

**NEC**

浮動小数点演算サンプルソフト

# 使用法説明書（78K0S シリーズ）

## ご注意

本ソフトウェアはあくまで参考用のソフトであり、当社がこの動作を保証するものではありません。本ソフトウェアを使用する場合、お客様のセット上で十分な評価の上ご使用いただきますようお願いいたします。

改版履歴

版数	作成年月日	記事
第一版	平成14年 7月25日	第一版作成

## 目次

1 . 概要 .....	P 4
2 . 浮動小数点形式 (RAM上のデータの格納形式) .....	P 5
2 . 1 数値の管理構成 .....	P 5
2 . 2 浮動小数点レジスタ .....	P 7
2 . 3 浮動小数点形式の各部の詳細 .....	P 8
2 . 4 浮動小数点レジスタへの格納例 .....	P 1 0
3 . 四則演算 .....	P 1 1
3 . 1 浮動小数点加算 (SFPADD) .....	P 1 2
3 . 2 浮動小数点減算 (SFPSUB) .....	P 1 5
3 . 3 浮動小数点乗算 (SFPMULT) .....	P 1 7
3 . 4 浮動小数点除算 (SFPDIV) .....	P 2 6
4 . その他のルーチン .....	P 3 4
4 . 1 正規化 (SNML) .....	P 3 4
4 . 2 仮数部加算 (SADDBCD) .....	P 3 4
4 . 3 仮数部減算 (SSUBBCD) .....	P 3 4
4 . 4 レジスタ交換 (SEXCHXW) .....	P 3 5
4 . 5 レジスタ・アップ・シフト (SSHUPX) .....	P 3 5
4 . 6 レジスタ・ゼロ・クリア .....	P 3 5
5 . RAM説明 .....	P 3 6
5 . 1 RAMおよびフラグ説明 .....	P 3 6
6 . フローチャート .....	P 3 7
6 . 1 浮動小数点加算 / 減算 .....	P 3 7
6 . 2 浮動小数点乗算 .....	P 3 9
6 . 3 浮動小数点除算 .....	P 4 1

## 1 . 概要

本説明書では、78K0Sシリーズ共通で使用できる浮動小数点演算プログラムを説明します。

以下の順序で説明します。

浮動小数点形式

四則演算

その他のルーチン

R A M説明

フローチャート

## 2. 浮動小数点形式 (RAM上のデータの格納形式)

### 2.1 数値の管理構成

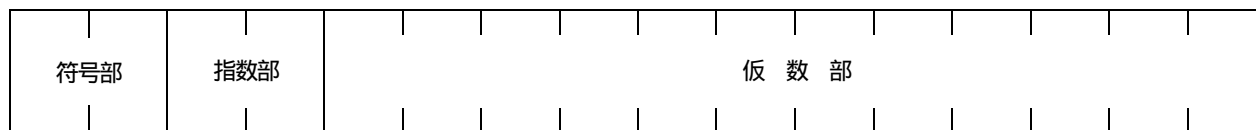
この演算パッケージでは、被演算データ、演算データ、および演算結果を、図1に示すような浮動小数点付き10進数で扱います。

各演算データは、正負の符号を表す符号部(1ビット)、小数点位置( $10^n$ の $n$ )を格納する指数部(8ビット)、および、符号も小数点も付いていない10進数を扱う仮数部(12バイト)に分けて管理します。

仮数部は命令効率を優先し、1桁に1バイト(上位4ビットは0固定)を割当てています。

以下の説明では上位4ビットの0は、特に必要のない限り明記しないものとします。

図1 数値の管理構成



符号部 : 1ビット (8ビットの最下位ビット)

指数部 (16進 2桁): 2 × 4ビット

仮数部 (10進 12桁): 12 × 8ビット (下位4ビット使用, 上位4ビットは0固定) 整数部 1桁  
 小数部 11桁

この数値の管理構成を式で表すと、次のようになります。

(指数部値)	$(-1) \text{ (符号部値)} \times \text{ (仮数部値)} \times 10$
--------	---

## 2.2 浮動小数点レジスタ

この演算パッケージでは、浮動小数点付き10進数を扱うレジスタとして、3つの浮動小数点レジスタを用意しています。図2に、浮動小数点レジスタのRAM配置イメージを示します。

図2 浮動小数点レジスタのRAM配置イメージ



この演算パッケージを使用するときは、REGXに被演算数を、REGYに演算数をセットした後に、各演算処理を呼び出します。各演算処理は、REGXに演算結果を格納して終了します。

各レジスタ間のデータの流れの詳細については、2. 四則演算 を参照してください。

## 2.3 浮動小数点形式の各部の詳細

この項では、浮動小数点付き10進数の各部の詳細について説明します。

### (1) 仮数部

仮数部は有効数字の部分を、整数部1桁と小数部11桁の合計12桁で表現します。仮数部には、正規化された10進数の絶対値が格納されます。

正規化とは、仮数部をあらかじめ定められた範囲内に収めるために、指数部と仮数部を調整することです。この演算パッケージでは、数値が0の場合を除いて、仮数部の範囲を  $1 \leq \text{仮数部} < 10$  としていますので、正規化を行うと整数部に“0”以外の数が格納されるように指数部が調整されます。

表1に、正規化の例を示します。

表1 正規化

正規化される前の値	正規化された値
3729.45	$3.72945 \times 10^3$
0.8765	$8.765 \times 10^{-1}$
0.00054	$5.4 \times 10^{-4}$

(2) 指数部

指数部は、表現する数値の底10の指数を、2桁の16進数で構成しています。  
負の場合は2の補数表現になります。

指数部と指数値の対応は、図3の通りです。

図3 指数部と指数値の対応

指数値 ...	$10^7$	$10^6$ ...	$10^2$	$10^1$	$10^0$	$10^{-1}$	$10^{-2}$ ...	$10^{-5}$	$10^{-6}$
指数部 ...	07	06 ...	02	01	00	FF	FE ...	FB	FA

備考 2の補数とは、n桁の2進数に対し、 $2^n$ からその数を引いた数をいいます。

(3) 符号部

数値の符号は1ビットで表現します。

正数の場合は、符号に対応するビットをリセット(0)、負数の場合はセット(1)します。



### 3 . 四則演算

この演算パッケージでは、四則演算関数として、次の4つを用意しました。

( 1 ) 浮動小数点加算 ( S F P A D D )

REG Xの値を被加数, REG Yの値を加数として加算を行います。

( 2 ) 浮動小数点減算 ( S F P S U B )

REG Xの値を被減数, REG Yの値を減数として減算を行います。

( 3 ) 浮動小数点乗算 ( S F P M U L T )

REG Xの値を乗数, REG Yの値を被乗数として乗算を行います。

( 4 ) 浮動小数点除算 ( S F P D I V )

REG Xの値を被除数, REG Yの値を除数として除算を行います。

演算結果は、各演算処理の終了時にREG Xに格納されます。

### 3.1 浮動小数点加算 (SFPADD)

#### (1) 処理内容

REGXとREGYの浮動小数点数を10進加算し、結果を正規化してREGXに格納します。

#### (2) 入力条件

REGX, REGYにそれぞれ正規化された被加数, 加数を格納します。

#### (3) 出力結果

・REGXには正規化された演算結果を格納しています。

#### (4) 使用ワーク・エリア

REGX

REGY

REGW (符号の退避, 指数差, 演算結果がマイナスになったときの結果補正)

#### (5) ネスティング・レベル

2レベル

## (6) 処理手順

加算の処理は、(a) (b) (c) (d) (e)の順で行います。

## (a) 被仮数と仮数の桁合わせ

REGXとREGYの桁を合わせます。固定小数点形式では桁を揃えて演算を行いますが、この演算パッケージで使用している浮動小数点形式では、指数を調整することで、仮数部が固定小数点の加算を行えるようにします。

(i) REGXとREGYの指数値の差を求めます。

(ii) REGXとREGYの指数差を求めた結果、REGXよりREGYの方が指数値が大きい場合、フラグ(FEXCHG)をセット(1)します。

これは、のちに加算の処理を行う際、指数部の小さいレジスタの仮数部を指数部の大きいレジスタの指数部に合わせてダウンシフトさせるため、どちらのレジスタをシフトさせるか記憶しておく必要があるためです。

なお、指数差がない場合、およびREGXの指数部の方がREGYの指数部より大きい場合は、フラグ(FEXCHG)はリセット(0)されます。

(iii) 指数差が12未満のときは演算処理を継続します。

指数差が12以上のときは、指数部の大きい方のレジスタの内容をREGXにコピーしてREGXを演算結果として演算を終了します。これは、この演算パッケージで用意した仮数部が12桁なので、小さい数値が格納できないためです。

(iv) 加算の演算処理は、REGXの仮数部にREGYの仮数部を足し込めるように指数部の大きい方のレジスタの指数部の値に揃えるため、指数部の小さいほうのレジスタの仮数部のダウン・シフト(桁合わせ)を行います。

桁合わせの例を次に示します。

$$68.372(\text{REGX}) + 245.91(\text{REGY})$$

$$= (6.8372 \times \underline{10^1}) + (2.4591 \times \underline{10^2})$$

;指数値を  $10^2$  に揃えるため、REGXの仮数部をダウン・シフトします。

$$= (0.68372 \times 10^2)(\text{REGX})$$

$$+ (2.45910 \times 10^2)(\text{REGY})$$

(b) 被演算データと演算データが同符号の場合の加算処理

REGXとREGYが同符号の場合は、符号を変えずに仮数部の絶対値の加算で演算結果が求められます。そこで、仮数部を10進加算するルーチン(SADDBCD)を呼び出します。

(c) 被演算データと演算データが異符号の場合の加算処理

REGXとREGYが異符号の場合は、加数、被加数をそれぞれ減数、被減数と考えることにより、減算の形で演算結果が求められます。そこで、仮数部を10進減算するルーチン(SSUBBCD)を呼び出します。演算結果はREGXの仮数部に格納されます。

(d) 演算結果の補正

(c)の演算結果がマイナスになった(演算前のREGXの仮数部よりREGYの仮数部が大きい)場合は、仮数部の補数を取り、符号を反転し、演算結果とします。

(e) 正規化

演算結果が格納されたREGXを正規化するルーチン(SNML)を呼び出し、処理を終了します。

### 3.2 浮動小数点減算 (SFPSUB)

#### (1) 処理内容

REGXとREGYの浮動小数点数を10進減算し、結果を正規化してREGXに格納します。

#### (2) 入力条件

REGX, REGYにそれぞれ正規化された被減数, 減数を格納します。

#### (3) 出力結果

・REGXには正規化された演算結果を格納しています。

#### (4) 使用ワーク・エリア

REGX

REGY

REGW (符号の退避, 指数差, 演算結果がマイナスになったときの結果補正)

#### (5) ネスティング・レベル

2レベル

(6) 処理手順

減算は、減数の符号を反転することにより、加算として考えることができます。  
このため、(a) (b)の順で処理を行います。

(a) 減数の符号反転

REGYの符号部を反転します。

(b) 浮動小数点加算処理

この後の処理は加算と同様なので、浮動小数点加算(SFPADD)に処理を移行します。  
詳細は、浮動小数点加算(SFPADD)の処理手順を参照してください。

### 3.3 浮動小数点乗算 (SFPMULT)

#### (1) 処理内容

REGXとREGYの浮動小数点数を10進乗算し、結果を正規化してREGXに格納します。

#### (2) 入力条件

REGX, REGYにそれぞれ正規化された乗数, 被乗数を格納します。

注 乗算処理に限り, REGXを演算データ, REGYを被演算データとして扱います。

#### (3) 出力結果

- ・REGXには正規化された演算結果を格納しています。
- ・REGYには乗数を保持しています。

#### (4) 使用ワーク・エリア

REGX

REGY

REGW (指数値の退避, 乗数の格納, カウンタ)

#### (5) ネスティング・レベル

2レベル

(6) 処理手順

この演算パッケージの乗算は、筆算の要領で行っています。

具体例として、 $25.7 \times (-0.32)$  を挙げて、筆算による乗算の手順を考えます。

図5 - 4 筆算による乗算例

$$\begin{array}{r}
 + 25.7 \\
 \times (-0.32) \\
 \hline
 514 \quad \dots (1) \\
 771 \quad \dots (2) \\
 \hline
 - 8.224
 \end{array}$$

演算結果の符号を求めます。被乗数と乗数が同符号なら結果の符号は正で、異符号なら負となります。この例は正数と負数の乗算なので、結果は負となります。

被乗数と、乗数のそれぞれの桁の乗算を、乗数の下位桁から行います。このとき、小数点は無視して演算を行います。

(1)  $257 \times 2$  を行います。

(2)  $257 \times 3$  を行います。乗数の3は10の位にあたるので、結果は1桁左にずらして書きます。

桁ごとの乗算の結果を加算します。

演算結果の小数点の位置を求めます。この例の場合、被乗数が  $257 \times 10^{-1}$ 、乗数が  $32 \times 10^{-2}$  なので、の加算結果を  $10^{-3}$ 倍します。つまり、下から3桁目と4桁目の間に小数点を置きます。

以上より、 $25.7 \times (-0.32)$  の結果は  $-8.224$  となります。

この演算パッケージでは、筆算による乗算を、次のように実現しています。

ここでは具体例として、 $351.82(\text{REGX}) \times (-0.0947)(\text{REGY})$  を挙げて説明します。

$$351.82 = 3.5182 \times 10^2$$

$$-0.0947 = -9.47 \times 10^{-2}$$

REGX	0	2・0	0・0・0・0・0・0・0・0・2・8・1・5・3
REGY	1	E・F	0・0・0・0・0・0・0・0・0・0・7・4・9
			LSD <span style="float: right;">MSD</span>
	符号部	指数部	仮数部

(a) 符号部の乗算

演算結果の符号を求めます (REGXの符号部 REGYの符号部)。乗数 (REGX) と被乗数 (REGY) が同符号なら結果の符号は正 (0) で、異符号なら負 (1) となります。結果はREGXの符号部に退避します。

REGX	0	2・0	0・0・0・0・0・0・0・0・2・8・1・5・3
REGY	1	E・F	0・0・0・0・0・0・0・0・0・0・7・4・9
			LSD <span style="float: right;">MSD</span>
	符号部	指数部	仮数部

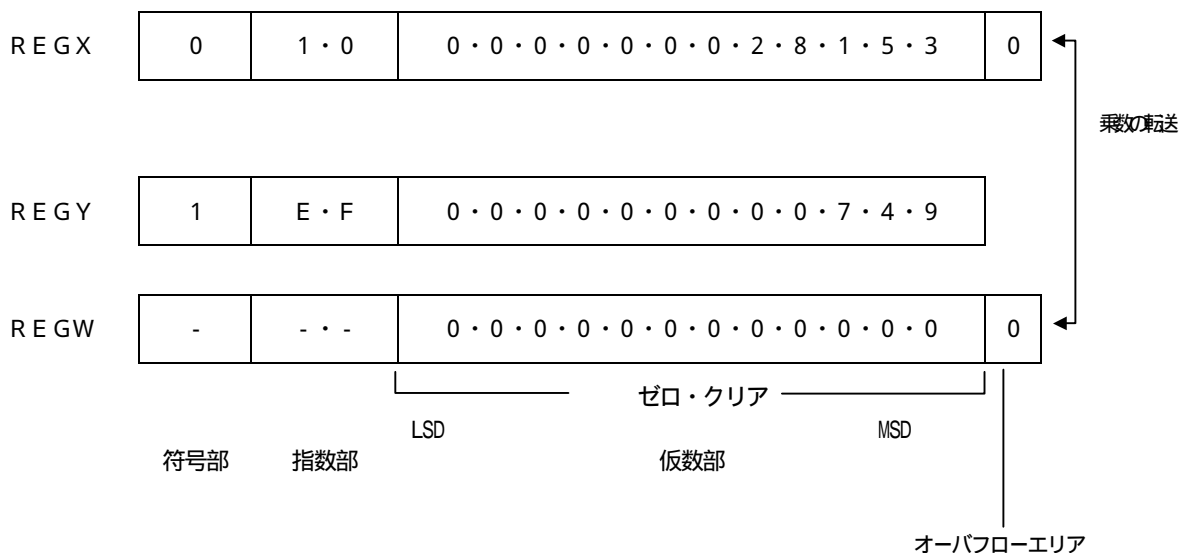
(b) 指数部の加算

演算結果の指数を求めます。乗数 (REGX) と被乗数 (REGY) の指数部を加算し、結果を REGX の指数部に格納します。

REGX	0	2・0	0・0・0・0・0・0・0・0・2・8・1・5・3
		+	
REGY	1	E・F	0・0・0・0・0・0・0・0・0・0・0・7・4・9
REGX	1	0・0	0・0・0・0・0・0・0・0・2・8・1・5・3
		LSD	MSD
	符号部	指数部	仮数部

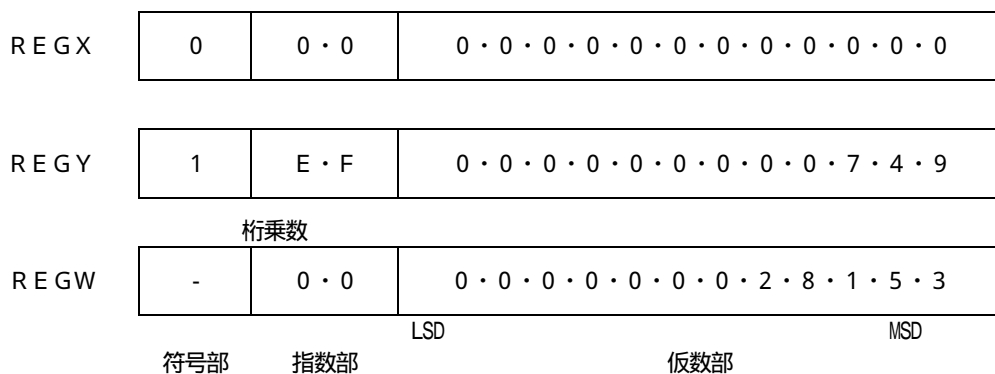
(c) 乗数の転送と演算結果エリアのクリア

REGWをゼロ・クリア(SRWCLR)し、REGXとREGWの仮数部を交換(SEXCHW)します。これは、REGXに演算結果を格納するためです。



(d) 仮数部12桁ダウン・シフト・カウンタの設定

桁カウンタ(RDIGCN)仮数部12桁ダウン・シフト・カウンタとします。



仮数部12桁ダウン・シフト・カウンタは、仮数部の桁数を初期値として設定します。

これは、乗数の下位桁から順に乗算(桁乗数 × REGY)を行うため、仮数部のダウン・シフトの回数を記憶しておく必要があるためです。

( e ) 乗数の 1 桁ごとの乗算

( i ) 仮数部 1 2 桁ダウン・シフト・カウンタをダウン・カウントします。カウンタの値が F F H になったところで ,( f ) の処理を終了します。

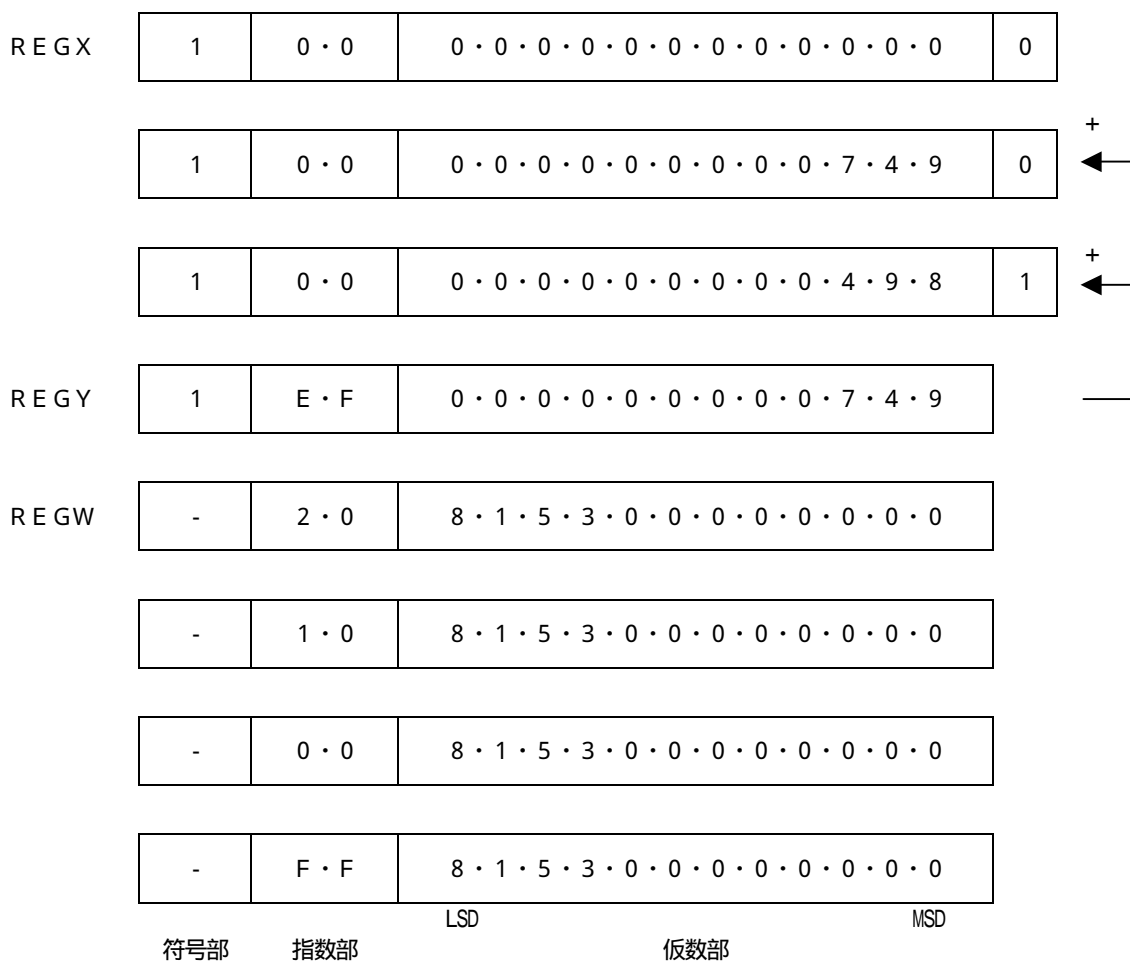
REGX	1	1・0	0・0・0・0・0・0・0・0・0・0・0・0・0									
REGY	1	E・F	0・0・0・0・0・0・0・0・0・0・0・7・4・9									
REGW	-	0・0	0・0・0・0・0・0・0・0				2・8・1・5・3					
			LSD						MSD			
	符号部	指数部	仮数部									
RDI GCN	C・0											
	仮数部 1 2 桁ダウン・シフト・カウンタ - 1											

( ii ) 乗数を格納している REGW の仮数部を 1 桁ダウン・シフトします。ダウン・シフトの結果 , あふれた最下位桁は , 指数部の下位桁に格納します。  
このあふれた最下位桁が 1 桁ごとの乗算の乗数 ( 以後桁乗数 ) になります。

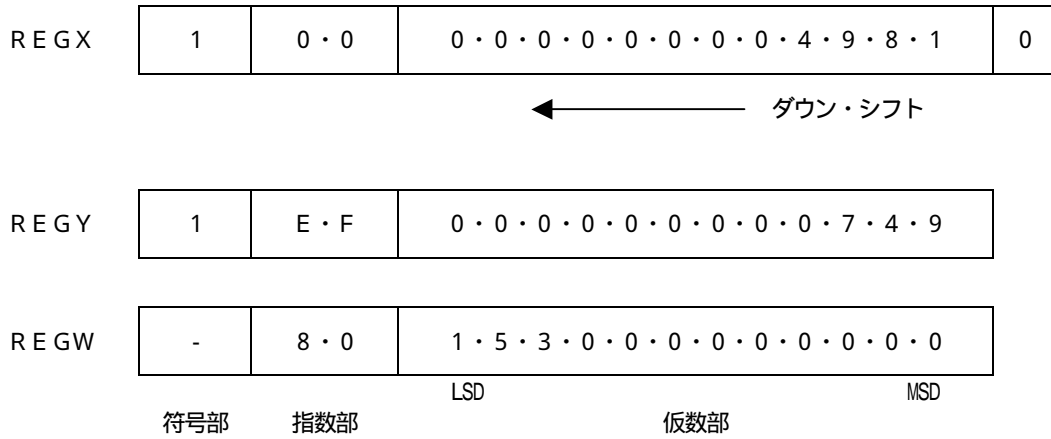
REGX	1	1・0	0・0・0・0・0・0・0・0・0・0・0・0・0									
REGY	1	E・F	0・0・0・0・0・0・0・0・0・0・0・7・4・9									
REGW	-	2・0	8・1・5・3				0・0・0・0・0・0・0・0					
			LSD						MSD			
	符号部	指数部	桁乗数 ←				ダウン・シフト					
			仮数部									
RDI GCN	5・0											

注 8 桁目の処理です。

(iii) 乗数の1桁ごとの乗算を行います。具体的には、演算結果を格納しているREGXに、REGYの被乗数を桁乗数の回数だけ加算します。



(iv) REGXの仮数部を1桁ダウン・シフトします。筆算による乗算では、乗数の桁ごとに乗算結果を上位桁にずらして書きますが、この演算パッケージでは、逆に、それまでの桁の乗算結果の合計を下位桁にずらすことにより、同じ処理を実現しています。



(v)(i) - (iv)の動作を仮数部12桁分(仮数部12桁ダウン・シフト・カウンタがFFHになるまで)行います。

(g) 正規化

正規化 (SNML) を行い、処理を終了します。

REGX	1	1・0	0・0・0・0・4・5・3・7・1・3・3・3			
REGY	1	E・F	0・0・0・0・0・0・0・0・0・7・4・9			
REGW	-	F・F	0・0・0・0・0・0・0・0・0・0・0・0			
	符号部	指数部	LSD	仮数部		MSD

### 3.4 浮動小数点除算 (SFPDIV)

#### (1) 処理内容

REGXとREGYの浮動小数点数を10進除算し、結果を正規化してREGXに格納します。

#### (2) 入力条件

REGX, REGYにそれぞれ正規化された被除数, 除数を格納します。

#### (3) 出力結果

- ・REGXには正規化された演算結果を格納しています。

除数がゼロの場合, エラー (FDVERRフラグ セット) となり, REGXに格納されている被除数を保持したまま処理を終了します。

- ・REGYには除数を保持しています。

#### (4) 使用ワーク・エリア

REGX

REGY

REGW (演算結果, カウンタ)

#### (5) ネスティング・レベル

2レベル

(6) 処理手順

具体例として、 $94.68 (REGX) \div 7.2531 (REGY)$  を挙げて説明します。

$$94.68 = 9.468 \times 10^1$$

$$7.2531 = 7.2531 \times 10^0$$

REGX	0	1・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9
REGY	0	0・0	0・0・0・0・0・0・0・0・1・3・5・2・7
	符号部	指数部	LSD 仮数部 MSD

(a) 符号部の除算

演算結果の符号を求め (REGXの符号部 REGYの符号部), REGXの符号部に格納します。被除数 (REGX) と除数 (REGY) が同符号なら結果の符号は正 (0) で, 異符号なら負 (1) となります。

REGX	0	1・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9
REGY	0	0・0	0・0・0・0・0・0・0・0・1・3・5・2・7
REGX	0	1・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9
	符号部	指数部	LSD 仮数部 MSD

(b) 除数のゼロ・チェック

REGYの仮数部がゼロならエラー (FDVERRフラグ セット) となり, 処理を終了します。

(c) 指数部の減算

演算結果の指数を求めます。被除数 (REGX) と除数 (REGY) の指数部を減算 (SSUBEX) し、結果を REGX の指数部に格納します。

REGX	0	2・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9
REGY	0	1・0	0・0・0・0・0・0・0・0・1・3・5・2・7
REGX	0	1・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9

LSD  MSD

符号部      指数部  仮数部

(d) 演算結果エリアのゼロ・クリア

演算結果が格納される REGW の仮数部をゼロ・クリア (SRWCLR) し、符号部を仮数部 12 桁アップ・シフト・カウンタとします。

REGX	0・0	1・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9
REGY	0・0	1・0	0・0・0・0・0・0・0・0・1・3・5・2・7
REGW	C・0	-・-	0・0・0・0・0・0・0・0・0・0・0・0・0

LSD  MSD

符号部      指数部  仮数部

仮数部 12 桁アップ・シフト・カウンタ      ゼロ・クリア

注

注 - の部分には不定の値が入っています。

( e ) 復帰法による除算

この演算パッケージの除算は、復帰法を使っています。これは被除数を被減数、除数を減数として減算を行い、2 回目以降は減算結果から除数を減算していき、減算結果が負になるまでの減算回数をその桁の商とします。

また、減算結果が負になった場合は、除数を減算結果に加算することで正数に戻し、その結果を1桁アップ・シフトして減算を進めます。シフトすることにより、次の減算過程における商が、前に算出された商の10分の1になることを意味します。

このような過程を減算結果がゼロになるまで行います。

図5に、復帰法による除算例を示します。

図5 復帰法による除算例

		商
$\begin{array}{r} 1.2 \\ 5 \overline{) 6} \\ \underline{5} \quad \dots \\ 10 \quad \dots \\ \underline{10} \quad \dots \\ \hline 0 \end{array}$	$\begin{array}{r} 6 \\ - 5 \\ \hline 1 \end{array}$	被除数から除数が減算できたので、商として1をたてます。
	$\begin{array}{r} 1 \\ - 5 \\ \hline -4 \end{array}$	減算結果が負になったので、商は数えませんが、
	$\begin{array}{r} -4 \\ + 5 \\ \hline 1 \\ 10 \end{array}$	減算結果を正数に戻すため、除数を加算します。さらに正数になった減算結果をアップ・シフトします。それにより商の桁が1つ下がります。
	$\begin{array}{r} 10 \\ - 5 \\ \hline 5 \end{array}$	減算できたので、商として0.1をたて、現在の商の1と加算します。
	$\begin{array}{r} 5 \\ - 5 \\ \hline 0 \end{array}$	減算できたので、商に0.1を加算します。減算結果が0になったので終了します。

( i ) 仮数部12桁アップ・シフト・カウンタをダウン・カウントします。

REGX	0・0	1・0	0・0・0・0・0・0・0・0・0・0・8・6・4・9	0
------	-----	-----	-----------------------------	---

REGY	0・0	1・0	0・0・0・0・0・0・0・0・0・1・3・5・2・7	0
------	-----	-----	-----------------------------	---

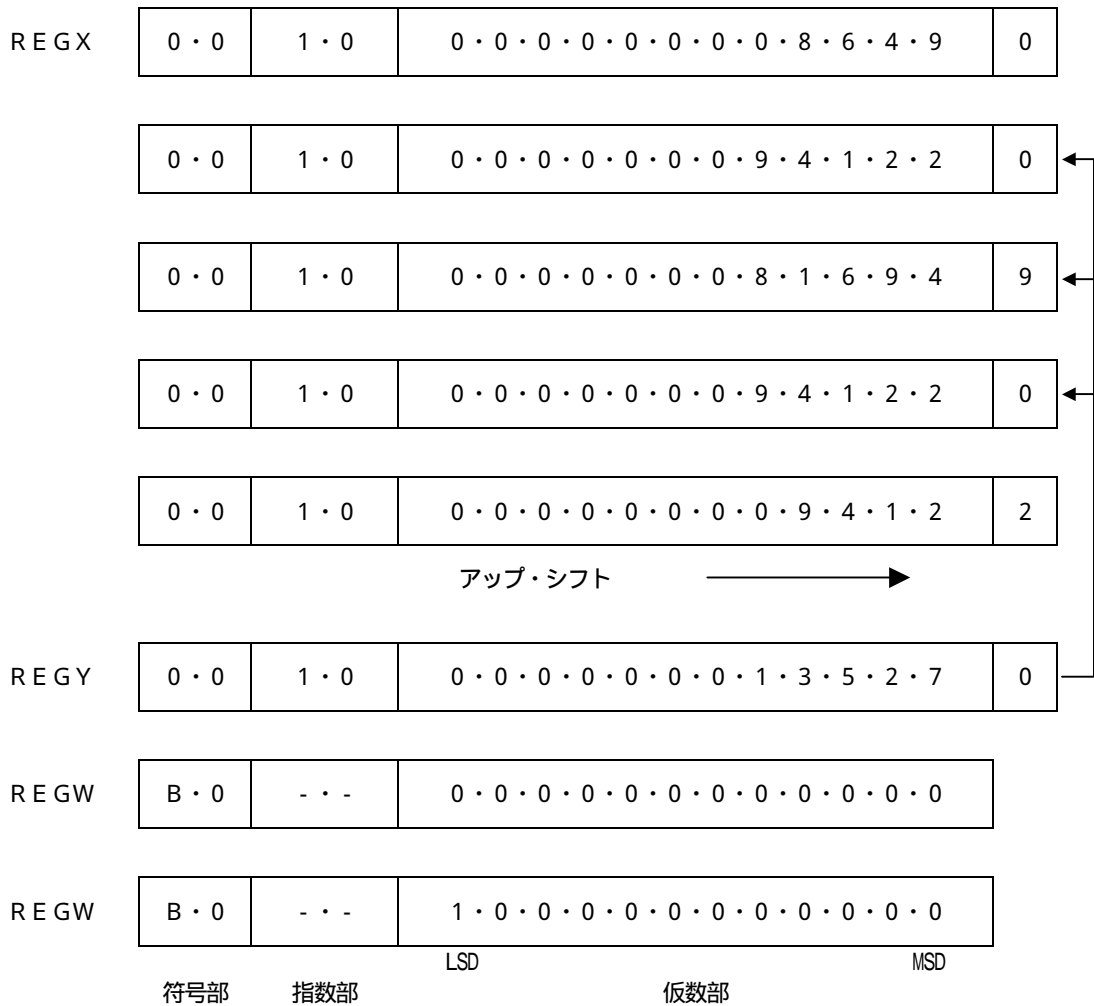
REGW	B・0	-・-	0・0・0・0・0・0・0・0・0・0・0・0・0	
------	-----	-----	---------------------------	--

仮数部12桁アップ  
・シフト・カウンタ - 1

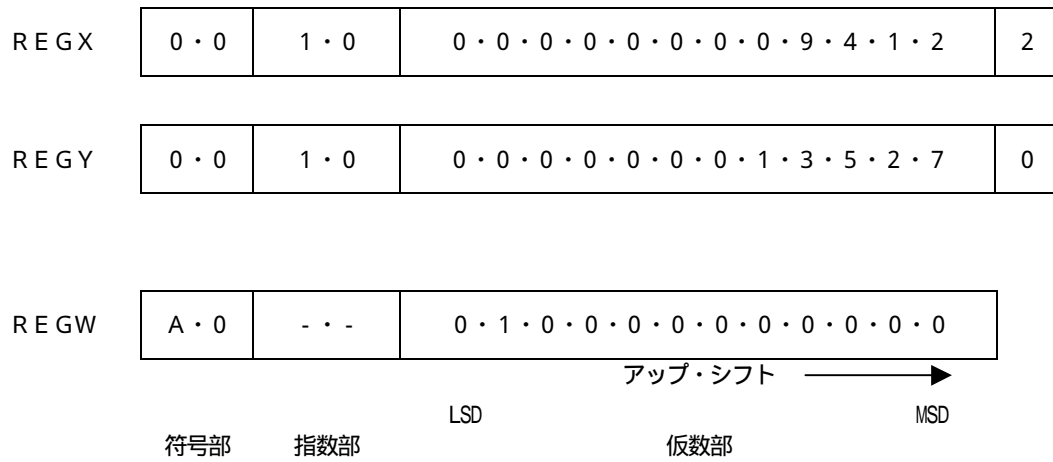
符号部      指数部      LSD      仮数部      MSD

(ii) REGXからREGYを減算します。アンダフローしなければ、REGWの最下位桁に商として1を加えます。

アンダフローした場合は、REGXとREGYを加算し、REGXを1桁アップ・シフトします。これは、減算結果をアップ・シフトすることにより次の減算に備えるためです。



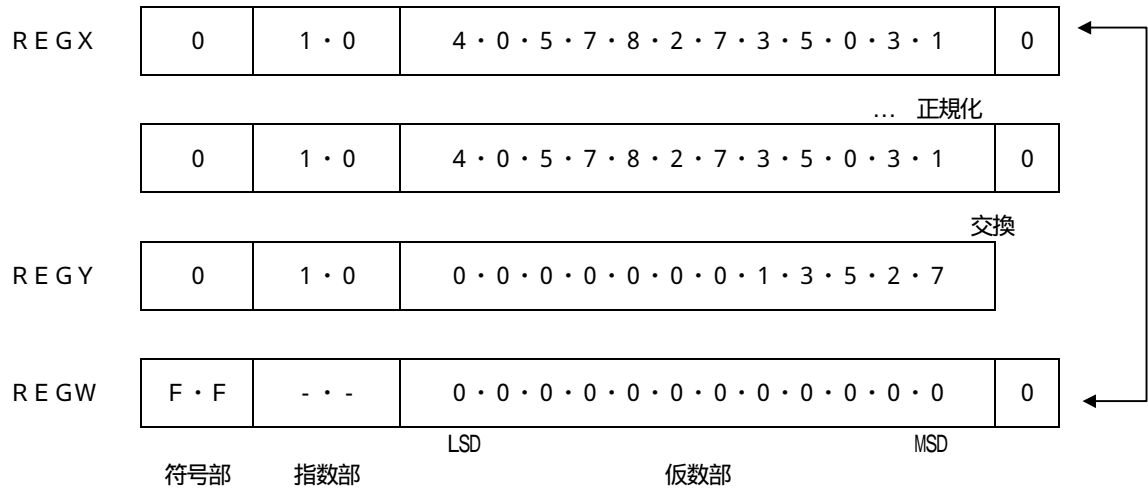
(iii) REGWの仮数部を1桁アップ・シフトします。これは、演算結果をアップ・シフトすることにより、商の桁を更新するためです。



(iv) (i) - (iii)の動作を仮数部12桁分(仮数部12桁アップ・シフト・カウンタがFFHになるまで)行います。

( f ) 正規化

REGXとREGWの仮数部を交換 (SCHGXW) し、演算結果をREGXに格納し、正規化 (SNML) を行い、処理を終了します。



## 4 . その他のルーチン ( ユーザから呼ばれることのない内部ルーチンです )

### 4 . 1 正規化 ( SNML )

#### 処理内容

REGXに格納された演算結果を正規化します。

演算結果がゼロの場合、演算結果ゼロ・フラグ ( FZERO ) がセット ( 1 ) されます。

演算結果がオーバーフロー、またはアンダフローした場合、オーバーフロー・フラグ ( FOVER ) がセット ( 1 ) されます。

オーバーフロー、アンダフローの検出は、REGXの指数値により行います。指数値は、16進数2桁で、2の補数表現を用いて表していますので、表せる指数値の範囲は - 128 指数値 127となりますが、この演算パッケージでは、指数値が - 12 指数値 12の範囲を超えた場合をオーバーフロー、アンダフローとし、オーバーフロー・フラグをセットしています。オーバーフロー・フラグをシステム制御部でチェックすることにより、オーバーフロー、アンダフローの発生による演算結果の破壊を防ぐことができます。

### 4 . 2 仮数部加算 ( SADDBCD )

#### 処理内容

REGXとREGYの仮数部を10進加算し、演算結果をREGXに格納します。

### 4 . 3 仮数部減算 ( SSUBBCD )

#### 処理内容

REGXとREGY、またはREGWの仮数部を10進減算し、演算結果をREGXに格納します。

#### 4.4 レジスタ交換 (SEXCHXW)

処理内容

REGXとREGWの仮数部を交換します。

#### 4.5 レジスタ・アップ・シフト (SSHUPX)

処理内容

REGXの仮数部を1桁アップ・シフトします。

#### 4.6 レジスタ・ゼロ・クリア

##### (1) レジスタ・ゼロ・クリア1 (SRXCLR)

処理内容

REGXの符号部, 指数部, 仮数部をゼロ・クリアします。

##### (2) レジスタ・ゼロ・クリア2 (SRWCLR)

処理内容

REGWの仮数部をゼロ・クリアします。

## 5 . R A M説明

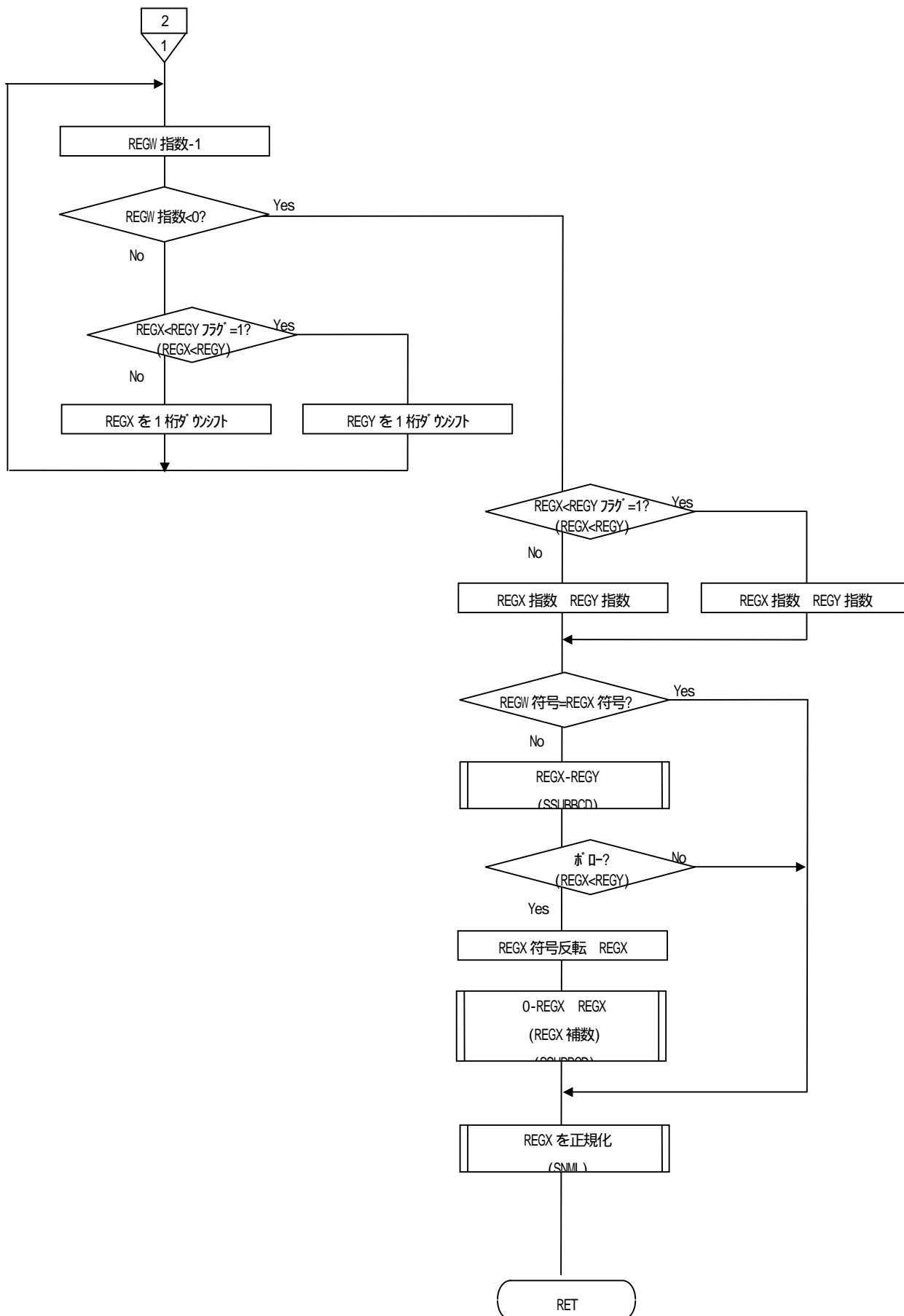
ここでは、浮動小数点演算で使用しているRAMについて説明します。

### 5 . 1 R A Mおよびフラグ説明

名称	サイズ(Byte)	呼称	説明	属性
RREGX	14	浮動小数点レジスタ 1	演算用レジスタです。演算データ、および演算結果を格納します。	L
RXSIGN	(1)		RREGX の符号部	P
RXEXP	(1)		RREGX の指数部	P
RXLSD	(1)		RREGX の仮数部 (最下位桁)	P
RXMSD	(1)		RREGX の仮数部 (最上位桁)	P
RXOVF	(1)		乗除算オーバーフローエリア	L
RREGY	14	浮動小数点レジスタ 2	演算用レジスタです。演算データを格納します。	L
RYSIGN	(1)		RREGY の符号部	P
RYEXP	(1)		RREGY の指数部	P
RYLSD	(1)		RREGY の仮数部 (最下位桁)	P
RYMSD	(1)		RREGY の仮数部 (最上位桁)	P
RYOVF	(1)		乗除算オーバーフローエリア	L
RREGW	14	浮動小数点レジスタ 3	演算用のワーク・レジスタです。	L
RWSIGN	(1)		RREGW の符号部	L
RWEXP	(1)		RREGW の指数部	L
RWLSD	(1)		RREGW の仮数部 (最下位桁)	L
RWMSD	(1)		RREGW の仮数部 (最上位桁)	L
RWOVF	(1)		乗除算オーバーフローエリア (予備)	L
RDIGCN	1	桁カウンタ	乗算時桁処理カウンタ	
FEXCHG	1bit	レジスタ交換フラグ	加減算時に、浮動小数点レジスタ 1 と浮動小数点レジスタ 2 の内容を交換したときにセットします。	P
FZERO	1bit	演算結果ゼロフラグ	演算結果が 0 のときにセットします。	P
FDIVERR	1bit	除数ゼロ・エラー・フラグ	除算で、除数が 0 のときにセットします。	P
FOVER	1bit	オーバーフロー・フラグ	演算によりオーバーフローが発生したときにセットします。	P

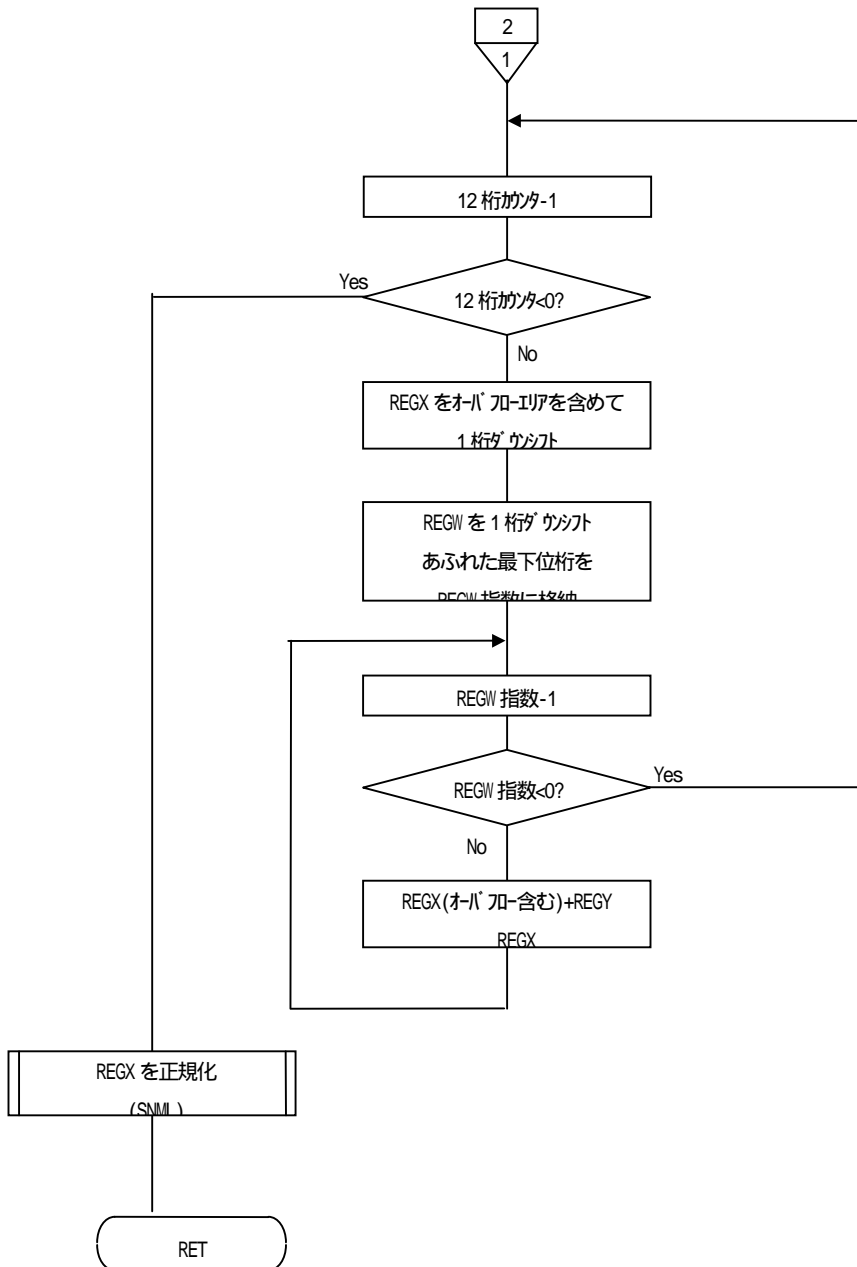
\*属性は P : PUBLIC , L : LOCAL





6.2 浮動小数点乗算





・ 6 . 3 浮動小数点除算

