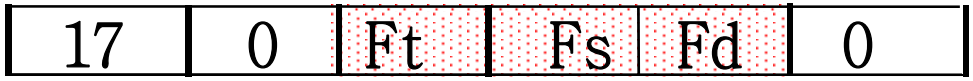

- 2つの数値の指数を比較。小さい方の数値を右にシフトして、大きい方と指数が一致するように調整する
- 仮数を加える
- 和を正規化する。① 右にシフトして指数を大きくするか、(結果の数が大きい)② 左にシフトして指数を小さくするか(引算で結果の数が小さくなった場合)
- 指数を変化させた時は、オーバーフロー、アンダーフローしていないかを確認。
- 丸める -> 正規化のままか？

# MIPSにおける浮動小数点演算

## R (Register)形式

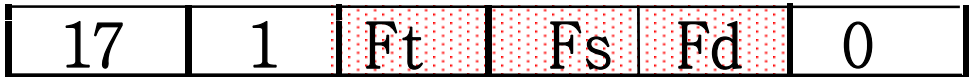



(1) add.s Fd, Fs, Ft




単精度の加算 Fd = Fs + Ft

(2) add.d Fd, Fs, Ft




倍精度の加算 Fd = Fs + Ft

$f0 - $f31
32bitのレジスタ
倍精度の場合は
2つ一組で使う。
$f0, $f2の
ように、偶数レジ
スタで指定