お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

アプリケーション・ノート

78K/0 シリーズ

8 ビット・シングルチップ・マイクロコンピュータ

浮動小数点演算プログラム編

μPD78014 シリーズ
μPD78014Y シリーズ
μPD78044 シリーズ
μPD78054 シリーズ
μPD78064 シリーズ

資料番号　U13482JJ2V0AN00（第 2 版）
（旧資料番号　IEA-718）
発行年月　May 1998　N　CP(K)

〔メ　モ〕

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

注意：本製品はI^2Cバス・インタフェース回路を内蔵しています。
I^2Cバス・インタフェースを使用される場合には，カスタム・コードをご発注いただく時に，事前にその旨ご申告下さい。申告に基づき，以下の特典が受けられます。
日本電気株式会社のI^2Cバス対応部品をご購入いただくことにより，これらの部品をI^2Cシステムに使用する実施権がフィリップス社I^2C特許に基づき許諾されることになります。ただし，これらのI^2Cシステムはフィリップス社によって設定されたI^2C標準規格に合致しているものとします。

Purchase of NEC I^2C components conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

該当製品：μPD78011BY, 78012BY, 78013Y, 78014Y, 78P014Y

本製品のうち，外国為替および外国貿易管理法の規定により戦略物資等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

本版で改訂された主な箇所

箇　所	内　容
全　般	対象製品の品名変更 • μPD78011，78012 → 78011B，78012B 対象製品の追加 • μPD78011BY，78012BY，78013Y，78014Y • μPD78042，78043，78044，78P044 • μPD78052，78053，78054，78P054，78056，78058 • μPD78062，78063，78064，78P064
p.203	**付録　SPDチャートの説明**を追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者　このアプリケーション・ノートは，78K/0 シリーズ製品の機能を理解し，78K/0 シリーズ製品を用いた浮動小数点演算プログラムを設計するエンジニアを対象とします。

78K/0 シリーズ製品

- ●μPD78014シリーズ　：μPD78011B，78012B，78013，78014，78P014
- ●μPD78014Yシリーズ：μPD78011BY，78012BY，78013Y，78014Y，78P014Y
- ●μPD78044シリーズ　：μPD78042，78043，78044，78P044
- ●μPD78054シリーズ　：μPD78052，78053，78054，78P054，78056[注]，78058[注]
- ●μPD78064シリーズ　：μPD78062[注]，78063，78064，78P064[注]

注　開発中

目　的　このアプリケーション・ノートは，78K/0 シリーズ製品の浮動小数点演算応用プログラムについてユーザに理解していただくことを目的とします。なお，記載のプログラムは例示的に示したものであり，量産設計を対象とするものではありません。

構　成　このアプリケーション・ノートでは，次の内容について説明しています。

- ●計算アルゴリズム
- ●四則演算
- ●関数（数学，座標変換，型変換）
- ●実行結果
- ●プログラム・リスト

なお，次のアプリケーション・ノートも別に用意しています。

- ●基礎編 I （IEA-715）
- ●基礎編 II （IEA-740）

品質水準　標準（一般電子機器用）

品質水準とその応用分野の詳細については当社発行の資料「NEC 半導体デバイスの品質水準」（IEI-620）をご覧ください。

応用分野　●民生機器分野

関連資料

○78K/0シリーズ共通資料

資料名		資料番号
アプリケーション・ノート	基礎編Ⅰ	IEA-715
	基礎編Ⅱ	IEA-740
	浮動小数点演算プログラム編	このアプリケーション・ノート
セレクション・ガイド		IF-375
インストラクション活用表		IEM-5522
インストラクション・セット		IEM-5521

○個別資料

●μPD78014シリーズ

品名 / 資料名	μPD78011B	μPD78012B	μPD78013	μPD78014	μPD78P014
データ・シート	作成中		IC-8201		IC-8111
ユーザーズ・マニュアル	IEU-780				
特殊機能レジスタ活用表	IEM-5527				

●μPD78014Yシリーズ

品名 / 資料名	μPD78011BY	μPD78012BY	μPD78013Y	μPD78014Y	μPD78P014Y
データ・シート	作成中				IC-8572
ユーザーズ・マニュアル	IEU-780				
特殊機能レジスタ活用表	IEM-5527				

●μPD78044シリーズ

品名 / 資料名	μPD78042	μPD78043	μPD78044	μPD78P044
データ・シート	IC-8497			IC-8499
ユーザーズ・マニュアル	IEU-801			
特殊機能レジスタ活用表	IEM-5556			

●μPD78054シリーズ

品名 / 資料名	μPD78052	μPD78053	μPD78054	μPD78P054	μPD78056	μPD78058
データ・シート	作成中					
ユーザーズ・マニュアル	IEU-824					
特殊機能レジスタ活用表	IEM-5574					

●μPD78064シリーズ

品名 / 資料名	μPD78062	μPD78063	μPD78064	μPD78P064
データ・シート	作成中			IP-8636
ユーザーズ・マニュアル	IEU-817			
特殊機能レジスタ活用表	IEM-5568			

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには必ず最新の資料をご使用ください。

目　　次

第1章　概　　要

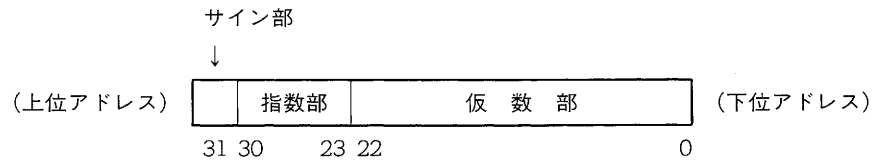
1.1　浮動小数点形式

この演算プログラムでは，浮動小数点数を４バイトで表現します。内訳は次のとおりです（下図参照)。

●仮数部　：23ビット

●指数部　：　8ビット

●サイン部：　1ビット



この形式の数値は，次のようになります。

$$(-1)^{(\text{サイン部値})} \times (\text{仮数部値}) \times 2^{(\text{指数部値})}$$

次に，各部の詳細について説明します。

(1)仮 数 部

仮数部は絶対値で表現され，仮数部のビット位置22-0が２進数の小数点第１位-第23位に相当します。

仮数部値は浮動小数点値が０になる場合を除いて常に1-2の範囲になるように指数部の値が調整されます。そのため１の位（１の値を意味する）は常に１となり，この形式では省略した形で表現しています。

備考　最上位ビット（MSB）を常に１にする操作を正規化といいます。

(2)サイン部

正数の場合０，負数の場合１とします。

(3) 指 数 部

底2の指数を1バイトの整数型（負の場合，2の補数表現）で表し，この値にさらに7FHのバイアスを加えた値を用いています。これらの関係は，具体的には下表のようになります。

指数部（16進）	指数部の値
FF	128
FE	127
⋮	⋮
81	2
80	1
7F	0
7E	−1
⋮	⋮
01	−126

注意　指数部が0のときにのみ，浮動小数点値は0を表します。この場合，仮数部，およびサイン部は無視されます。

(4) 数値の表現範囲

浮動小数点値xは，次の範囲の値と“0”を表現することが可能です。

$$約1.1755 \times 10^{-38} \leqq |x| < 約6.8056 \times 10^{38}$$

1.2 提供関数

この演算プログラムの提供関数について概説します。

提供関数として次の４つを用意しています。

●四則演算

●数学関数

●座標変換関数

●数値型変換関数

提供関数	関　　数	関数名
四則演算	加算	LADD
	減算	LSUB
	乗算	LMLT
	除算	LDIV
数学関数	三角関数 sin 関数	LSIN
	三角関数 cos 関数	LCOS
	三角関数 tan 関数	LTAN
	自然対数関数（log）	LLOG
	常用対数関数（$\log_{10}$）	LLOG10
	指数関数（底＝e）	LEXP
	指数関数（底＝10）	LEXP10
	べき乗（a^b）	LPOW
	平方根	LSQRT
	逆三角関数 arcsin 関数	LASIN
	逆三角関数 arccos 関数	LACOS
	逆三角関数 arctan 関数	LATAN
	双曲線関数 sinh 関数	LHSIN
	双曲線関数 cosh 関数	LHCOS
	双曲線関数 tanh 関数	LHTAN
	絶対値	LABS
	逆数	LRCPN
座標変換関数	極座標 → 直交座標への変換	POTORA
	直交座標 → 極座標への変換	RATOPO
数値型変換関数	文字列 → 浮動小数点形式への変換	ATOL
	浮動小数点形式 → 文字列への変換	LTOA
	２バイト整数型 → 浮動小数点形式への変換	FTOL
	浮動小数点形式 → ２バイト整数型への変換	LTOF

1.3 ファイル構成

この演算プログラムは，次の４種類のファイルにより構成されています。

●オブジェクト・ソース・ファイル[注] ：24
●共通サブルーチン・ファイル[注] ： 2
●共通データ・ファイル ： 1
●インクルード・ファイル ： 4

注　オブジェクト・ソース・ファイルは，構造化アセンブラ言語で記述されています。

(1) オブジェクト・ソース・ファイル

ソース・ファイル名	提　供　関　数
LFLT1　.SRC	四則演算
LSIN　.SRC	sin 関数
LCOS　.SRC	cos 関数
LTAN　.SRC	tan 関数
LLOG　.SRC	自然対数関数
LLOG10 .SRC	常用対数関数
LEXP　.SRC	指数関数（底＝e）
LEXP10　.SRC	指数関数（底＝10）
LPOW　.SRC	べき乗関数
LSQRT　.SRC	平方根
LASIN　.SRC	arcsin 関数
LACOS　.SRC	arccos 関数
LATAN　.SRC	arctan 関数
LHSIN　.SRC	sinh 関数
LHCOS　.SRC	cosh 関数
LHTAN　.SRC	tanh 関数
LABS　.SRC	絶対値
LRCPN　.SRC	逆数
POTORA.SRC	極座標 → 直交座標への変換
RATOPO.SRC	直交座標 → 極座標への変換
ATOL　.SRC	文字列 → 浮動小数点形式への変換
LTOA　.SRC	浮動小数点形式 → 文字列への変換
FTOL　.SRC	２バイト整数型 → 浮動小数点形式への変換
LTOF　.SRC	浮動小数点形式 → ２バイト整数型への変換

(2) 共通サブルーチン・ファイル

ソース・ファイル名	提　供　関　数
LFLT2　.SRC	多項式計算関数
LLD　.SRC	浮動小数点レジスタ間ロード/置換関数

(3) 共通データ・ファイル

ソース・ファイル名	定　義　内　容
DFLT　.SRC	浮動小数点レジスタ定義

(4) インクルード・ファイル

ソース・ファイル名	定　義　内　容
EQU　.INC	EQU 定義
REF1　.INC	浮動小数点レジスタ参照宣言
REF2　.INC	浮動小数点レジスタ間ロード/置換関数使用宣言
ASCII　.INC	アスキー・コード定義

各関数の使用にあたって，リンケージをとるべきオブジェクト・モジュール・ファイルについては，各関数の解説のところで記述しています。

備考　当社の 78K/0 シリーズ用アセンブラ・パッケージには，ライブラリアンが付属しており，このライブラリアンを使用してライブラリ・ファイルを作成することができます。上記の全プログラムをライブラリ・ファイルに登録することで，リンク時にそのライブラリ・ファイルを指定するだけで必要なモジュールを自動的にリンクすることができるので便利です。

なお，ライブラリ・ファイルへの登録順序には，特に制限はありません。

1.4 プログラムの特性

1.4.1 プログラムの配置アドレス

(1) ワーク・エリア

この演算プログラムで使用するワーク・エリアを浮動小数点レジスタと呼びます（実際には，単にグローバル変数というだけですが，使用方法が浮動小数点演算用に限定されるため，ここではレジスタと呼んでいます）。

浮動小数点レジスタの領域確保は，共通データ・ファイル DFLT.SRC で行っています。浮動小数点レジスタはショート・ダイレクト・アドレシング・エリア内に確保されます。ショート・ダイレクト・アドレシング・エリア内での配置アドレスは任意となります。

(2) コード・エリア

ROM 内に配置される必要がありますが，その他の制限はありません。

1.4.2 リエントラント性

リエントラント性は持っていません。

多重タスクの処理系では，1タスクのみがこの関数を呼ぶことができるようにするなどの資源管理が必要になります。

1.4.3 スタック

各関数の解説のところで，スタックの最大消費サイズを示しています。各関数の使用にあたって最低限，示されたサイズ分のスタックを用意する必要があります。

1.4.4 レジスタ・バンク

この演算プログラムでは，レジスタ・バンクの選択命令は使用していません。呼び出された時点で選択されているレジスタ・バンクを用いています。

1.4.5 レジスタ，フラグなどの保存

レジスタ内容，およびフラグのスタックへの退避/復帰は，型変換関数の一部を除いて行っていません。

また，関数により使用するレジスタは異なります（使用レジスタとその内容の保存については，各関数の解説のところで示しています）。

1.5 データの受け渡し方法

1.5.1 パラメータおよび返値

前述の浮動小数点レジスタにより受け渡しを行います。

この演算プログラムでは統一して，被演算値と返値の格納レジスタを共通にしています。その意味からこのマニュアル中では，被演算値をデスティネーション，演算値をソースと呼びます。

パラメータおよび返値の設定の詳細については，各関数の解説のところで説明しています。

1.5.2 演算結果の報告

終了状態を細かく分けると，次の5種類になります。

(1) 正常終了
(2) アンダフロー
(3) オーバフロー
(4) 虚数化
(5) 演算不能（log(−1) など）

(2) のアンダフローは返値0を正常終了として返します。

終了状態の報告は，エラー ((3), (4), (5)) 状態を区別しません。正常終了か異常終了かだけを報告します。

終了状態	Aレジスタ内容	CYフラグ
正常終了	0	off
異常終了	81H	on

1.6 浮動小数点レジスタ

このアプリケーション・ノート中では，演算プログラムで使用するワーク・エリアを浮動小数点レジスタと呼びます。ここでは，各レジスタの役割について簡単に説明します。

以降の図で，1つの□が1バイトを，また，左側が上位アドレスを表します。

1.6.1 浮動小数点レジスタ1（FPR1）

ほとんどの関数でデスティネーションの格納用に使用するレジスタです。

FPR1は下図に示す連続した4バイトのレジスタです。ショート・ダイレクト・アドレシング・エリア（ワード境界）に配置されます。

指数部	第1仮数部	第2仮数部	第3仮数部	
＋サイン	＋指数部LSB			
FPR1_4	FPR1_3	FPR1_2	FPR1_1	FPR1
	FPR1_HP		FPR1_LP	

↑

(on word border)

FPR1を構成する各バイト，および上位ワード（FPR1_HP），下位ワード（FPR1_LP）に広域名が付けられています。

1.6.2 浮動小数点レジスタ2（FPR2）

ソースの格納用に使用するレジスタです。

FPR2は下図に示す連続した4バイトのレジスタです。ショート・ダイレクト・アドレシング・エリア（ワード境界）に配置されます。

指数部	第1仮数部	第2仮数部	第3仮数部	
＋サイン	＋指数部LSB			
FPR2_4	FPR2_3	FPR2_2	FPR2_1	FPR2
	FPR2_HP		FPR2_LP	

↑

(on word border)

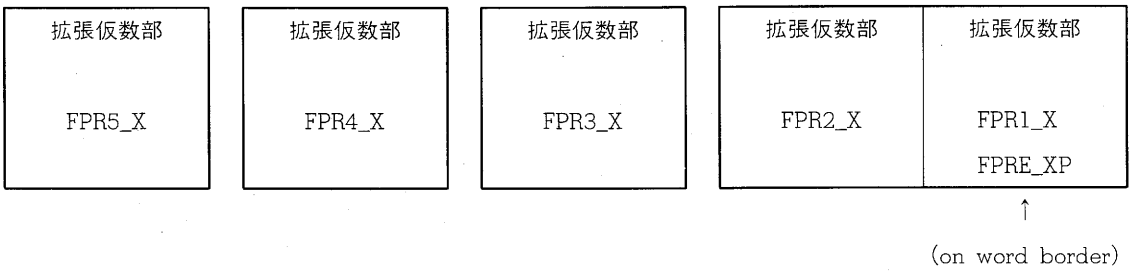
レジスタの構成は，FPR1と同様です。

1.6.3 浮動小数点レジスタ 3-5（FPR3-5）

数学関数などで，一時的なワークとして使用するレジスタです。

FPR3-5は，FPR1，2同様の構成と広域名を持ち，ショート・ダイレクト・アドレシング・エリア（ワード境界）に配置されます。

1.6.4 拡張仮数部レジスタ



仮数部を内部的に1バイト拡張して第4仮数部（小数点第24位-31位）として計算するためのエリアです。

FPR1，2，3，4，5用にFPR1_X，FPR2_X，FPR3_X，FPR4_X，FPR5_Xを用います。

FPR1_X，FPR2_X，FPR3_X，FPR4_X，FPR5_Xは，ショート・ダイレクト・アドレシング・エリアに配置されます。

備考 これらのレジスタは，すべての関数で使用するわけではありません。各関数で何番レジスタまで使用するかは，各関数の解説のところで説明します。

1.6.5 浮動小数点レジスタ間のロード/置換

数学関数や型変換関数では，頻繁にレジスタのロード/ストア/置換を行います。そのため，浮動小数点レジスタ間のロード/ストア関数，および置換関数を用意しています。

ロード/ストア関数および置換関数の一覧を示します。

関　数　名	機　　能
LLD21, LLD21X	1番レジスタ内容を2番レジスタへロード
LLD31, LLD31X	1番レジスタ内容を3番レジスタへロード
LLD41, LLD41X	1番レジスタ内容を4番レジスタへロード
LLD51, LLD51X	1番レジスタ内容を5番レジスタへロード
LLD32	2番レジスタ内容を3番レジスタへロード
LLD52	2番レジスタ内容を5番レジスタへロード
LLD13	3番レジスタ内容を1番レジスタへロード
LLD23, LLD23X	3番レジスタ内容を2番レジスタへロード
LLD24, LLD24X	4番レジスタ内容を2番レジスタへロード
LLD15	5番レジスタ内容を1番レジスタへロード
LLD25, LLD25X	5番レジスタ内容を2番レジスタへロード
LLD1C, LLD1CX	定数データを1番レジスタへロード
LLD2C, LLD2CX	定数データを2番レジスタへロード
LXC13, LXC13X	1番レジスタと3番レジスタ内容を置換
LXC14, LXC14X	1番レジスタと4番レジスタ内容を置換
LXC15, LXC15X	1番レジスタと5番レジスタ内容を置換

注意　関数名の最後が，“X”で終わっているものは，拡張仮数部を含めたロード/置換関数です。

第2章　計算アルゴリズム

ここでは，計算の基本となるアルゴリズムについて簡単に説明します。

2.1　関数の展開方法

3種類の展開法を使用します。

●平方根　　：Newton-Raphson 法
●逆三角関数：最良近似法
●その他　　：Taylor 展開法

2.2　丸め方法

0に向かって丸めを行います。

2.3　桁落ち防止方法

Taylor 展開の展開多項式が階差級数となった場合，使用する値の範囲によっては極端な桁落ちが起こる場合があります。このような桁落ちを防止するために，この演算プログラムでは，展開式の第1項から第N項までの各項の値が，単調かつ急速に0に近付くような展開式を使用しています。

2.4　多項式加算/乗算時の誤差

有効桁24ビットの数体系で，0に向かって丸めを行った8項の多項式の加算を行った場合，最大4ビットの誤差が含まれます。また，x^8 を7回の乗算で求めた場合，同様の誤差が含まれます。

このような誤差の累積を少なくするために，この演算プログラムでは，内部的に仮数部を31ビット（拡張仮数部8ビットを付加）で計算しています。

第３章　四則演算

四則演算関数として，次の４つを用意しています。

（1）浮動小数点加算（LADD）

FPR1 の値を被加数，FPR2 の値を加数として加算を行います。

（2）浮動小数点減算（LSUB）

FPR1 の値を被減数，FPR2 の値を減数として減算を行います。

（3）浮動小数点乗算（LMLT）

FPR1 の値を被乗数，FPR2 の値を乗数として乗算を行います。

（4）浮動小数点除算（LDIV）

FPR1 の値を被除数，FPR2 の値を除数として除算を行います。

演算結果は，FPR1 に格納されます。

3.1　浮動小数点加算（LADD）

(1) 処理内容

FPR1 の値を x，FPR2 の値を y として，x+y を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LADD からの戻り番地 2 バイトのみ）

(4) 使用レジスタ

AX，C，DE

(5) 使用ワーク・エリア

FPR1，FPR2，FPR1_X，FPR2_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：162 μs

最大：332 μs（0.5＋(−0.50000006)）

(7) 処理手順

(a) x または，y が 0 の場合，0 でない方を解として演算を終了します。

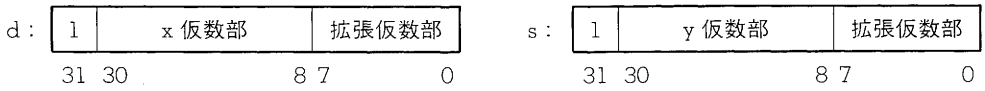
(b) 指数部差が 32 以上の場合，大きい方を解として演算を終了します。

(c) y 指数部＞ x 指数部の場合，x と y の内容を入れ替えます。

(d) x の指数部を記憶します。

(e) 次に示す方法で，仮数部の加減算を行います。

x，y の仮数部にそれぞれ，MSB (1) と拡張仮数部 (8 ビット) を加え，ダブルワード型の変数 d，s とみなします。



次の手順で，仮数部の和（または差）を求めます。

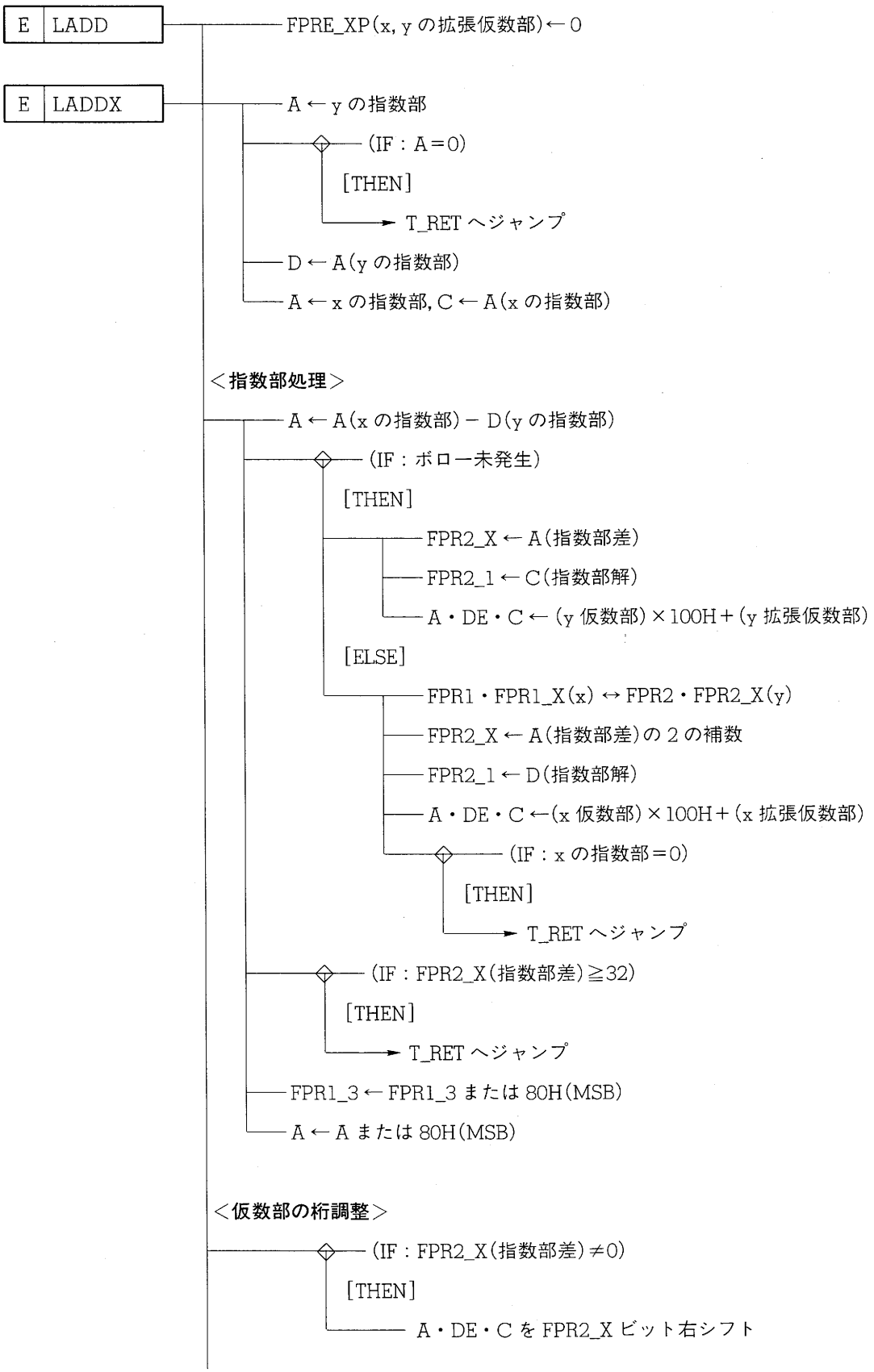
(i) x，y の指数部が不一致の場合，s を指数部差のビット数分右シフトします。

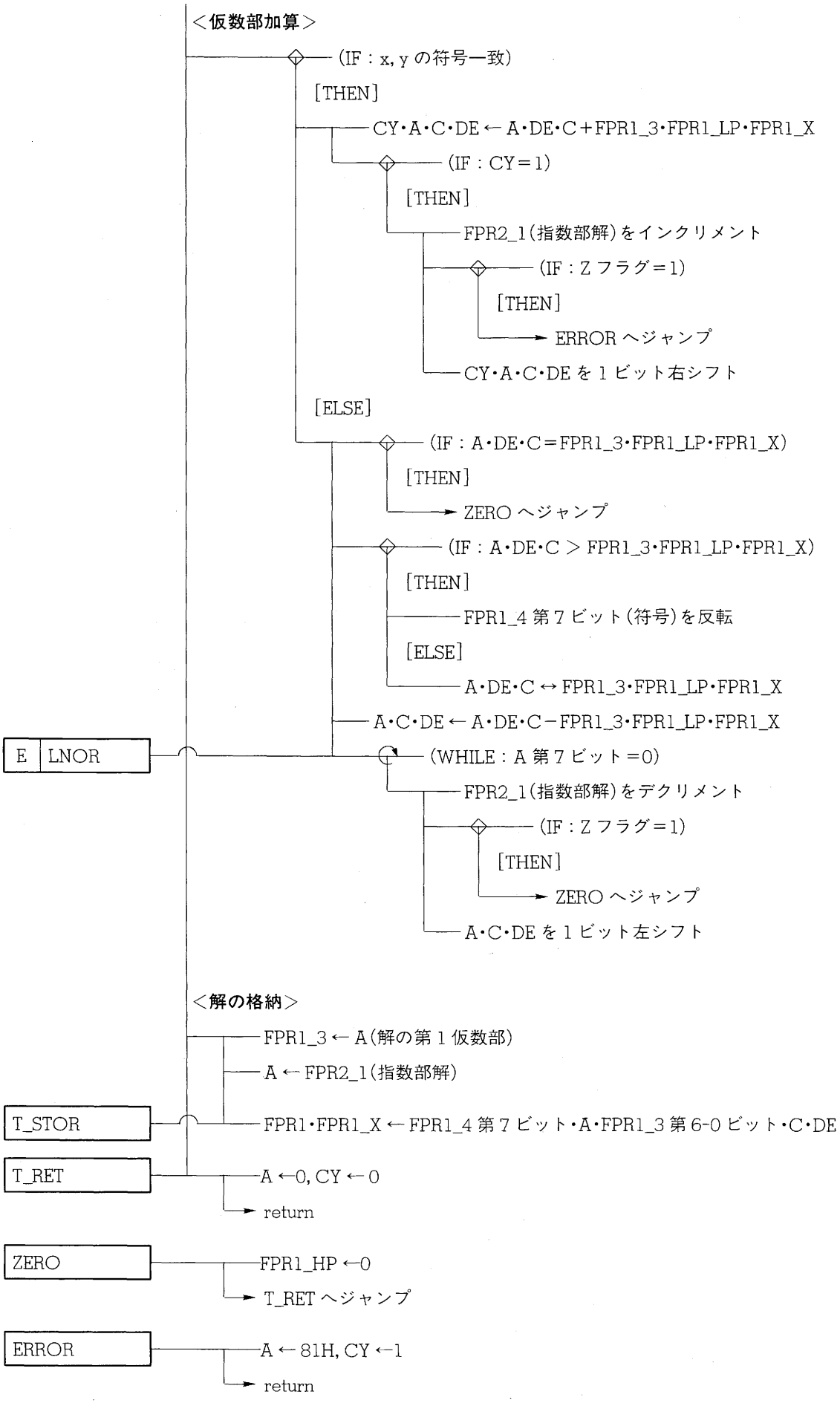
(ii) x，y の符号一致の場合，d，s の加算を行い，符号を記憶します。

(iii) 符号不一致の場合，d，s の大きい方から小さい方を減算し，大きい側に対応する符号を記憶します。

(f) 記憶していた指数部と，(e) で求めた仮数部解に対して正規化を行い，符号ビットとともに FPR1 に格納します。

(8) 処 理 図


E LADD
FPRE_XP(x, y の拡張仮数部)←0
E LADDX
A ← y の指数部
(IF : A=0)
[THEN]
T_RET へジャンプ
D ← A(y の指数部)
A ← x の指数部, C ← A(x の指数部)
＜指数部処理＞
A ← A(x の指数部) − D(y の指数部)
(IF : ボロー未発生)
[THEN]
FPR2_X ← A(指数部差)
FPR2_1 ← C(指数部解)
A・DE・C ← (y 仮数部)×100H+(y 拡張仮数部)
[ELSE]
FPR1・FPR1_X(x) ↔ FPR2・FPR2_X(y)
FPR2_X ← A(指数部差)の 2 の補数
FPR2_1 ← D(指数部解)
A・DE・C ←(x 仮数部)×100H+(x 拡張仮数部)
(IF : x の指数部=0)
[THEN]
T_RET へジャンプ
(IF : FPR2_X(指数部差)≧32)
[THEN]
T_RET へジャンプ
FPR1_3 ← FPR1_3 または 80H(MSB)
A ← A または 80H(MSB)
＜仮数部の桁調整＞
(IF : FPR2_X(指数部差)≠0)
[THEN]
A・DE・C を FPR2_X ビット右シフト


＜仮数部加算＞
(IF：x, y の符号一致)
[THEN]
CY・A・C・DE ← A・DE・C＋FPR1_3・FPR1_LP・FPR1_X
(IF：CY＝1)
[THEN]
FPR2_1(指数部解)をインクリメント
(IF：Z フラグ＝1)
[THEN]
ERROR へジャンプ
CY・A・C・DE を 1 ビット右シフト
[ELSE]
(IF：A・DE・C＝FPR1_3・FPR1_LP・FPR1_X)
[THEN]
ZERO へジャンプ
(IF：A・DE・C ＞ FPR1_3・FPR1_LP・FPR1_X)
[THEN]
FPR1_4 第 7 ビット(符号)を反転
[ELSE]
A・DE・C ↔ FPR1_3・FPR1_LP・FPR1_X
A・C・DE ← A・DE・C－FPR1_3・FPR1_LP・FPR1_X
E
LNOR
(WHILE：A 第 7 ビット＝0)
FPR2_1(指数部解)をデクリメント
(IF：Z フラグ＝1)
[THEN]
ZERO へジャンプ
A・C・DE を 1 ビット左シフト
＜解の格納＞
FPR1_3 ← A(解の第 1 仮数部)
A ← FPR2_1(指数部解)
T_STOR
FPR1・FPR1_X ← FPR1_4 第 7 ビット・A・FPR1_3 第 6-0 ビット・C・DE
T_RET
A ←0, CY ← 0
return
ZERO
FPR1_HP ←0
T_RET へジャンプ
ERROR
A ← 81H, CY ←1
return

備考 1. レーベル | E | | は広域名を示します。

2. レーベル | T_STOR | | T_RET | | ZERO | | ERROR |
は，LMLT，および LDIV 関数でも参照されます。

3. レーベル | E | LADDX | は数学関数などで，拡張仮数部を使用した加算を実行するための内部的な広域名です。

4. レーベル | E | LNOR | は型変換関数で，桁落ち状態からの正規化を実行するための内部的な広域名です。

5. CY・A・C・DE は，MSB＝CY の 33 ビット型変数を表現します。
この処理図内の他の組み合わせも同様の意味です。
また他の関数でも，処理図内において同様の表現を用いています。

3.2　浮動小数点減算（LSUB）

(1) 処理内容

FPR1 の値を x，FPR2 の値を y として，x－y を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LSUB からの戻り番地 2 バイトのみ）

(4) 使用レジスタ

AX，B，DE

(5) 使用ワーク・エリア

FPR1，FPR2，FPR1_X，FPR2_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

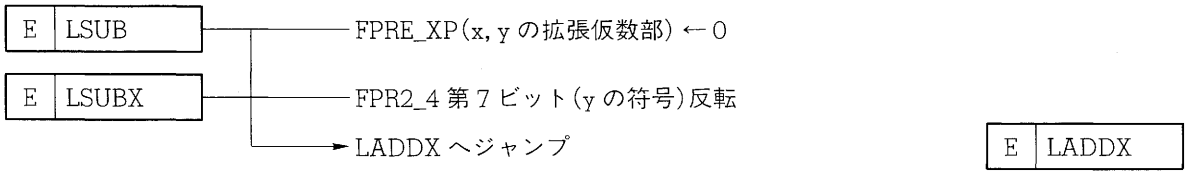
平均：155 μs

最大：335 μs（0.5－0.50000006）

(7) 処理手順

y の符号を反転し，LADD へジャンプします。

(8) 処 理 図



備考　レーベル E LSUBX は数学関数などで，拡張仮数部を使用した減算を実行するための内部的な広域名です。

3.3　浮動小数点乗算（LMLT）

(1) 処理内容

FPR1 の値を x，FPR2 の値を y として，x×y を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LMLT からの戻り番地 2 バイトのみ）

(4) 使用レジスタ

AX，B，DE

(5) 使用ワーク・エリア

FPR1，FPR2，FPR1_X，FPR2_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：134 μs

最大：138 μs（2×2）

(7) 処理手順

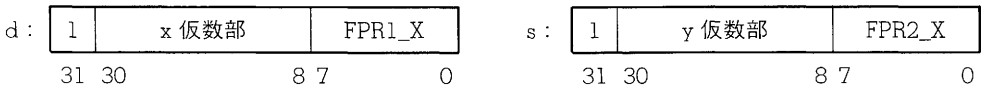
(a) x または y が 0 の場合，0 を返します。

(b) 指数部を加算し，指数部解とします。

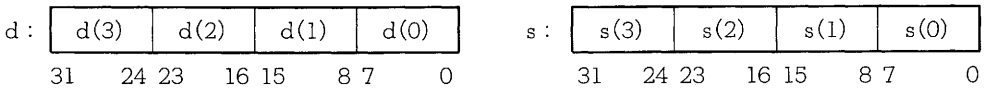
(c) 符号の XOR を取り，結果の符号とします。

(d) 仮数部の乗算を次に示す方法で行います。

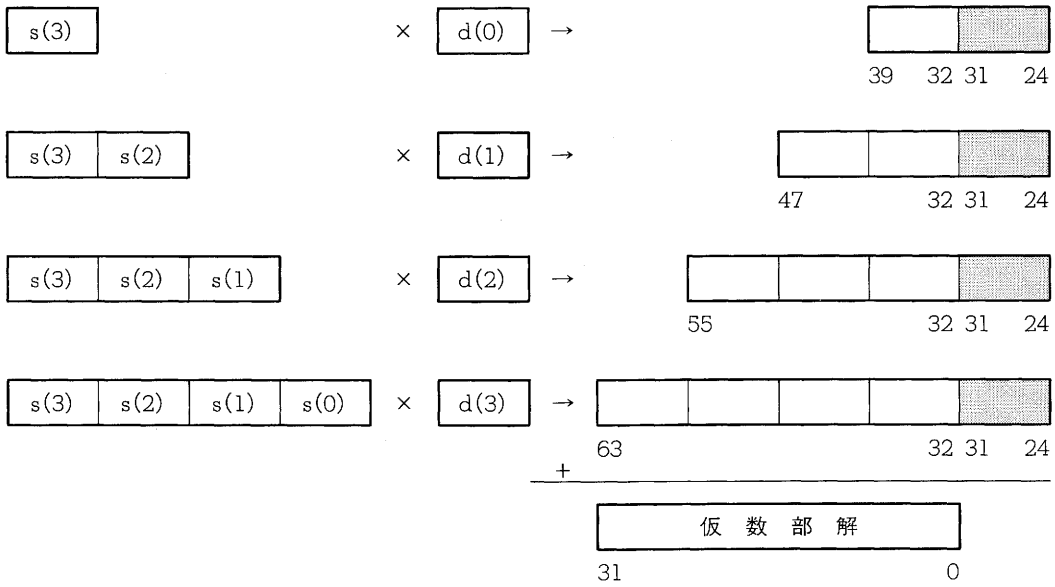
x，y の仮数部にそれぞれ MSB（1）と拡張仮数部（8 ビット）を加え，ダブルワード型の変数 d，s とみなします。



さらに，下図に示すようにd，sをそれぞれ要素数4のBYTE型配列とみなします。

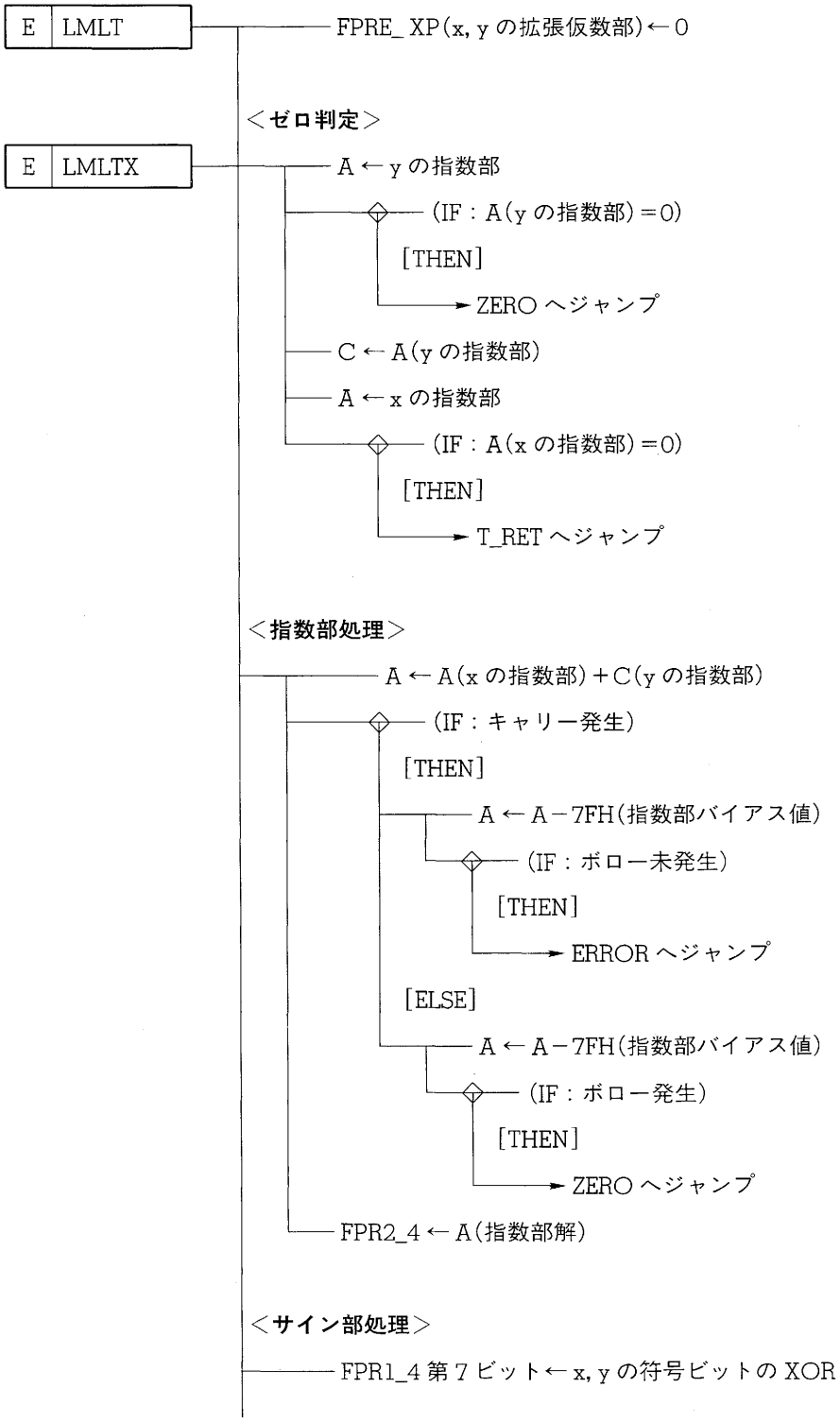


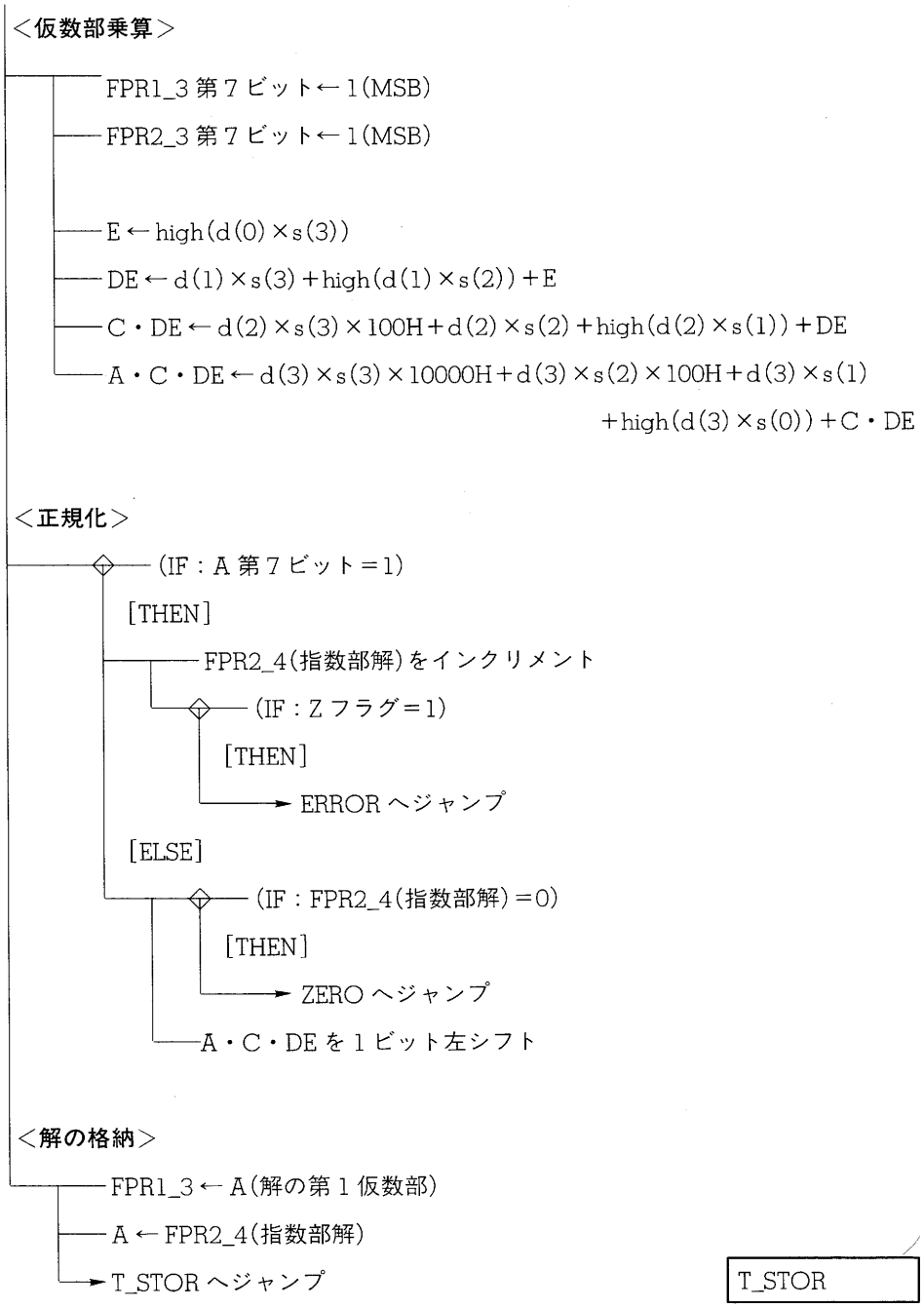
下図に示す手順で乗算結果の上位32ビットを算出します（▒▒の部分は切り捨てています）。



(e) 指数部解と，(d) で求めた仮数部解に対して正規化を行い，符号ビットとともにFPR1に格納します。

(8) 処 理 図


E LMLT
FPRE_XP(x, y の拡張仮数部)←0
＜ゼロ判定＞
E LMLTX
A ← y の指数部
(IF : A(y の指数部)＝0)
[THEN]
ZERO へジャンプ
C ← A(y の指数部)
A ← x の指数部
(IF : A(x の指数部)＝0)
[THEN]
T_RET へジャンプ
＜指数部処理＞
A ← A(x の指数部)＋C(y の指数部)
(IF : キャリー発生)
[THEN]
A ← A－7FH(指数部バイアス値)
(IF : ボロー未発生)
[THEN]
ERROR へジャンプ
[ELSE]
A ← A－7FH(指数部バイアス値)
(IF : ボロー発生)
[THEN]
ZERO へジャンプ
FPR2_4 ← A(指数部解)
＜サイン部処理＞
FPR1_4 第7ビット← x, y の符号ビットの XOR



備考 1. high（　）は，乗算結果の上位 1 バイトを示します。

2. レーベル | E | LMLTX | は数学関数などで，拡張仮数部を使用した乗算を実行するための内部的な広域名です。

3.4　浮動小数点除算（LDIV）

(1) 処理内容

FPR1 の値を x，FPR2 の値を y として，x÷y を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LDIV からの戻り番地 2 バイトのみ）

(4) 使用レジスタ

AX，BC，DE，HL

(5) 使用ワーク・エリア

FPR1，FPR2，FPR1_X，FPR2_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：516 μs

最大：620 μs（1.9999999/1）

(7) 処理手順

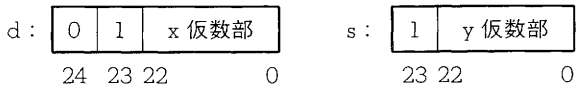
(a) y が 0 の場合，異常終了を，x が 0 の場合，0 を返します。

(b) x の指数部から y の指数部を減算し，指数部解とします。

(c) 符号の XOR を取り，結果の符号とします。

(d) 仮数部の除算を次に示す方法で行います。

x，y の仮数部にそれぞれ MSB（1）を加え，25，24ビットの変数 d，s とみなします。

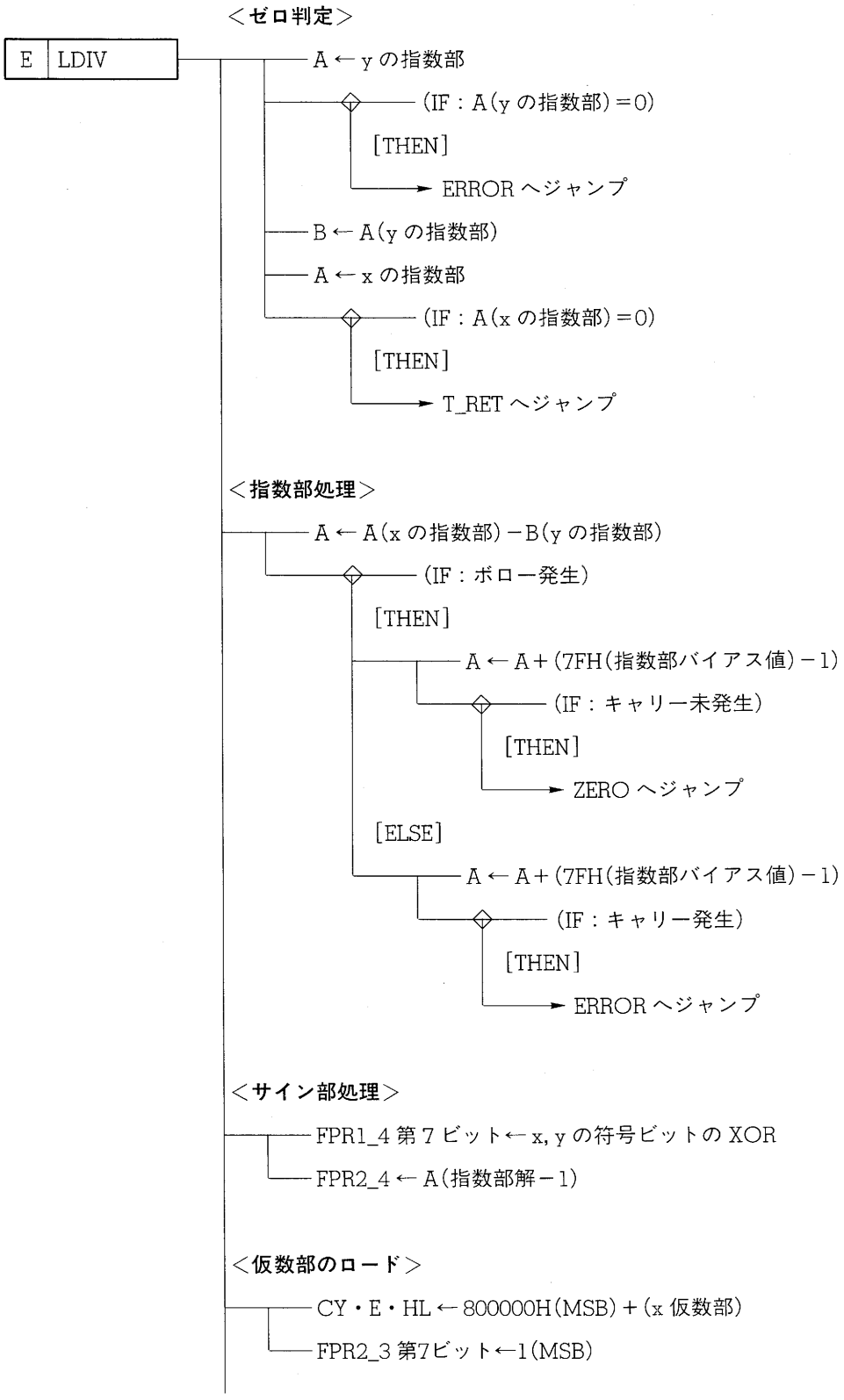


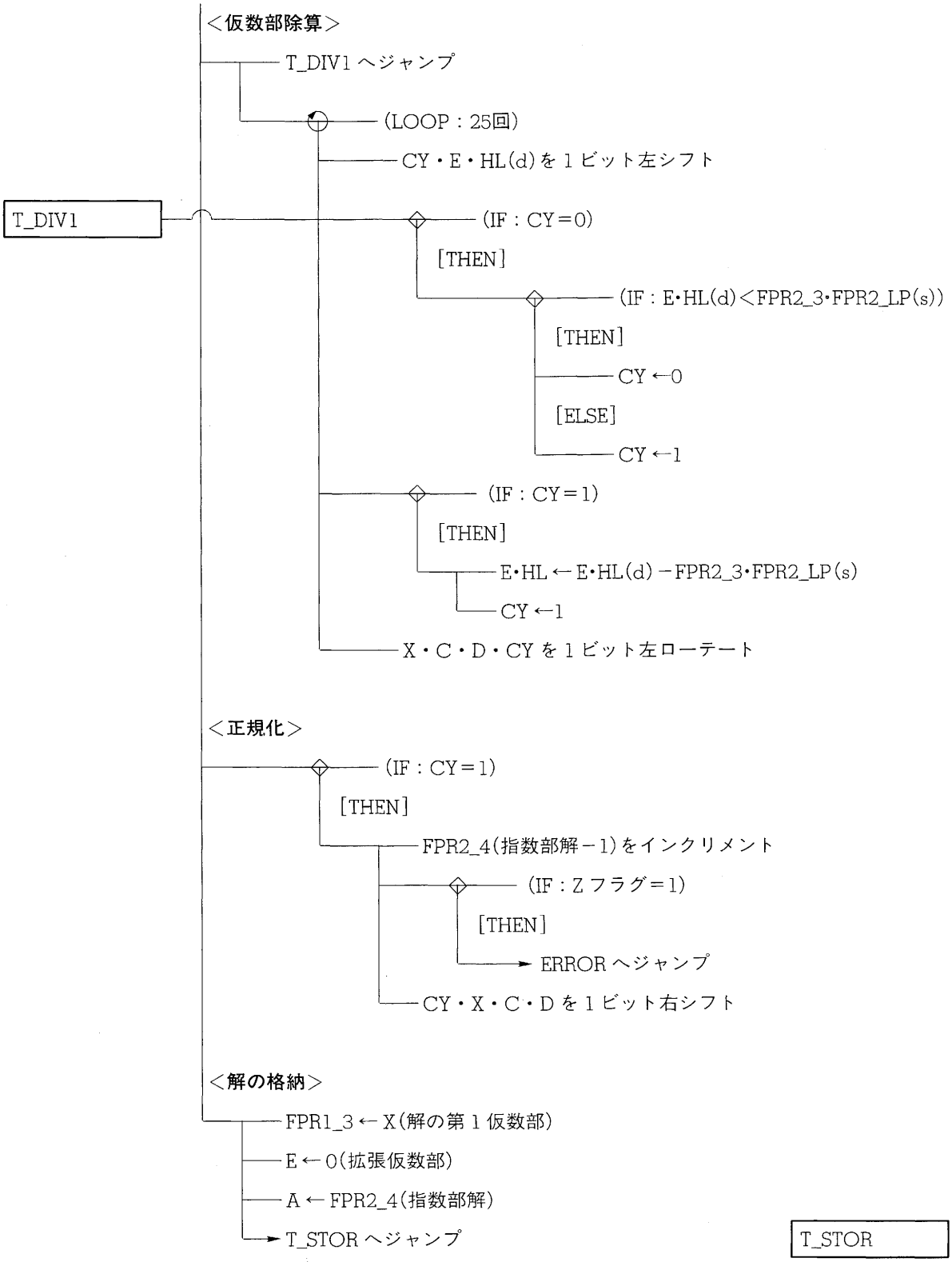
通常の手計算の手順で，商を25ビット算出します。

(i) d と s を比較し，d≧s なら商 1，d＜s なら商 0 を得ます。
(ii) d から s×（商）を引きます。
(iii) d を 1 ビット左シフトします。
(iv) (i) - (iii) を25回繰り返します（ただし，25回目は (iii) を実行しません)。

(e)指数部解と，(d) で求めた仮数部解に対して正規化を行い，符号ビットとともに FPR1 に格納します。

(8) 処 理 図


＜ゼロ判定＞
E
LDIV
A ← y の指数部
(IF : A(y の指数部) = 0)
[THEN]
ERROR へジャンプ
B ← A(y の指数部)
A ← x の指数部
(IF : A(x の指数部) = 0)
[THEN]
T_RET へジャンプ
＜指数部処理＞
A ← A(x の指数部) − B(y の指数部)
(IF : ボロー発生)
[THEN]
A ← A + (7FH(指数部バイアス値) − 1)
(IF : キャリー未発生)
[THEN]
ZERO へジャンプ
[ELSE]
A ← A + (7FH(指数部バイアス値) − 1)
(IF : キャリー発生)
[THEN]
ERROR へジャンプ
＜サイン部処理＞
FPR1_4 第 7 ビット ← x, y の符号ビットの XOR
FPR2_4 ← A(指数部解 − 1)
＜仮数部のロード＞
CY・E・HL ← 800000H(MSB) + (x 仮数部)
FPR2_3 第7ビット ← 1(MSB)


＜仮数部除算＞
T_DIV1 へジャンプ
(LOOP：25回)
CY・E・HL(d)を1ビット左シフト
T_DIV1
(IF：CY＝0)
[THEN]
(IF：E･HL(d)＜FPR2_3･FPR2_LP(s))
[THEN]
CY ←0
[ELSE]
CY ←1
(IF：CY＝1)
[THEN]
E･HL ← E･HL(d) －FPR2_3･FPR2_LP(s)
CY ←1
X・C・D・CY を1ビット左ローテート
＜正規化＞
(IF：CY＝1)
[THEN]
FPR2_4(指数部解－1)をインクリメント
(IF：Z フラグ＝1)
[THEN]
ERROR へジャンプ
CY・X・C・D を1ビット右シフト
＜解の格納＞
FPR1_3 ← X(解の第1仮数部)
E ← 0(拡張仮数部)
A ← FPR2_4(指数部解)
T_STOR へジャンプ
T_STOR

第４章　数学関数

数学関数として，次のものを用意しています。

(1) sin 関数（LSIN）

FPR1 の値の正弦を求めます。

(2) cos 関数（LCOS）

FPR1 の値の余弦を求めます。

(3) tan 関数（LTAN）

FPR1 の値の正接を求めます。

(4) 自然対数関数（LLOG）

FPR1 の値の自然対数をとります。

(5) 常用対数関数（LLOG10）

FPR1 の値の常用対数をとります。

(6) 指数関数（底＝e）（LEXP）

FPR1 の値を指数値，底を e とした指数関数解を求めます。

(7) 指数関数（底＝10）（LEXP10）

FPR1 の値を指数値，底を 10 とした指数関数解を求めます。

(8) べき乗関数（LPOW）

FPR1 の値と FPR2 の値のべき乗を求めます。

(9) 平方根関数（LSQRT）

FPR1 の値の平方根を求めます。

(10) arcsin 関数（LASIN）

FPR1 の値の逆正弦を求めます。

(11) arccos 関数（LACOS）

FPR1 の値の逆余弦を求めます。

(12) arctan 関数（LATAN）

FPR1 の値の逆正接を求めます。

(13) sinh 関数（LHSIN）

FPR1 の値の hyperbolic sine 関数解を求めます。

(14) cosh 関数（LHCOS）

FPR1 の値の hyperbolic cosine 関数解を求めます。

(15) tanh 関数（LHTAN）

FPR1 の値の hyperbolic tangent 関数解を求めます。

(16) 絶対値関数（LABS）

FPR1 の値の絶対値をとります。

(17) 逆数関数（LRCPN）

FPR1 の値の逆数を求めます。

演算結果は，FPR1 に格納されます。

4.1　共通サブルーチン（LPLY，LPLY2）

先に述べたように，平方根関数を除く他の数学関数は，Taylor 近似式，または最良近似式を用いて計算しています。これら近似式は高次の多項式で表され，かつ，Taylor 展開，および最良近似の性格上共通のパターンを持っています。

そこで，2つのタイプの多項式の計算関数（LPLY，LPLY2）を共通サブルーチンとして用意し，使用しています。

(1) 処理内容

多項式の計算結果を，拡張仮数部を含めて，FPR1・FRR1_X へ返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD

(3) 消費スタック・サイズ

4（LPLY，LPLY2 からの戻り番地2バイトを含む）

(4) 使用レジスタ

AX，BC，DE，HL

(5) 使用ワーク・エリア

FPR1，FPR2，FPR3，FPR4，FPR1_X，FPR2_X，FPR3_X，FPR4_X（FPR4，FPR4_X の内容は保存されます）

(6) アルゴリズム

次の2つのタイプの多項式計算が可能です。

① $x+k_1xy+k_1k_2xy^2+k_1k_2k_3xy^3+\cdots+k_1k_2\cdots k_nxy^n$

② $z+k_1xy+k_1k_2xy^2+k_1k_2k_3xy^3+\cdots+k_1k_2\cdots k_nxy^n$

備考　x　：多項式第1項

y　：各項の次数変化に対する乗算定数（x^2 など）

$(k_1, k_1k_2, \cdots, k_1k_2\cdots k_n)$　：各項の係数

多項式	使　用　条　件	計算関数
①	各項が共通約数 x を持っている場合	LPLY
②	第1項が定数（z）の場合	LPLY2

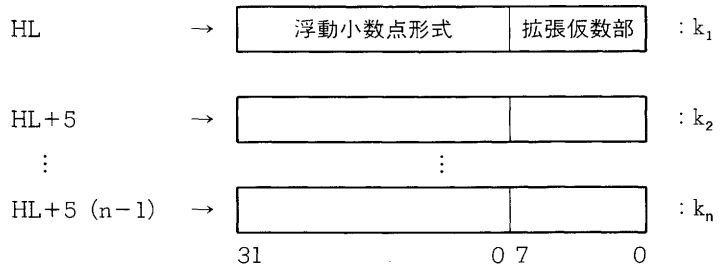
誤差を最小限におさえるために，x，y，z，k_1，k_2，・・・k_nは拡張仮数部を持つ浮動小数点数を使用しています。

注意　Taylor近似の性格上，｜第１項｜＞｜第２項｜＞・・・＞｜第ｎ項｜が成立している必要があります。

(7) 入力条件

多項式	x	y	z	n	係数列（k_1，k_2，k_3，・・・，k_n）の先頭アドレス
①	FPR1・FPR1_X	FPR4・FPR4_X	――――	B	HL
②	FPR3・FPR3_X	FPR4・FPR4_X	FPR1・FPR1_X	B	HL

係数列の格納形式は下図のとおりです。



(8) 処理手順

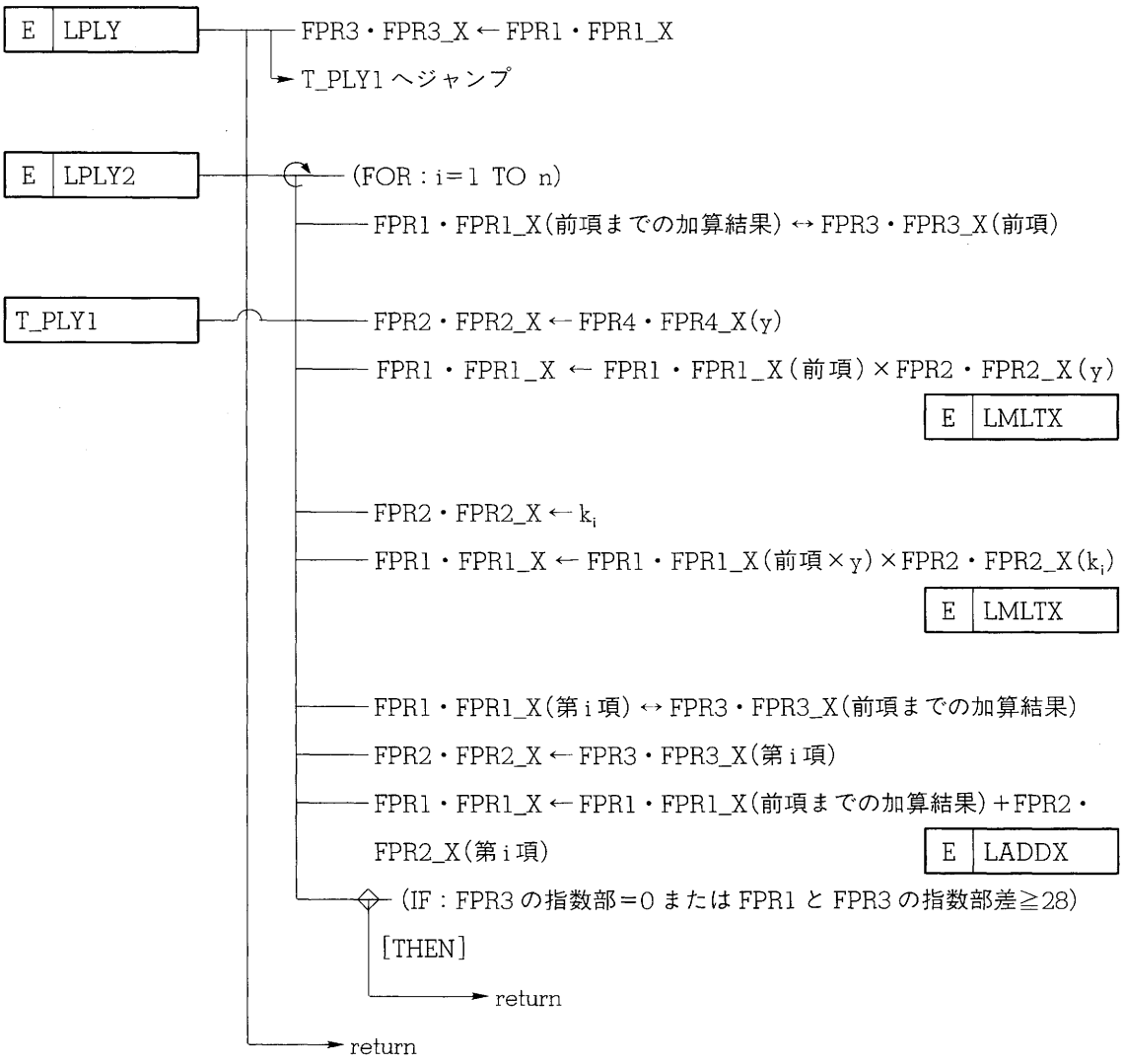
(a) 第i項を第i−1項とy，k_iの乗算によって求めます。

(b) 第i−1項までの和に第i項の値を加算します。

(c) 第i項が第i項までの和に比して十分に小さければ，計算を終了します。

(d) (a) ～ (c) を第ｎ項まで繰り返します。

(9) 処 理 図


E LPLY
FPR3・FPR3_X ← FPR1・FPR1_X
T_PLY1 へジャンプ
E LPLY2
(FOR : i=1 TO n)
FPR1・FPR1_X(前項までの加算結果) ↔ FPR3・FPR3_X(前項)
T_PLY1
FPR2・FPR2_X ← FPR4・FPR4_X(y)
FPR1・FPR1_X ← FPR1・FPR1_X(前項)×FPR2・FPR2_X(y)
E LMLTX
FPR2・FPR2_X ← k_i
FPR1・FPR1_X ← FPR1・FPR1_X(前項×y)×FPR2・FPR2_X(k_i)
E LMLTX
FPR1・FPR1_X(第i項) ↔ FPR3・FPR3_X(前項までの加算結果)
FPR2・FPR2_X ← FPR3・FPR3_X(第i項)
FPR1・FPR1_X ← FPR1・FPR1_X(前項までの加算結果)＋FPR2・FPR2_X(第i項)
E LADDX
(IF : FPR3 の指数部＝0 または FPR1 と FPR3 の指数部差≧28)
[THEN]
return
return

4.2　sin 関数（LSIN）

(1) 処理内容

FPR1 の値を x として，sin(x) を FPR1 に返します。

●単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LSIN, FTOL, LTOF

(3) 消費スタック・サイズ

6（LSIN からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：2343 μs

最大：8005 μs（sin(6.8056469e+38)）

(7) アルゴリズム

次の Taylor 近似式を用いて求めます。

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!}$$

(8) 処理手順

(a) x の符号を記憶し，x の絶対値をとります。

(b) x を $0 \leqq x < \pi/2$ の範囲にスケーリングします。

スケーリング後の値を x' とすると，次のようになります。

$$\sin(\pi/2+x') = \sin(\pi/2-x')$$
$$\sin(\pi+x') = \sin(-x')$$
$$\sin(3\pi/2+x') = \sin(x'-\pi/2)$$

したがって，x を $\pi/2$ で割った商を n，余りを x' とすると，sin(x) は次のように置き換えることができます。

n/4 の余り	sin(x)	x"
0	sin(x')	x'
1	sin($\pi/2-$x')	$\pi/2-$x'
2	sin(−x')	−x'
3	sin(x'−$\pi/2$)	x'−$\pi/2$

(c) 記憶していた符号を x" に掛けます。

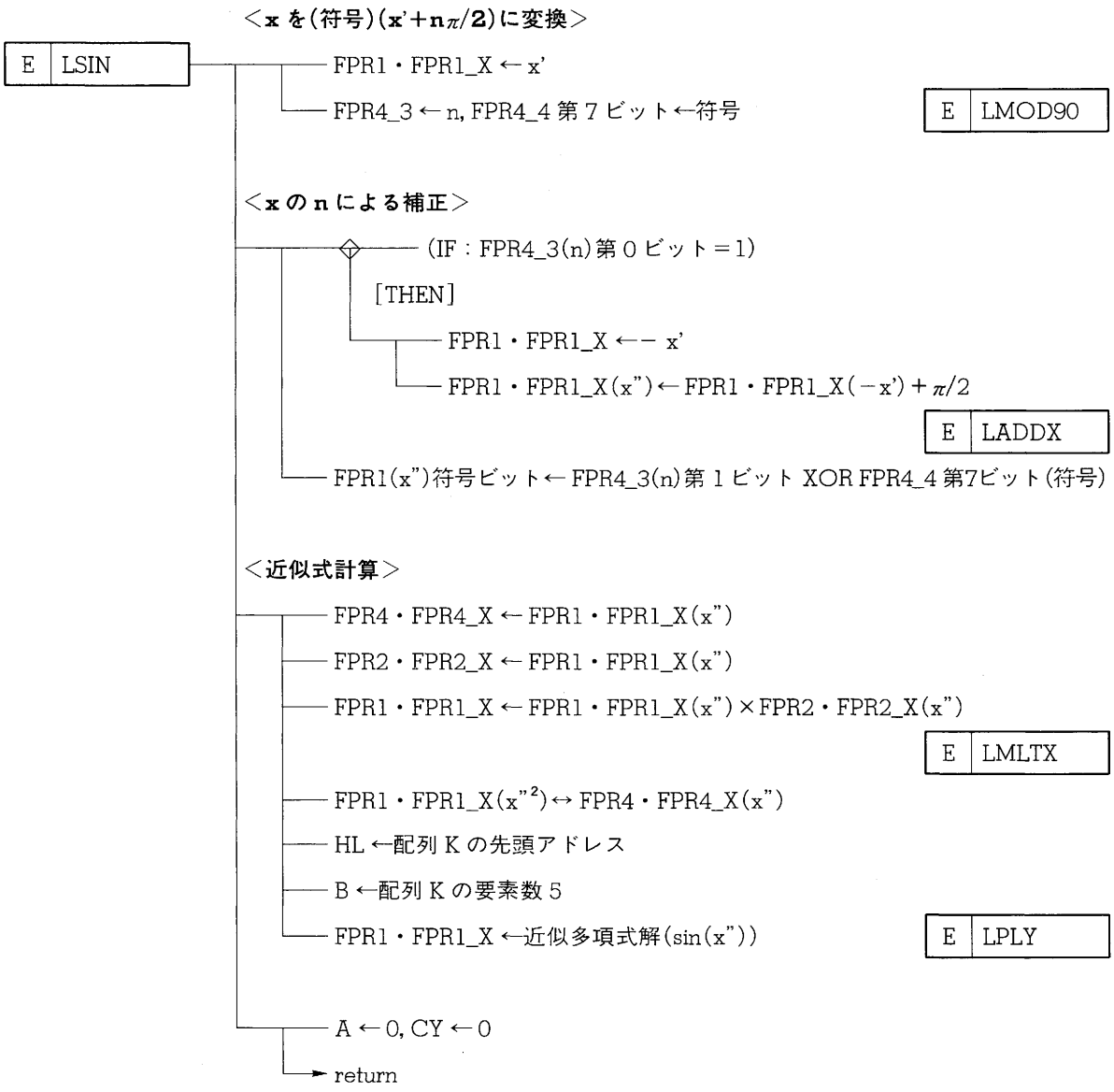
(d) sin(x") を Taylor 近似式により求め，解とします。

(9) 浮動小数点定数データ

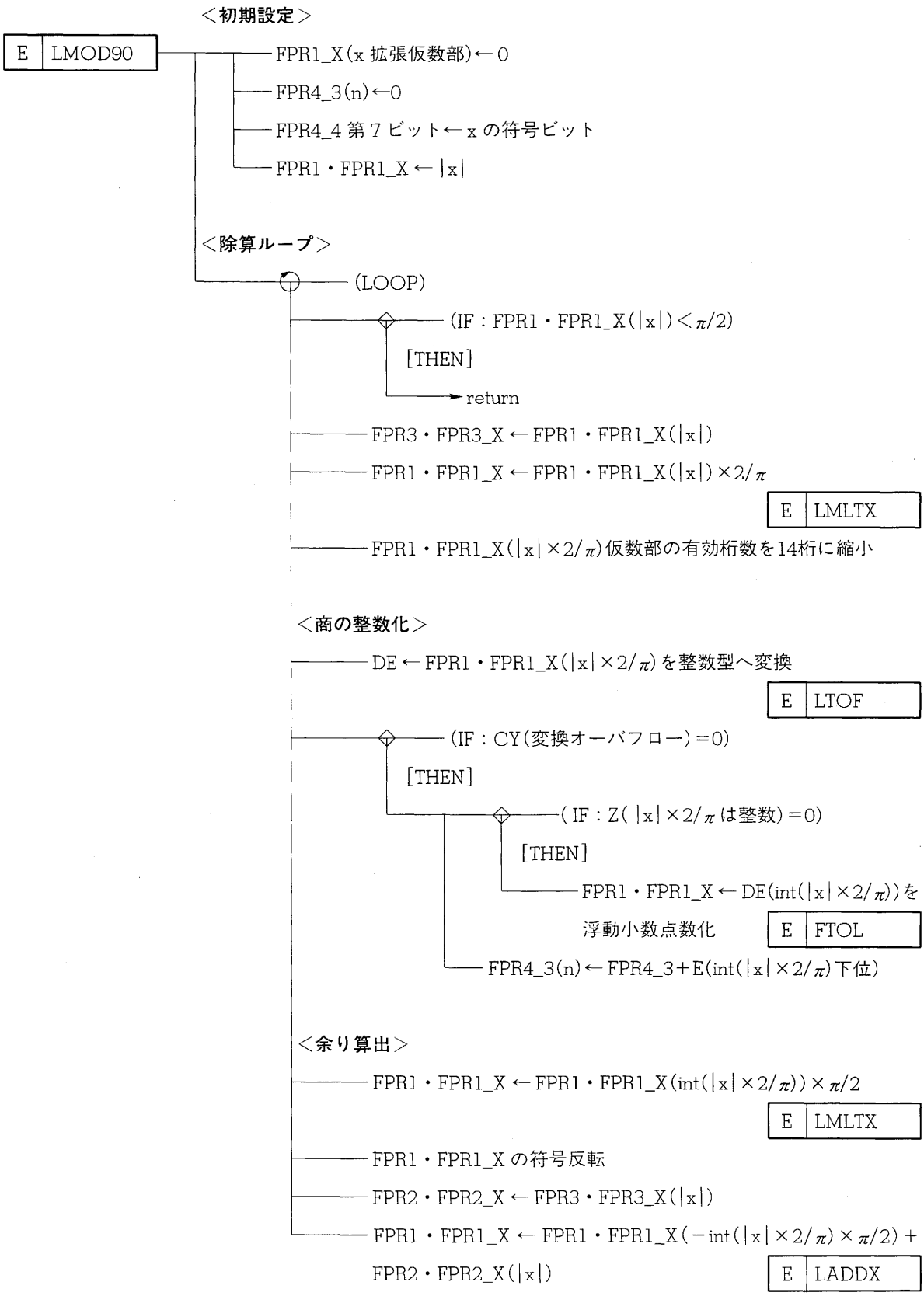
(a) 拡張仮数部を持つ定数データ $2/\pi$，および $\pi/2$ を用いています。

(b) 近似多項式の係数列に対して，拡張仮数部を持つ定数データ −1/3!，−3!/5!，−5!/7!，−7!/9!，−9!/11! を，要素数 5 の配列 K として使用しています。

(10) 処 理 図


＜x を(符号)(x'+nπ/2)に変換＞
E | LSIN
FPR1・FPR1_X ← x'
FPR4_3 ← n, FPR4_4 第7ビット←符号
E | LMOD90
＜x の n による補正＞
(IF：FPR4_3(n)第0ビット＝1)
[THEN]
FPR1・FPR1_X ←− x'
FPR1・FPR1_X(x") ← FPR1・FPR1_X(−x') + π/2
E | LADDX
FPR1(x")符号ビット ← FPR4_3(n)第1ビット XOR FPR4_4 第7ビット(符号)
＜近似式計算＞
FPR4・FPR4_X ← FPR1・FPR1_X(x")
FPR2・FPR2_X ← FPR1・FPR1_X(x")
FPR1・FPR1_X ← FPR1・FPR1_X(x") × FPR2・FPR2_X(x")
E | LMLTX
FPR1・FPR1_X(x"²) ↔ FPR4・FPR4_X(x")
HL ←配列 K の先頭アドレス
B ←配列 K の要素数 5
FPR1・FPR1_X ←近似多項式解(sin(x"))
E | LPLY
A ← 0, CY ← 0
return

（π/2 による除算商，および余り算出サブルーチン）



備考　int(x)は x の整数部を示します。

4.3　cos 関数（LCOS）

(1) 処理内容

FPR1 の値を x として，cos(x) を FPR1 に返します。

●単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LSIN, LCOS, FTOL, LTOF

(3) 消費スタック・サイズ

6（LCOS からの戻り番地２バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：2842 μs

最大：7804 μs（cos(6.8056469e＋38)）

(7) アルゴリズム

次の式により，cos(x) を求めます。

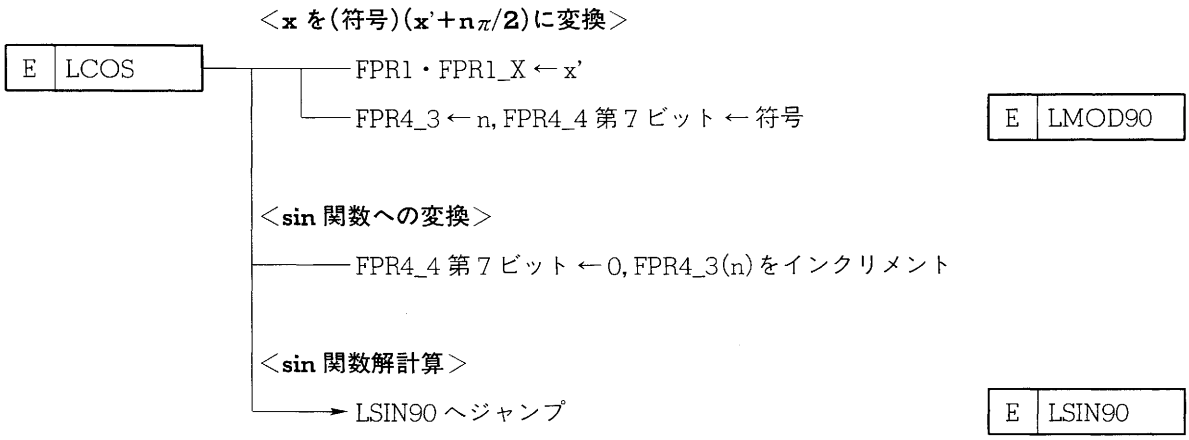
$$\cos(x) = \sin(|x| + \pi/2)$$

(8) 処理手順

(a) LMOD90 サブルーチンにより，|x| の $\pi/2$ による除算商（n）と余り（x'）を求めます。

(b) LSIN90 サブルーチンにより，$\sin(x' + (n+1)\pi/2)$ を求め解とします。

(9) 処 理 図


＜x を(符号)(x'+nπ/2)に変換＞
E
LCOS
FPR1・FPR1_X ← x'
FPR4_3 ← n, FPR4_4 第7ビット ← 符号
E
LMOD90
＜sin 関数への変換＞
FPR4_4 第7ビット ← 0, FPR4_3(n) をインクリメント
＜sin 関数解計算＞
LSIN90 へジャンプ
E
LSIN90

4.4　tan 関数（LTAN）

(1) 処理内容

FPR1 の値を x として，tan(x) を FPR1 に返します。

●単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LSIN, LCOS, LTAN, FTOL, LTOF

(3) 消費スタック・サイズ

10（LTAN からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR5, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：5428 μs

最大：11040 μs（tan(6.8056469e＋38)）

(7) アルゴリズム

次の式により求めます。

$$\tan(x) = \frac{\sin(x)}{\cos(|x|)}$$

(8) 処理手順

(a) LMOD90 サブルーチンにより，|x| の $\pi/2$ による除算商（n）と余り（x'），および x の符号（s）を得ます。

(b) LSIN90 サブルーチンにより，(s)sin(x'＋nπ/2) を求めます。

(c) LCOS90 サブルーチンにより，cos(x'＋nπ/2) を求めます。

(d) (s)sin(x'＋nπ/2)÷cos(x'＋nπ/2) を求め解とします。

(9) 処 理 図


＜x を(符号)(x'+nπ/2)に変換＞
E LTAN
FPR1・FPR1_X ← x'
FPR4_3 ← n, FPR4_4 第7ビット←符号
E LMOD90
FPR4_3(n), FPR4_4 第7ビット(符号)をスタックへ退避
FPR5・FPR5_X ← FPR1・FPR1_X(x')
＜cos 関数解計算＞
FPR1 ← cos(FPR1・FPR1_X(x')+FPR4_3(n)π/2)
E LCOS90
＜sin 関数解計算＞
FPR1(cos(|x|)) ↔ FPR5(x')
FPR1_X ← FPR5_X(x' 拡張仮数部)
FPR4_4 第7ビット←スタックより符号を復帰,
FPR4_3 ←スタックより n を復帰
FPR1 ← FPR4_4 第7ビット(符号)・sin(FPR1・FPR1_X(x')+FPR4_3(n)π/2)
E LSIN90
＜tan 関数解計算＞
FPR2 ← FPR5(cos(|x|))
LDIV へジャンプ
E LDIV

4.5　自然対数関数（LLOG）

(1) 処理内容

FPR1 の値を x として，log(x) を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LLOG, FTOL

(3) 消費スタック・サイズ

6（LLOG からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：2857 μs

最大：3659 μs（log(1.4397301e−25)）

(7) アルゴリズム

log(x) は次の式に Taylor 展開することができます。

$$x' = \frac{x-1}{x+1} \quad \text{とおく}$$

$$\log(x) = \log(x'+1) - \log(1-x')$$

$$= 2x' + \frac{2}{3}x'^{3} + \frac{2}{5}x'^{5} + \frac{2}{7}x'^{7} + \frac{2}{9}x'^{9}$$

この式は，$0 < x < \infty$ に対して理論的に成立しますが，x＝1の周辺以外では収束が遅く，実用になりません。

そのため次に示す方法で，近似式で用いる x' の範囲を約 −0.17〜0.17 にします。

- x の指数部＝xe，仮数部＝xf とおく
- xf＜$\sqrt{2}$ の場合，xe'＝xe，xf'＝xf とする
- xf≧$\sqrt{2}$ の場合，xe'＝xe＋1，xf'＝xf /2 とする
- 次式に関数変換する

$$\log(x)=xe'\times\log 2+\log(xf')$$

(8) 処理手順

(a) x≦0の場合，異常終了します。

(b) xf＜$\sqrt{2}$ の場合，xe'＝xe，xf'＝xf とします。

(c) xf≧$\sqrt{2}$ の場合，xe'＝xe＋1，xf'＝xf/2 とします。

(d) xe'×log2 を求めます。

(e) x' を（xf'－1）/（xf'＋1）により求めます。

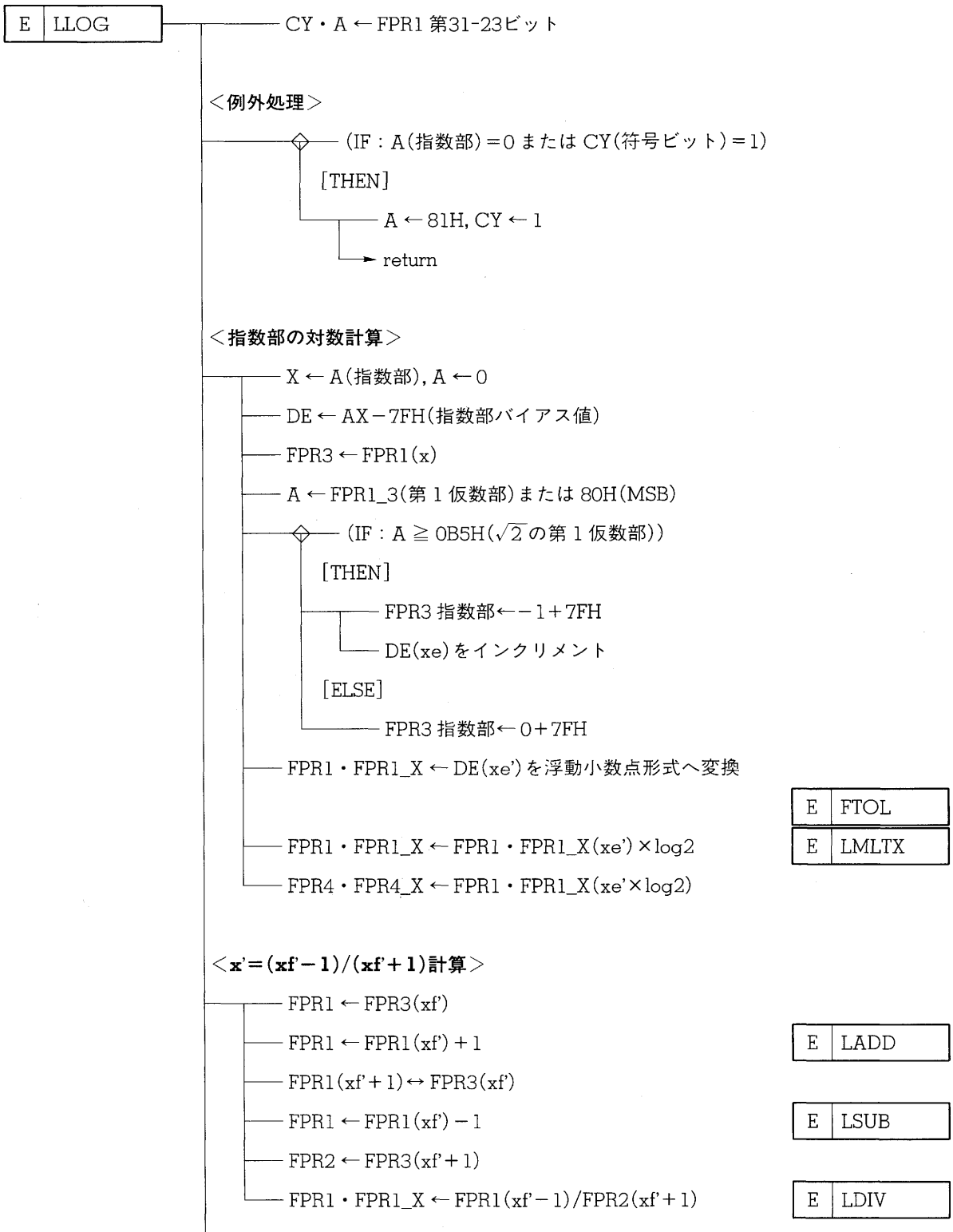
(f) 近似多項式の第1項（2x'）を xe'×log2＋2x' に置き換え，近似多項式を計算します。

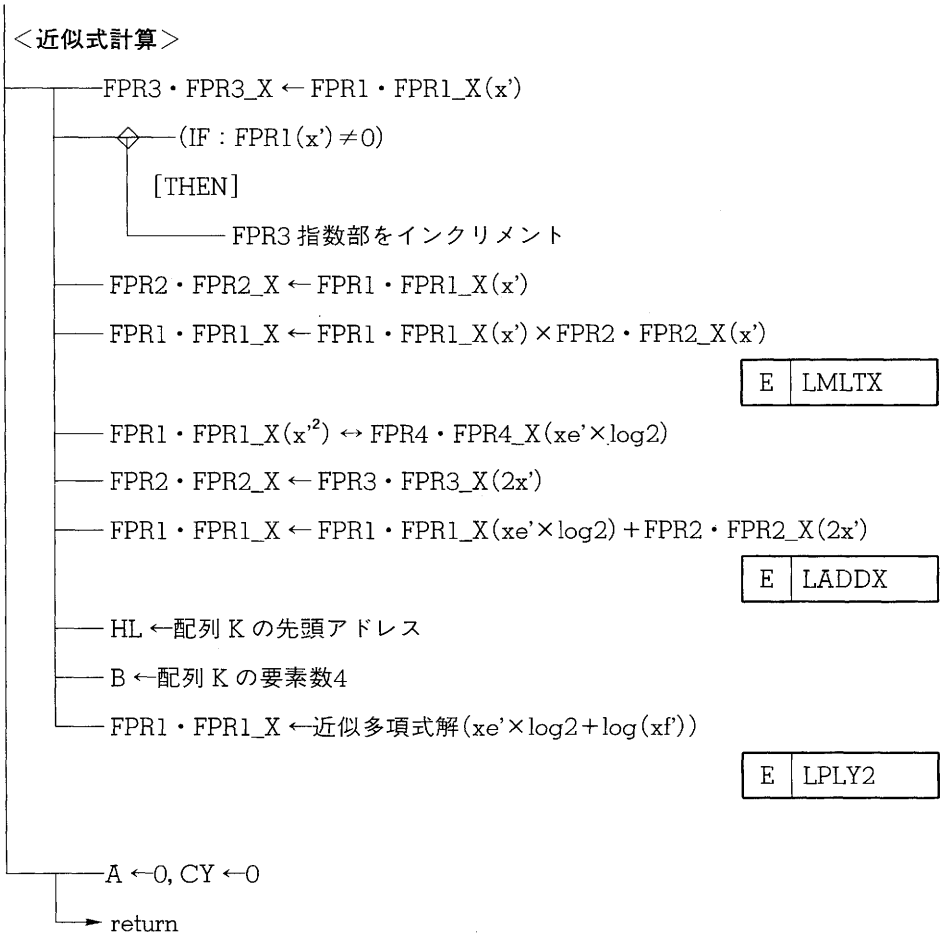
(9) 浮動小数点定数データ

(a) 拡張仮数部を持つ定数データ log2，および1を使用しています。

(b) 近似多項式の係数列に対して，拡張仮数部を持つ定数データ 1/3，3/5，5/7，7/9 を要素数4の配列 K として使用しています。

(10) 処 理 図


E LLOG
CY・A ← FPR1 第31-23ビット
＜例外処理＞
(IF：A(指数部)＝0 または CY(符号ビット)＝1)
[THEN]
A ← 81H, CY ← 1
return
＜指数部の対数計算＞
X ← A(指数部), A ← 0
DE ← AX－7FH(指数部バイアス値)
FPR3 ← FPR1(x)
A ← FPR1_3(第1仮数部) または 80H(MSB)
(IF：A ≧ 0B5H(√2 の第1仮数部))
[THEN]
FPR3 指数部← －1＋7FH
DE(xe)をインクリメント
[ELSE]
FPR3 指数部← 0＋7FH
FPR1・FPR1_X ← DE(xe')を浮動小数点形式へ変換
E FTOL
FPR1・FPR1_X ← FPR1・FPR1_X(xe')×log2
E LMLTX
FPR4・FPR4_X ← FPR1・FPR1_X(xe'×log2)
＜x'＝(xf'－1)/(xf'＋1)計算＞
FPR1 ← FPR3(xf')
FPR1 ← FPR1(xf')＋1
E LADD
FPR1(xf'＋1)↔ FPR3(xf')
FPR1 ← FPR1(xf')－1
E LSUB
FPR2 ← FPR3(xf'＋1)
FPR1・FPR1_X ← FPR1(xf'－1)/FPR2(xf'＋1)
E LDIV


＜近似式計算＞
FPR3・FPR3_X ← FPR1・FPR1_X(x')
(IF : FPR1(x') ≠ 0)
[THEN]
FPR3 指数部をインクリメント
FPR2・FPR2_X ← FPR1・FPR1_X(x')
FPR1・FPR1_X ← FPR1・FPR1_X(x') × FPR2・FPR2_X(x')
E
LMLTX
FPR1・FPR1_X(x'^2) ↔ FPR4・FPR4_X(xe' × log2)
FPR2・FPR2_X ← FPR3・FPR3_X(2x')
FPR1・FPR1_X ← FPR1・FPR1_X(xe' × log2) + FPR2・FPR2_X(2x')
E
LADDX
HL ←配列 K の先頭アドレス
B ←配列 K の要素数4
FPR1・FPR1_X ←近似多項式解(xe' × log2 + log(xf'))
E
LPLY2
A ←0, CY ←0
return

4.6　常用対数関数（LLOG10）

(1) 処理内容

FPR1 の値を x として，$\log_{10}(x)$ を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LLOG, LLOG10, FTOL

(3) 消費スタック・サイズ

8（LLOG10 からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：3014 μs

最大：3801 μs（$\log_{10}(2.4001264e-18)$）

(7) アルゴリズム

次式により求めます。

$$\log_{10}(x) = \frac{\log(x)}{\log 10}$$

(8) 処理手順

(a) LLOG 関数により log(x) を求めます。

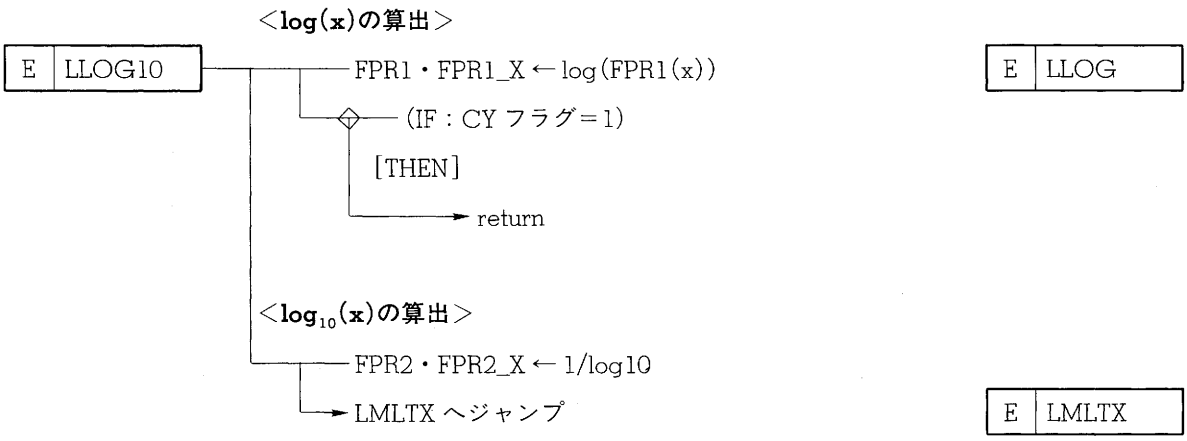
(b) LLOG 関数が異常終了した場合，そのまま異常終了します。

(c) log(x)/log10 を求め，解とします。

(9) 浮動小数点定数データ

拡張仮数部を持つ定数データ 1/log10 を使用しています。

(10) 処 理 図


＜log(x)の算出＞
E
LLOG10
FPR1・FPR1_X ← log(FPR1(x))
(IF：CY フラグ＝1)
[THEN]
return
＜log10(x)の算出＞
FPR2・FPR2_X ← 1/log10
LMLTX へジャンプ
E
LLOG
E
LMLTX

4.7　指数関数（底＝e）（LEXP）

(1) 処理内容

FPR1 の値を x として，e^x を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, FTOL, LTOF

(3) 消費スタック・サイズ

6（LEXP からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：3591 μs

最大：4084 μs（$e^{-0.82129019}$）

(7) アルゴリズム

指数底を 2 に変換することにより，先に解の指数部を求める方法をとっています。

$$e^x = 2^{x/\log 2} = 2^{\mathrm{floor}(x/\log 2)} \times 2^{\mathrm{dec}(x/\log 2)}$$

備考 1. floor(x) は，x の下位整数（floor(x)≦x＜floor(x)＋1）を示します。

2. dec(x) は，x の小数部（dec(x)＝x－floor(x)；0≦dec(x)＜1）を示します。

$1 \leqq 2^{\mathrm{dec}(x/\log 2)} < 2$ となるため，floor(x/log2) は解の指数部となります。

仮数部は，次の Taylor 近似式により求めます。

dec(x/log2)＜1/2 の場合，x'＝dec(x/log2)

dec(x/log2)≧1/2 の場合，x'＝dec(x/log2)－1

$$2^{x'} = 1 + \frac{\log 2}{1!}x' + \frac{(\log 2)^2}{2!}x'^2 + \frac{(\log 2)^3}{3!}x'^3 + \cdots + \frac{(\log 2)^7}{7!}x'^7$$

備考 1. $2^{x'}$ に 1/2 をかけても得られる仮数部は同じため，近似式に最も有利な $-1/2 \leqq x' < 1/2$ の範囲を使っています。

2. dec(x/log2)≧1/2 の場合において数学的には x'<0 ですが，計算誤差により x'=0 となり得ます。この場合，$2^{x'}=1$ となり，期待とまるで異なった仮数部が得られることになります。

対策として，x' の計算には以下の式を使用しています。

dec(x/log2)≧1/2 の場合，$x'=dec(x/log2)-(1+2^{-30})$

(8) 処理手順

(a) x/log2 を求めます。

(b) オーバフローの場合，

- x<0 なら 0 を解として演算を終了します。
- x>0 なら異常終了します。

(c) floor(x/log2) を求めます。

(d) floor(x/log2)<−126 なら 0 を解として演算を終了します。

floor(x/log2)>128 なら異常終了します。

(e) dec(x/log2) を求め x' とします。

(f) x'≧1/2 なら，x'=x'−1 とします。

(g) $2^{x'}$ の Taylor 近似式を計算します。

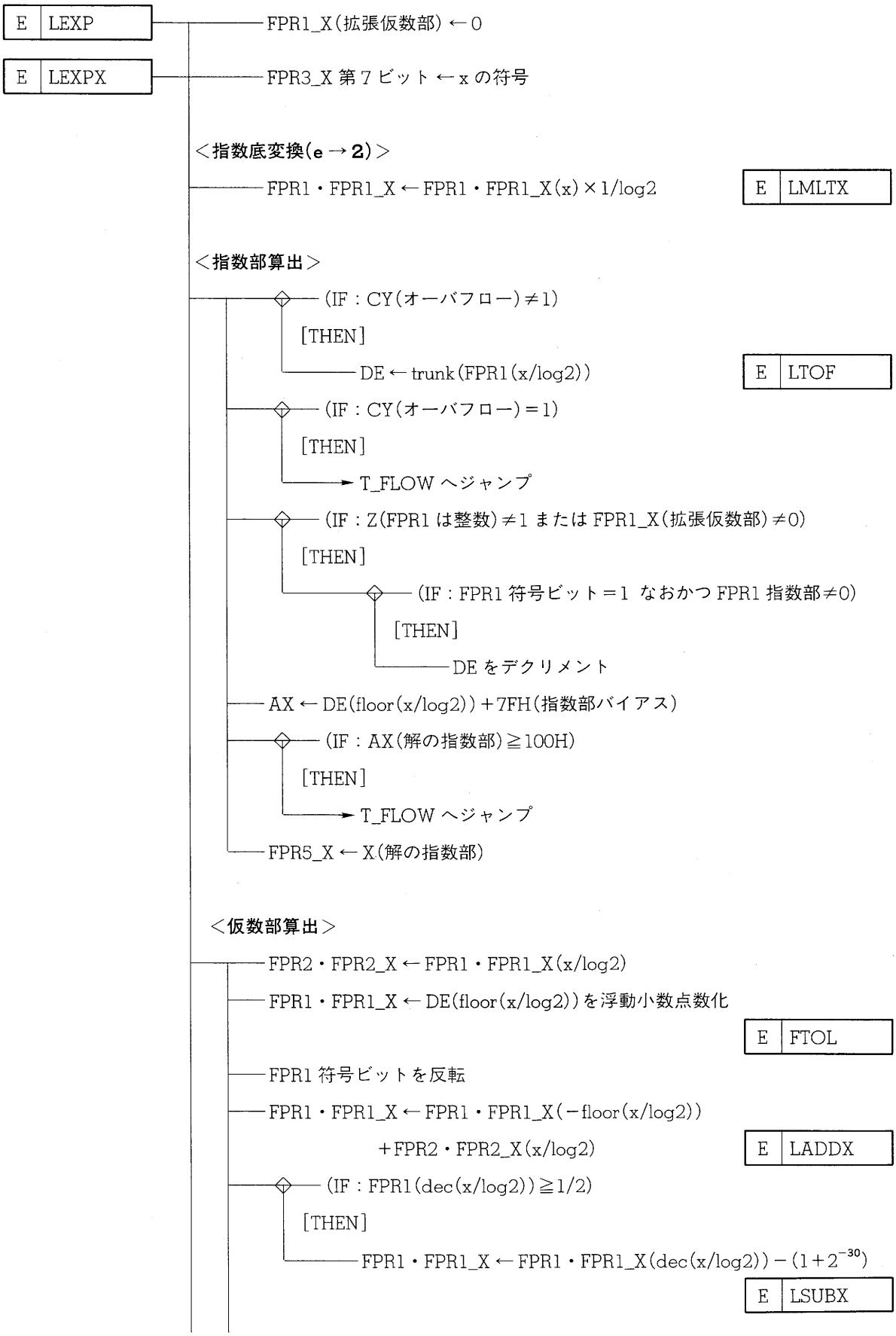
(h) 近似式の計算結果に指数部値 floor(x/log2) を埋め込み解とします。

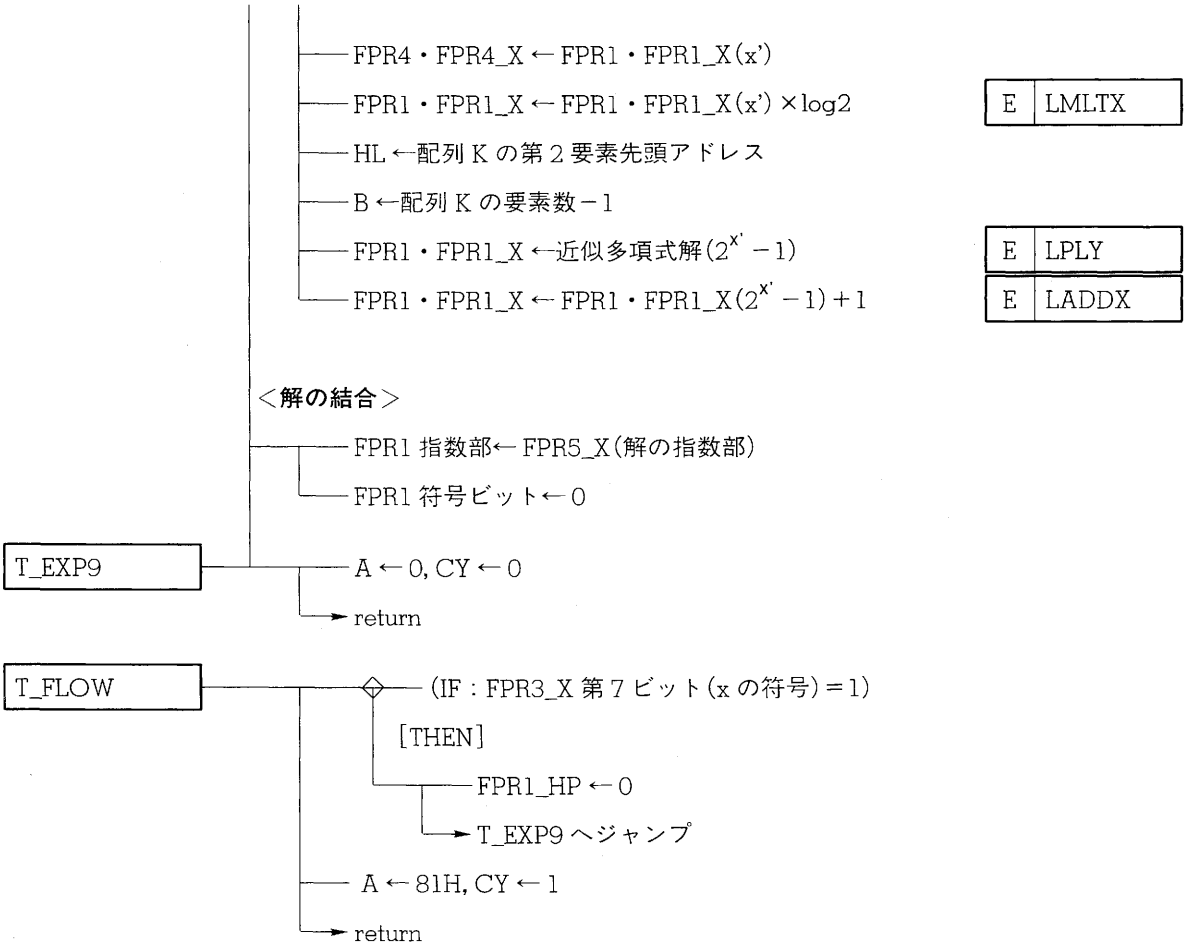
(9) 浮動小数点定数データ

(a) 拡張仮数部を持つ定数データ 1/log2，および 1 を使用しています。

(b) 近似多項式の係数列に対して，拡張仮数部を持つ定数データ log2，log2/2，log2/3，・・・，log2/7 を要素数 7 の配列 K として使用しています。

(10) 処 理 図


E LEXP
E LEXPX
FPR1_X(拡張仮数部) ← 0
FPR3_X 第7ビット ← x の符号
＜指数底変換(e → 2)＞
FPR1・FPR1_X ← FPR1・FPR1_X(x) × 1/log2
E LMLTX
＜指数部算出＞
(IF : CY(オーバフロー) ≠ 1)
[THEN]
DE ← trunk(FPR1(x/log2))
E LTOF
(IF : CY(オーバフロー) = 1)
[THEN]
T_FLOW へジャンプ
(IF : Z(FPR1 は整数) ≠ 1 または FPR1_X(拡張仮数部) ≠ 0)
[THEN]
(IF : FPR1 符号ビット = 1 なおかつ FPR1 指数部 ≠ 0)
[THEN]
DE をデクリメント
AX ← DE(floor(x/log2)) + 7FH(指数部バイアス)
(IF : AX(解の指数部) ≧ 100H)
[THEN]
T_FLOW へジャンプ
FPR5_X ← X(解の指数部)
＜仮数部算出＞
FPR2・FPR2_X ← FPR1・FPR1_X(x/log2)
FPR1・FPR1_X ← DE(floor(x/log2)) を浮動小数点数化
E FTOL
FPR1 符号ビットを反転
FPR1・FPR1_X ← FPR1・FPR1_X(−floor(x/log2))
+FPR2・FPR2_X(x/log2)
E LADDX
(IF : FPR1(dec(x/log2)) ≧ 1/2)
[THEN]
FPR1・FPR1_X ← FPR1・FPR1_X(dec(x/log2)) − (1 + 2^-30)
E LSUBX



備考 1. trunk (x) は x の小数部を 0 に向かって丸めることを示します。

$$\begin{bmatrix} x \geqq 0 \text{ では, } \mathrm{trunk}(x) \leqq x < \mathrm{trunk}(x)+1 \\ x < 0 \text{ では, } \mathrm{trunk}(x)-1 < x \leqq \mathrm{trunk}(x) \end{bmatrix}$$

2. レーベル E LEXPX は，他の数学関数などで，拡張仮数部を使用した指数計算を実行するための内部的な広域名です。

4.8　指数関数（底＝10）（LEXP10）

(1) 処理内容

FPR1 の値を x として，10^x を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, LEXP10, FTOL, LTOF

(3) 消費スタック・サイズ

6（LEXP10 からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：3807 μs

最大：4241 μs（$10^{-0.35975304}$）

(7) アルゴリズム

次の式により求めます。

$$10^x = e^{x\log 10}$$

(8) 処理手順

(a) x×log10 を求めます。

(b) 乗算結果がオーバフローの場合，

x＜0 なら 0 を解として演算を終了します。

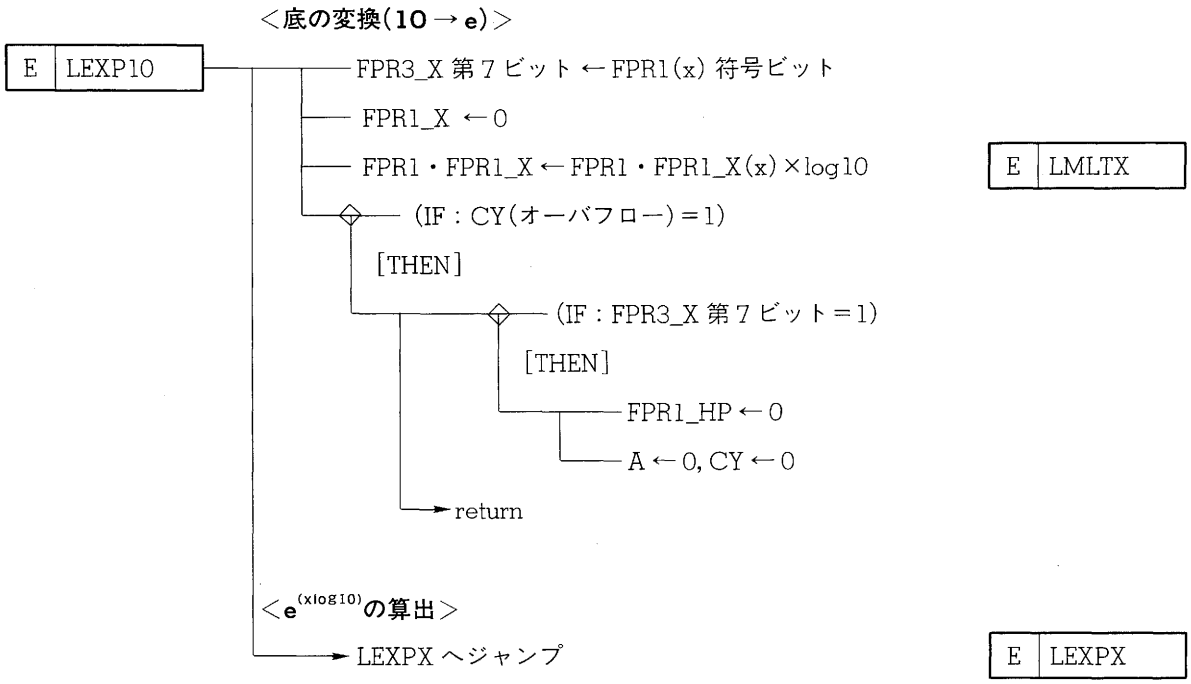
x＞0 ならそのまま異常終了します。

(c) LEXP 関数へジャンプします。

(9) 浮動小数点定数データ

拡張仮数部を持つ定数データ log10 を使用しています。

(10) 処 理 図


＜底の変換(10→e)＞
E LEXP10
FPR3_X 第7ビット←FPR1(x) 符号ビット
FPR1_X ←0
FPR1・FPR1_X←FPR1・FPR1_X(x)×log10
E LMLTX
(IF：CY(オーバフロー)＝1)
[THEN]
(IF：FPR3_X 第7ビット＝1)
[THEN]
FPR1_HP←0
A←0, CY←0
return
＜e^(xlog10)の算出＞
LEXPX へジャンプ
E LEXPX

4.9　べき乗関数（LPOW）

(1) 処理内容

FPR1の値をa，FPR2の値をbとして，a^b をFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LLOG，LEXP，LPOW，FTOL，LTOF

(3) 消費スタック・サイズ

8（LPOWからの戻り番地2バイトを含む）

(4) 使用レジスタ

AX，BC，DE，HL

(5) 使用ワーク・エリア

FPR1，FPR2，FPR3，FPR4，FPR5，FPR1_X，FPR2_X，FPR3_X，FPR4_X，FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：6672 μs

最大：7835 μs（$(1.50487e+12)^{(-0.20180109)}$）

(7) アルゴリズム

a，b数値の組み合わせにより計算方法が異なります。

a，b		a^b
a＝0，b≦0		エラー
a＝0，b＞0		0
a＞0		$e^{b\log(a)}$
a＜0，b＝0		1
a＜0，bは0以外の整数	bは偶数：	$e^{b\log(\lvert a\rvert)}$
	bは奇数：	$-e^{b\log(\lvert a\rvert)}$
a＜0，bは整数ではない		エラー

(8) 処理手順

(a) a＝0 かつ b≦0 なら，異常終了します。

(b) a＝0 かつ b＞0 なら，演算結果として 0 を返します。

(c) a＜0 かつ b＝0 なら，演算結果として 1 を返します。

(d) a＜0 かつ b≠0 なら，b の整数判定，および整数の場合，偶奇数判定を行います。
　　b は整数でないなら，異常終了します。

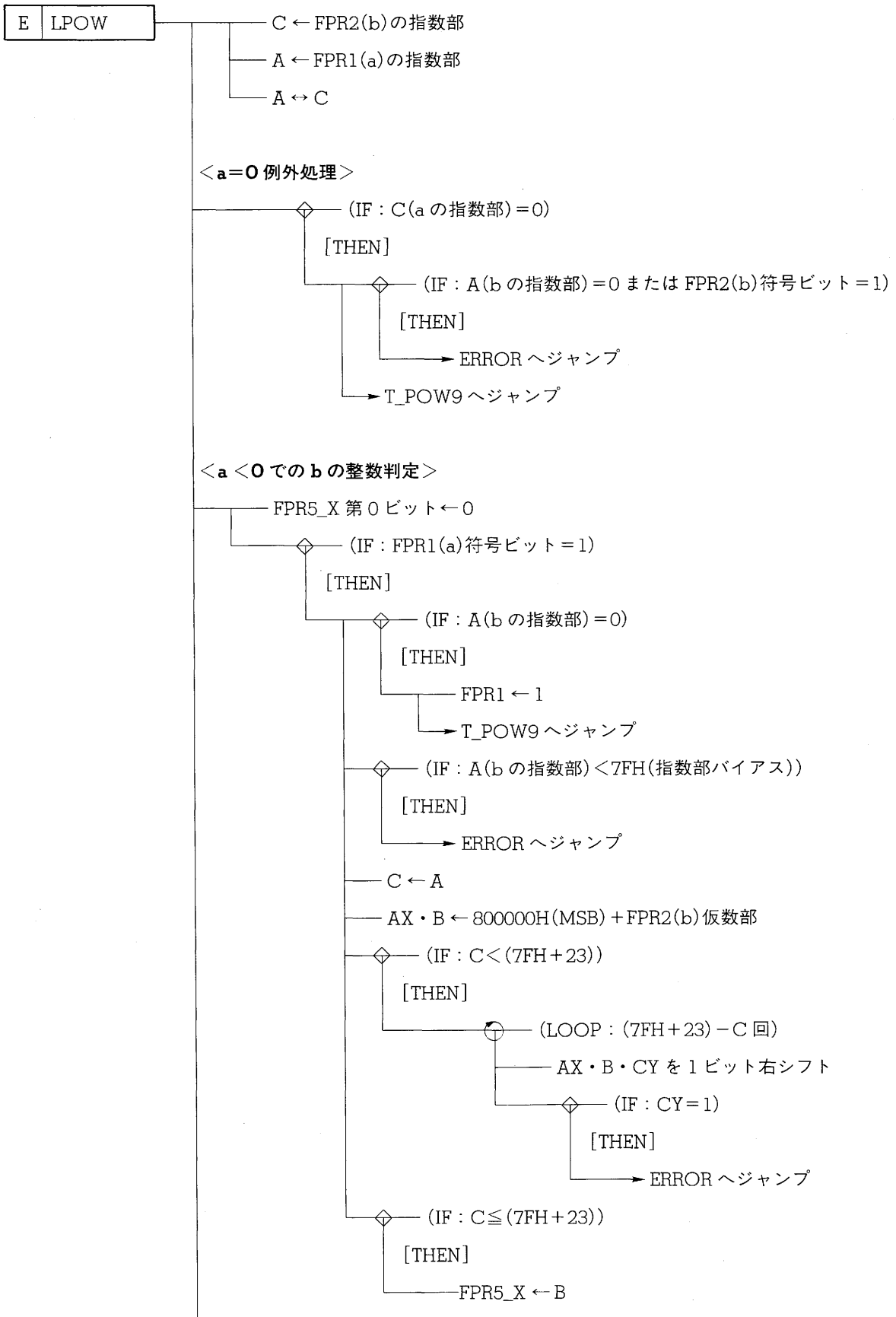
(e) $e^{b\log(|a|)}$ を LLOG，LEXP 関数により求めます。

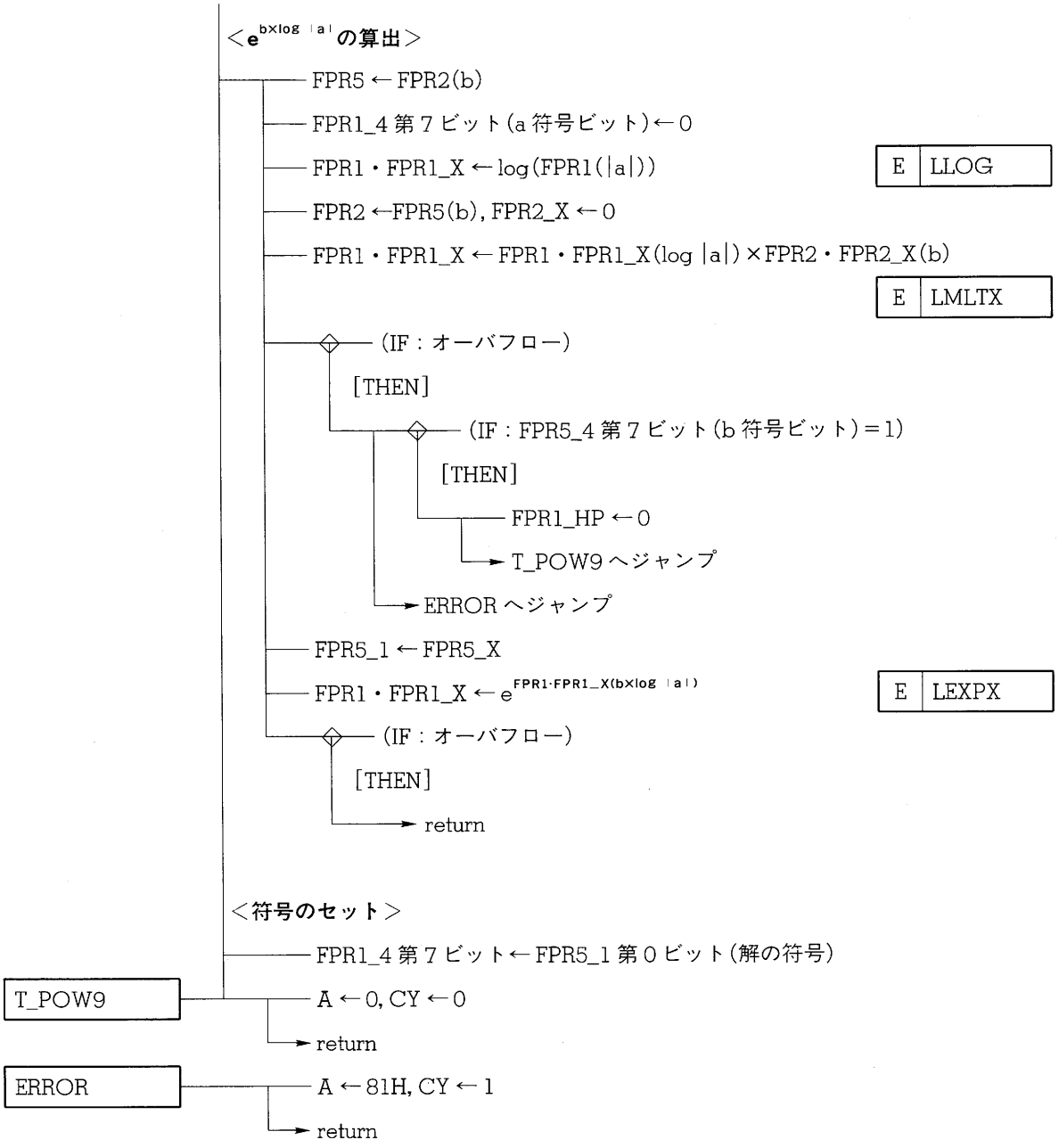
(f) a＜0，かつ b が奇整数なら，(e) で求めた解の符号を反転します。

(9) 浮動小数点定数データ

定数データ 1 を使用しています。

(10) 処 理 図


E
LPOW
C ← FPR2(b)の指数部
A ← FPR1(a)の指数部
A ↔ C
＜a＝0 例外処理＞
(IF：C(a の指数部)＝0)
[THEN]
(IF：A(b の指数部)＝0 または FPR2(b)符号ビット＝1)
[THEN]
ERROR へジャンプ
T_POW9 へジャンプ
＜a ＜0 での b の整数判定＞
FPR5_X 第 0 ビット ← 0
(IF：FPR1(a)符号ビット＝1)
[THEN]
(IF：A(b の指数部)＝0)
[THEN]
FPR1 ← 1
T_POW9 へジャンプ
(IF：A(b の指数部)＜7FH(指数部バイアス))
[THEN]
ERROR へジャンプ
C ← A
AX・B ← 800000H(MSB)＋FPR2(b)仮数部
(IF：C＜(7FH＋23))
[THEN]
(LOOP：(7FH＋23)－C 回)
AX・B・CY を 1 ビット右シフト
(IF：CY＝1)
[THEN]
ERROR へジャンプ
(IF：C≦(7FH＋23))
[THEN]
FPR5_X ← B


＜$e^{b \times \log |a|}$の算出＞
FPR5 ← FPR2(b)
FPR1_4 第 7 ビット(a 符号ビット)← 0
FPR1・FPR1_X ← log(FPR1(|a|))
E
LLOG
FPR2 ←FPR5(b), FPR2_X ← 0
FPR1・FPR1_X ← FPR1・FPR1_X(log |a|) × FPR2・FPR2_X(b)
E
LMLTX
(IF：オーバフロー)
[THEN]
(IF：FPR5_4 第 7 ビット(b 符号ビット)＝1)
[THEN]
FPR1_HP ← 0
T_POW9 へジャンプ
ERROR へジャンプ
FPR5_1 ← FPR5_X
FPR1・FPR1_X ← $e^{FPR1 \cdot FPR1\_X(b \times \log |a|)}$
E
LEXPX
(IF：オーバフロー)
[THEN]
return
＜符号のセット＞
FPR1_4 第 7 ビット ← FPR5_1 第 0 ビット(解の符号)
T_POW9
A ← 0, CY ← 0
return
ERROR
A ← 81H, CY ← 1
return

4.10　平方根関数（LSQRT）

(1) 処理内容

FPR1 の値を a として，$\sqrt{a}$ を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LLD, LSQRT

(3) 消費スタック・サイズ

4（LSQRT からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：1993 μs

最大：2076 μs（$\sqrt{(5.1001101e-35)}$）

(7) アルゴリズム

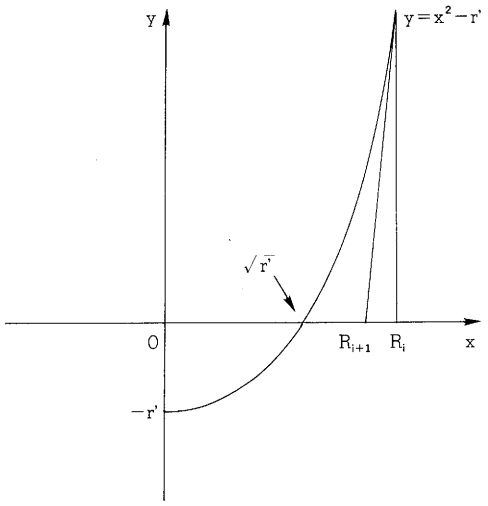
次の式より，$\sqrt{a}$ の指数部 n を得ます。

$$\sqrt{a}=\sqrt{(r\times 2^{2n})}=\sqrt{r}\times 2^{n} \qquad (1\leqq r<4)$$

$\sqrt{r}$ の計算には，Newton-Raphson 法を用います。

次ページの図に示すように，$\sqrt{r'}$ は 2 次関数 $y=x^2-r'$ と x 軸との交点の x 座標です。

R_i を $\sqrt{r'}$ の近似値として，点 (R_i, R_i^2-r') から $y=x^2-r'$ の接線を引きます。接線と x 軸との交点の x 座標を R_{i+1} とすると，R_{i+1} は R_i よりさらに近い $\sqrt{r'}$ の近似値となります。



R_{i+1} は R_i により，次の式で求めます。

$$R_{i+1}=\frac{R_i}{2}+\frac{r'}{2R_i}$$

この関数では，1≦r<2 なら r'=r，2≦r<4 なら r'=r/4，初期近似値 R1=1 として第5近似値 R5 を求め，$\sqrt{a}$ の仮数部を得ます。

(8) 処理手順

(a) a=0 の場合，0 を解として演算を終了します。

(b) a<0 の場合，異常終了します。

(c) a の指数部 ae（7FH のバイアスを含む）と仮数部 af に対して，n'=n+7FH，r'/2 を次式により得ます。

- ae は奇数：n'=(ae−7FH)/2+7FH=(ae−1)/2+40H，r'/2=af/2
- ae は偶数：n'=(ae−7FH−1)/2+7FH=ae/2+3FH，r'/2=((af×2)/4)/2

(d) $\sqrt{r'}$ の第2次近似値 R2=1/2+r'/2 を計算します。

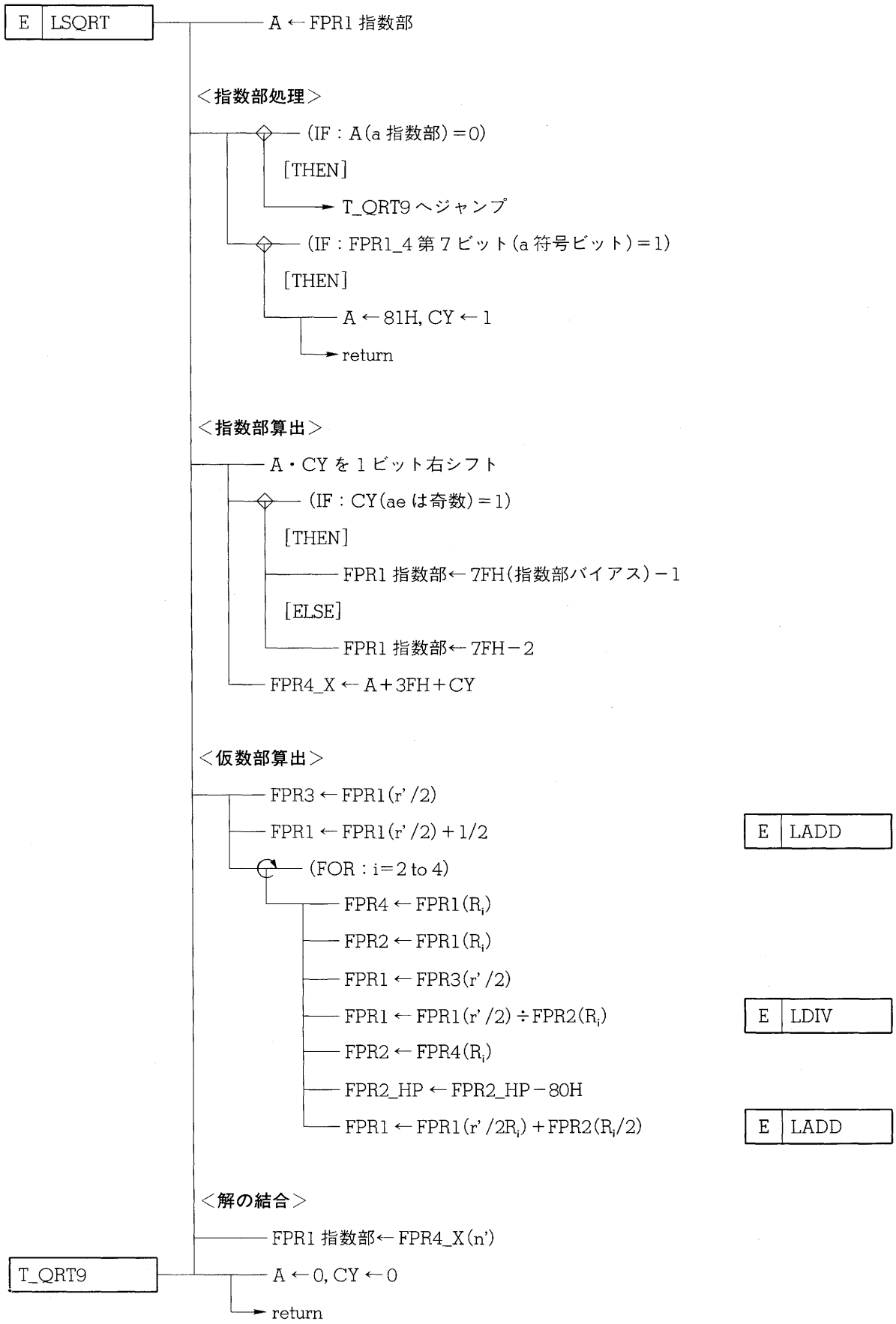
(e) 近似式により第3，第4，第5近似値を計算します。

(f) 第5近似値の指数部を n' に置き換え，解とします。

(9) 浮動小数点定数データ

定数データ 1/2 を使用しています。

(10) 処 理 図


E LSQRT
A ← FPR1 指数部
＜指数部処理＞
(IF : A(a 指数部) = 0)
[THEN]
T_QRT9 へジャンプ
(IF : FPR1_4 第 7 ビット(a 符号ビット) = 1)
[THEN]
A ← 81H, CY ← 1
return
＜指数部算出＞
A・CY を 1 ビット右シフト
(IF : CY(ae は奇数) = 1)
[THEN]
FPR1 指数部← 7FH(指数部バイアス) − 1
[ELSE]
FPR1 指数部← 7FH − 2
FPR4_X ← A + 3FH + CY
＜仮数部算出＞
FPR3 ← FPR1(r' /2)
FPR1 ← FPR1(r' /2) + 1/2
E LADD
(FOR : i = 2 to 4)
FPR4 ← FPR1(Ri)
FPR2 ← FPR1(Ri)
FPR1 ← FPR3(r' /2)
FPR1 ← FPR1(r' /2) ÷ FPR2(Ri)
E LDIV
FPR2 ← FPR4(Ri)
FPR2_HP ← FPR2_HP − 80H
FPR1 ← FPR1(r' /2Ri) + FPR2(Ri/2)
E LADD
＜解の結合＞
FPR1 指数部← FPR4_X(n')
T_QRT9
A ← 0, CY ← 0
return

4.11　arcsin 関数（LASIN）

(1) 処理内容

FPR1 の値を x として，arcsin(x) を FPR1 に返します。

- ●入力値 x の有効範囲：－1～1
- ●返値の範囲　　：$-\pi/2 \sim \pi/2$
- ●単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LSQRT, LASIN, LATAN, LRCPN

(3) 消費スタック・サイズ

6（LASIN からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR5, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：5265 μs

最大：6675 μs（arcsin(0.98437494)）

(7) アルゴリズム

次の式により，逆正接関数に変換して解を求めます。

$$\arcsin(x) = \arctan\left(x/\sqrt{(1-x^2)}\right)$$

(8) 処理手順

(a) x＝1 なら $\pi/2$ を，x＝－1 なら $-\pi/2$ を解として演算を終了します。

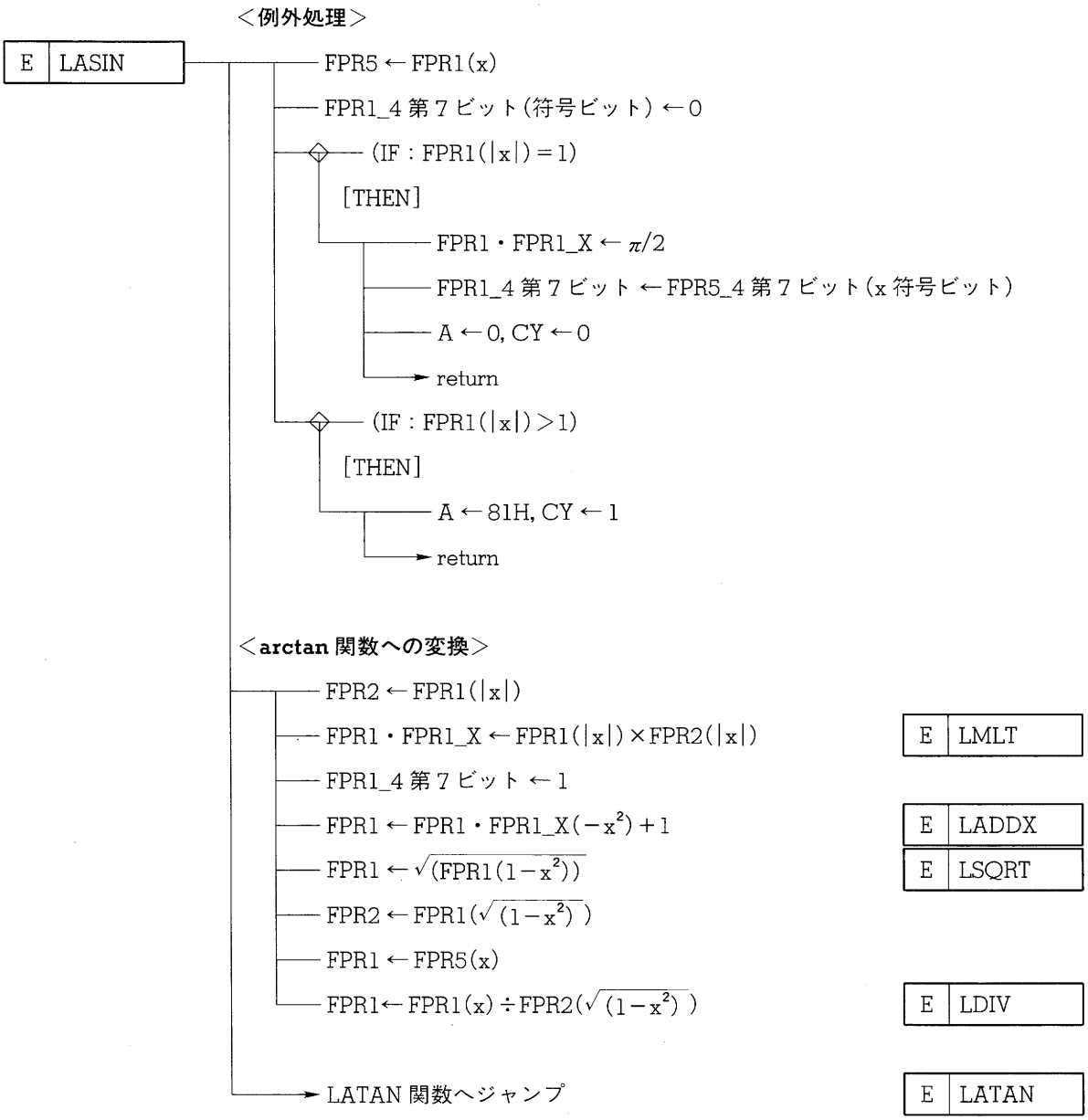
(b) |x|＞1 なら異常終了します。

(c) $x/\sqrt{(1-x^2)}$ を求め，LATAN 関数にジャンプします。

(9) 浮動小数点定数データ

拡張仮数部を持つ定数データ 1，および $\pi/2$ を使用しています。

(10) 処 理 図


＜例外処理＞
E LASIN
FPR5 ← FPR1(x)
FPR1_4 第7ビット(符号ビット) ← 0
(IF : FPR1(|x|) = 1)
[THEN]
FPR1・FPR1_X ← π/2
FPR1_4 第7ビット ← FPR5_4 第7ビット(x 符号ビット)
A ← 0, CY ← 0
return
(IF : FPR1(|x|) > 1)
[THEN]
A ← 81H, CY ← 1
return
＜arctan 関数への変換＞
FPR2 ← FPR1(|x|)
FPR1・FPR1_X ← FPR1(|x|) × FPR2(|x|)
E LMLT
FPR1_4 第7ビット ← 1
FPR1 ← FPR1・FPR1_X(−x²) + 1
E LADDX
FPR1 ← √(FPR1(1−x²))
E LSQRT
FPR2 ← FPR1(√(1−x²))
FPR1 ← FPR5(x)
FPR1 ← FPR1(x) ÷ FPR2(√(1−x²))
E LDIV
LATAN 関数へジャンプ
E LATAN

4.12　arccos 関数（LACOS）

(1) 処理内容

FPR1 の値を x として，arccos(x) を FPR1 に返します。

●入力値 x の有効範囲：－1～＋1

●返値の範囲　　：0～π

●単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LSQRT, LASIN, LACOS, LATAN, LRCPN

(3) 消費スタック・サイズ

8（LACOS からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR5, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：5230 μs

最大：6794 μs（arccos(0.98437494)）

(7) アルゴリズム

次の式により求めます。

$$\arccos(x) = \pi/2 - \arcsin(x)$$

(8) 処理手順

(a) LASIN 関数により arcsin(x) を求めます。

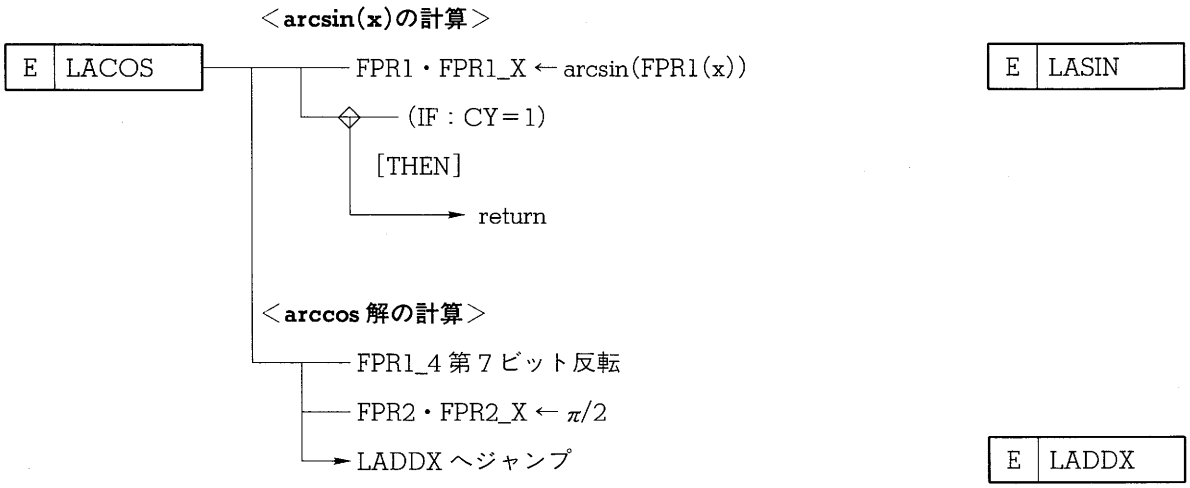
(b) LASIN 関数が異常終了した場合，そのまま異常終了します。

(c) $\pi/2$－arcsin(x) を解とします。

(9) 浮動小数点定数データ

拡張仮数部を持つ定数データ $\pi/2$ を使用しています。

(10) 処 理 図


＜arcsin(x)の計算＞
E
LACOS
FPR1・FPR1_X ← arcsin(FPR1(x))
(IF：CY＝1)
[THEN]
return
E
LASIN
＜arccos 解の計算＞
FPR1_4 第7ビット反転
FPR2・FPR2_X ← π/2
LADDX へジャンプ
E
LADDX

4.13　arctan 関数（LATAN）

(1) 処理内容

FPR1 の値を x として arctan(x) を FPR1 に返します。

●返値の範囲：$-\pi/2 \sim +\pi/2$

●単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LATAN, LRCPN

(3) 消費スタック・サイズ

6（LATAN からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：2630 μs

最大：3839 μs（arctan(1.0000001)）

(7) アルゴリズム

島内剛一氏（立教大学）による最良近似式を使用します。

$$\arctan(x) \fallingdotseq \sum_{i=0}^{n} (a_i \times (4x)^{2i+1})$$

この関数では n＝3 とし，係数 a_0, a_1, a_2, a_3 を次のように定めています。

$a_0 =$　0.24999　99999　43

$a_1 = -0.00520$　83303　18

$a_2 =$　0.00019　52689　40

$a_3 = -0.00000$　84855　00

注意　島内剛一氏の最良近似式は$|x| \leqq 1/8$の範囲でしか使用できません。したがって，$|x| \geqq 1/8$の場合，次のようにして求めます。

- $|x| \geqq 1$の場合，$x' = 1/|x|$ とし

  $\arctan(|x|) = \pi/2 - \arctan(1/|x|)$

- さらに$x' \geqq 1/8$の場合 V，W を以下に定め

  $V = \dfrac{x' - W}{1 + x' \times W}$ ，W は 1/8，3/8，5/8，7/8 のうち最も x' に近い数値

  $\arctan(x') = \arctan(W) + \arctan(V)$

備考　TAN 関数の加法定理を利用しています。

(8) 処理手順

(a) x の符号ビットを退避し，x の絶対値をとります。

(b) $|x| \geqq 1$の場合，$1/|x|$ を求めます。

(c) さらに次の表により W を求め，V を計算します。

x' の範囲	W
1/8 未満	0
1/8 以上，1/4 未満	1/8
1/4 以上，2/4 未満	3/8
2/4 以上，3/4 未満	5/8
3/4 以上	7/8

(d) arctan(V) の近似多項式第 1 項（$4a_0V$）を arctan(W) ＋$4a_0V$ に置き換え，近似多項式を計算します。

(e) $|x| \geqq 1$の場合，$\pi/2$－(近似式解) を求め arctan($|x|$) 解とします。

(f) x の元の符号を arctan($|x|$) 解に埋めこみます。

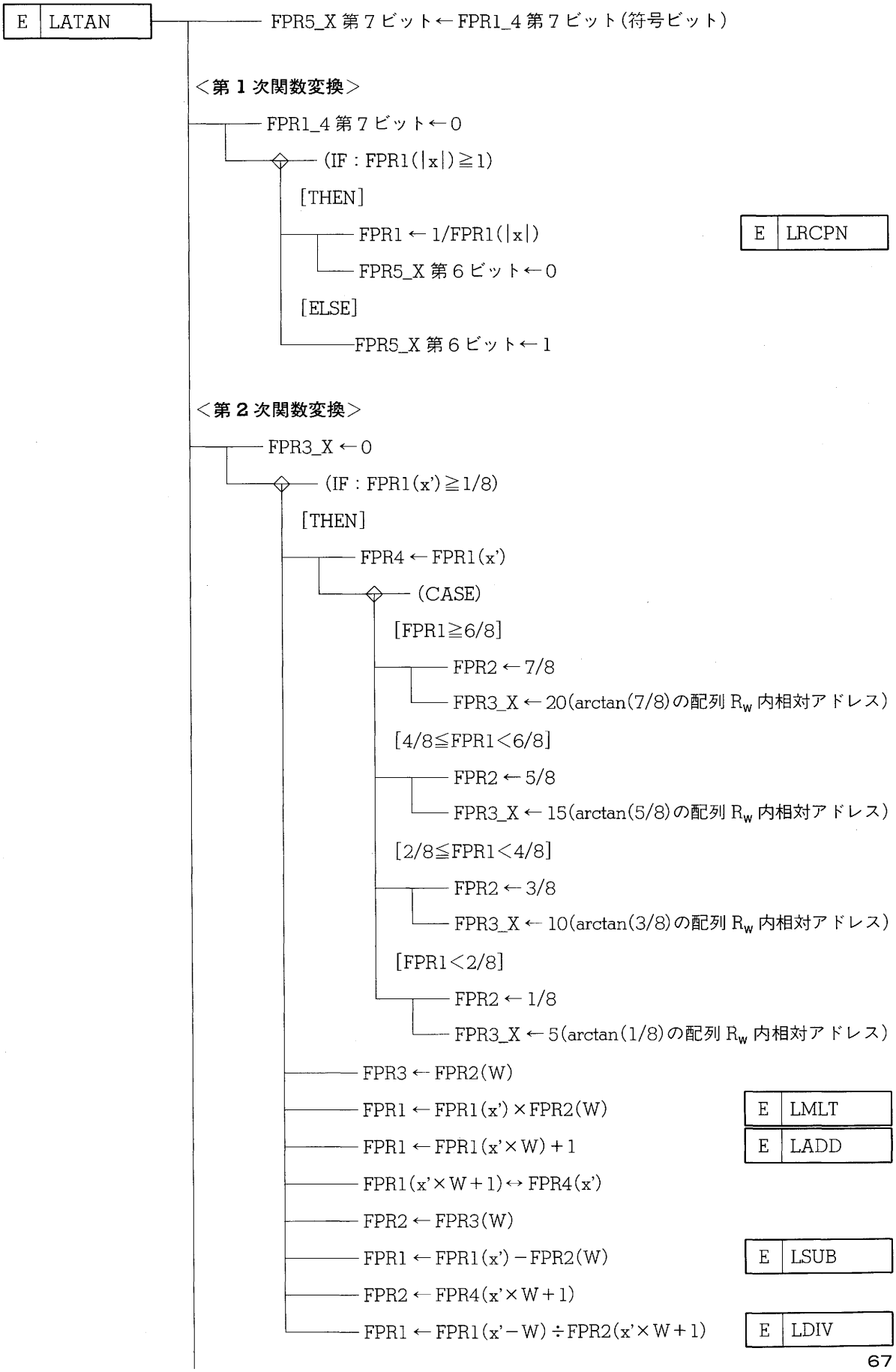
(9) 浮動小数点定数データ

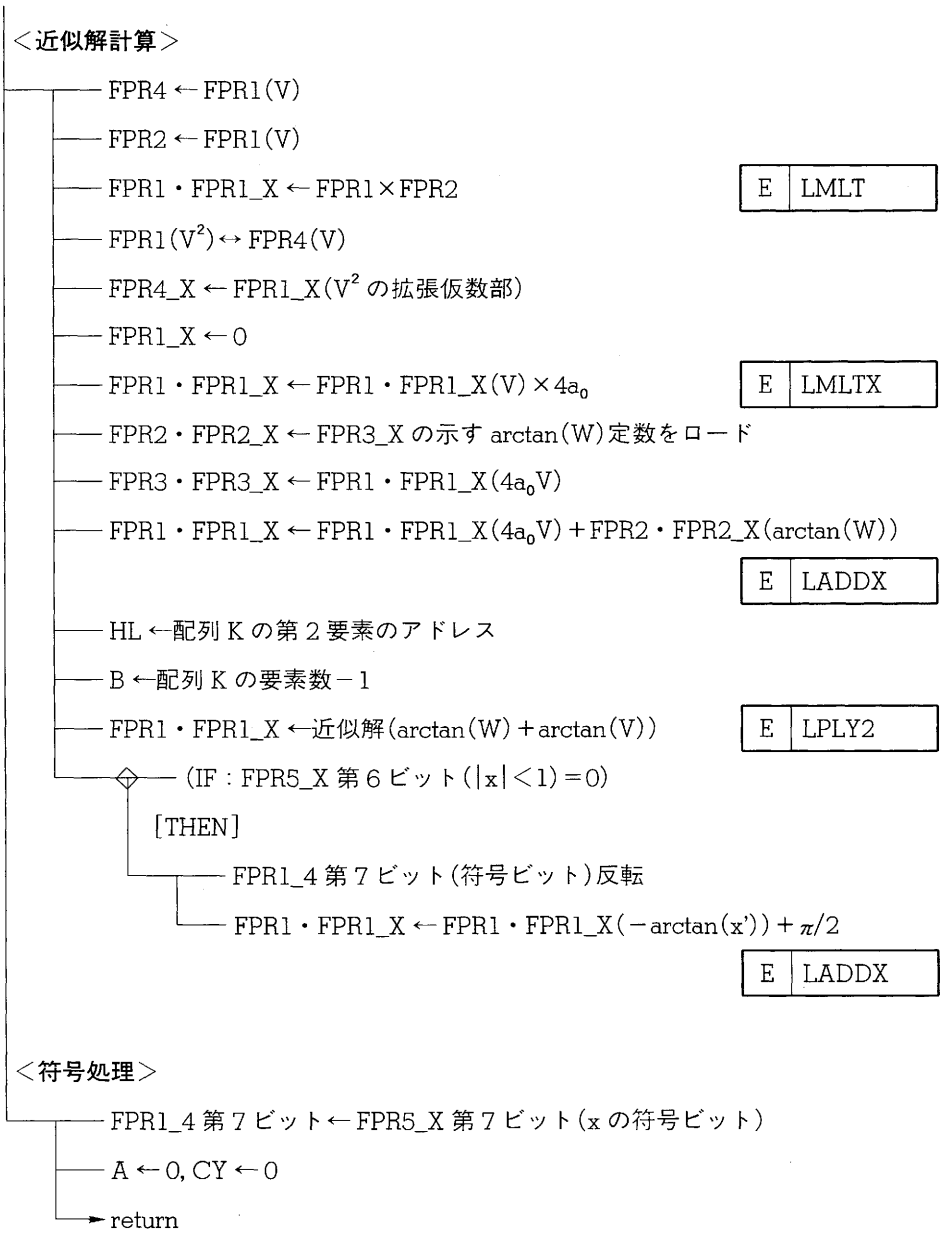
(a) 拡張仮数部を持つ定数データ 1，および $\pi/2$ を使用しています。

(b) 拡張仮数部を持つ定数データ arctan(0), arctan(1/8), arctan(3/8), arctan(5/8), arctan(7/8) を要素数 5 の配列 R_W として使用しています。

(c) 近似多項式の係数列に対して，拡張仮数部を持つ定数データ $4a_0$，$16a_1/a_0$，$16a_2/a_1$，$16a_3/a_2$ を要素数 4 の配列 K として用いています。

(10) 処 理 図


E LATAN
FPR5_X 第7ビット←FPR1_4 第7ビット(符号ビット)
＜第1次関数変換＞
FPR1_4 第7ビット←0
(IF : FPR1(|x|)≧1)
[THEN]
FPR1 ← 1/FPR1(|x|)
E LRCPN
FPR5_X 第6ビット←0
[ELSE]
FPR5_X 第6ビット←1
＜第2次関数変換＞
FPR3_X ← 0
(IF : FPR1(x')≧1/8)
[THEN]
FPR4 ← FPR1(x')
(CASE)
[FPR1≧6/8]
FPR2 ← 7/8
FPR3_X ← 20(arctan(7/8)の配列 Rw 内相対アドレス)
[4/8≦FPR1＜6/8]
FPR2 ← 5/8
FPR3_X ← 15(arctan(5/8)の配列 Rw 内相対アドレス)
[2/8≦FPR1＜4/8]
FPR2 ← 3/8
FPR3_X ← 10(arctan(3/8)の配列 Rw 内相対アドレス)
[FPR1＜2/8]
FPR2 ← 1/8
FPR3_X ← 5(arctan(1/8)の配列 Rw 内相対アドレス)
FPR3 ← FPR2(W)
FPR1 ← FPR1(x')×FPR2(W)
E LMLT
FPR1 ← FPR1(x'×W)+1
E LADD
FPR1(x'×W+1)↔FPR4(x')
FPR2 ← FPR3(W)
FPR1 ← FPR1(x')−FPR2(W)
E LSUB
FPR2 ← FPR4(x'×W+1)
FPR1 ← FPR1(x'−W)÷FPR2(x'×W+1)
E LDIV


＜近似解計算＞
FPR4 ← FPR1(V)
FPR2 ← FPR1(V)
FPR1・FPR1_X ← FPR1×FPR2
E LMLT
FPR1(V^2) ↔ FPR4(V)
FPR4_X ← FPR1_X(V^2 の拡張仮数部)
FPR1_X ← 0
FPR1・FPR1_X ← FPR1・FPR1_X(V)×$4a_0$
E LMLTX
FPR2・FPR2_X ← FPR3_X の示す arctan(W)定数をロード
FPR3・FPR3_X ← FPR1・FPR1_X($4a_0$V)
FPR1・FPR1_X ← FPR1・FPR1_X($4a_0$V)＋FPR2・FPR2_X(arctan(W))
E LADDX
HL ←配列 K の第 2 要素のアドレス
B ←配列 K の要素数－1
FPR1・FPR1_X ←近似解(arctan(W)＋arctan(V))
E LPLY2
(IF : FPR5_X 第 6 ビット(|x|＜1)＝0)
[THEN]
FPR1_4 第 7 ビット(符号ビット)反転
FPR1・FPR1_X ← FPR1・FPR1_X(－arctan(x'))＋π/2
E LADDX
＜符号処理＞
FPR1_4 第 7 ビット← FPR5_X 第 7 ビット(x の符号ビット)
A ← 0, CY ← 0
return

4.14　sinh 関数（LHSIN）

(1) 処理内容

FPR1 の値を x として，sinh(x) を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, LHSIN, LRCPN, FTOL, LTOF

(3) 消費スタック・サイズ

8（LHSIN からの戻り番地 2 バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR5_1, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：3310 μs

最大：4784 μs（sinh(3.351413)）

(7) アルゴリズム

●$|x| \geqq 0.5$ の場合，次式により求めます。

$$\sinh(x) = (x \text{ の符号}) \frac{e^{|x|} - e^{-|x|}}{2}$$

●$|x| < 0.5$ の場合，Taylor 近似式により求めます。

$$\sinh(x) = x + \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \frac{1}{7!}x^7$$

(8) 処理手順

●$|x| \geqq 0.5$ の場合

(a) x の符号を記憶し，x の絶対値をとります。

(b) $e^{|x|}$ を LEXP 関数により求めます。

(c) $e^{|x|}$ がオーバフローの場合，異常終了します。

(d) $(e^{|x|}-e^{-|x|})/2$ を求め，x の元の符号を埋めこみ，解とします。

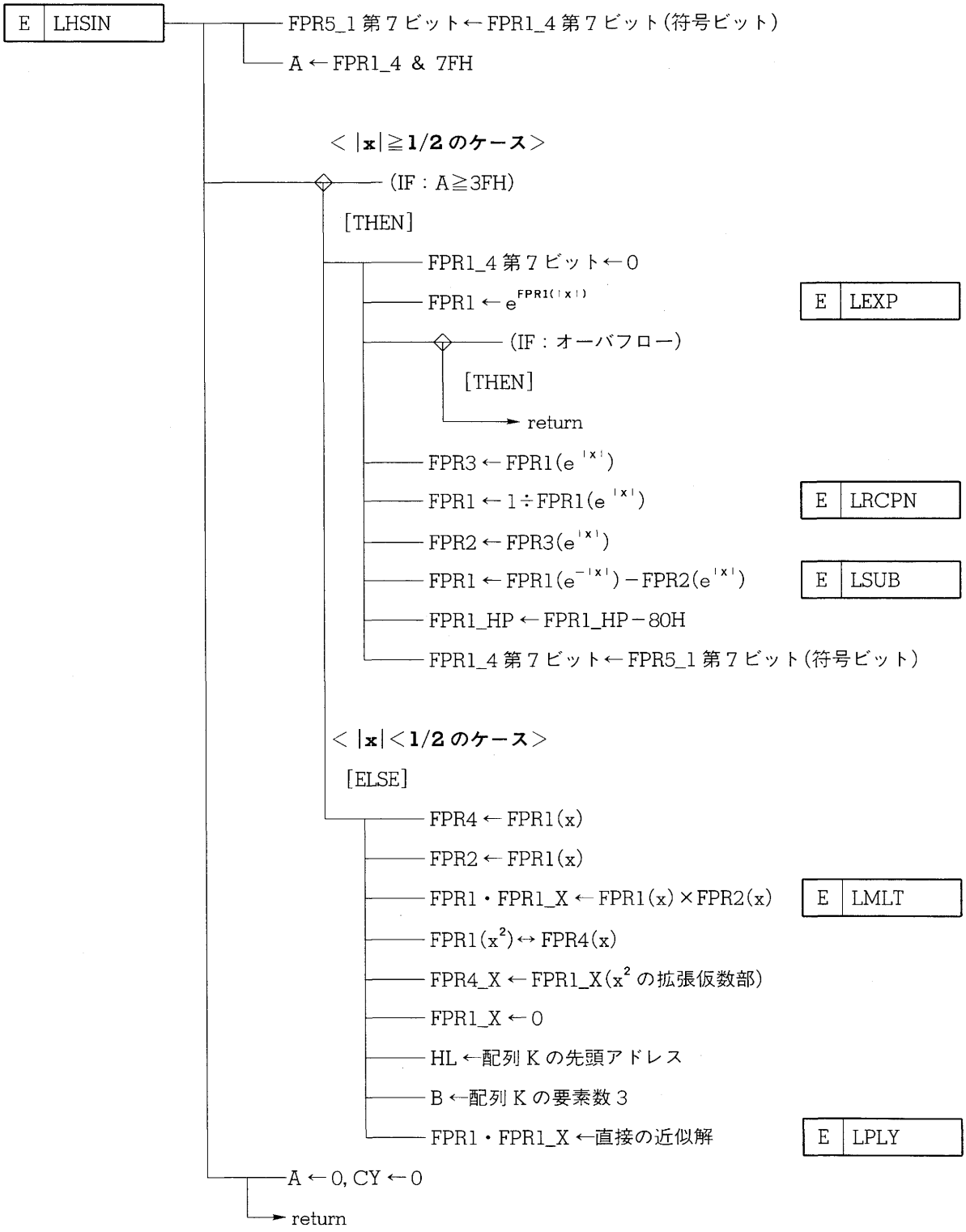
●$|x| < 0.5$ の場合

sinh(x) を Taylor 近似式により求めます。

(9) 浮動小数点定数データ

Taylor 近似式の係数列に対して，拡張仮数部を持つ定数データ1/3!，3!/5!，5!/7!を要素数3の配列 K として使用しています。

(10) 処 理 図


E | LHSIN
FPR5_1 第7ビット←FPR1_4 第7ビット(符号ビット)
A←FPR1_4 & 7FH
＜|x|≧1/2 のケース＞
(IF：A≧3FH)
[THEN]
FPR1_4 第7ビット←0
FPR1←e^FPR1(|x|)
E | LEXP
(IF：オーバフロー)
[THEN]
return
FPR3←FPR1(e^|x|)
FPR1←1÷FPR1(e^|x|)
E | LRCPN
FPR2←FPR3(e^|x|)
FPR1←FPR1(e^-|x|)−FPR2(e^|x|)
E | LSUB
FPR1_HP←FPR1_HP−80H
FPR1_4 第7ビット←FPR5_1 第7ビット(符号ビット)
＜|x|＜1/2 のケース＞
[ELSE]
FPR4←FPR1(x)
FPR2←FPR1(x)
FPR1・FPR1_X←FPR1(x)×FPR2(x)
E | LMLT
FPR1(x^2)↔FPR4(x)
FPR4_X←FPR1_X(x^2 の拡張仮数部)
FPR1_X←0
HL←配列 K の先頭アドレス
B←配列 K の要素数 3
FPR1・FPR1_X←直接の近似解
E | LPLY
A←0, CY←0
return

4.15　cosh 関数（LHCOS）

(1) 処理内容

FPR1 の値を x として，cosh(x) を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, LHCOS, LRCPN, FTOL, LTOF

(3) 消費スタック・サイズ

8（LHCOS からの戻り番地２バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：4139 μs

最大：4768 μs（cosh(4.0319099)）

(7) アルゴリズム

次式により求めます。

$$\cosh(x)=\frac{e^{|x|}+e^{-|x|}}{2}$$

(8) 処理手順

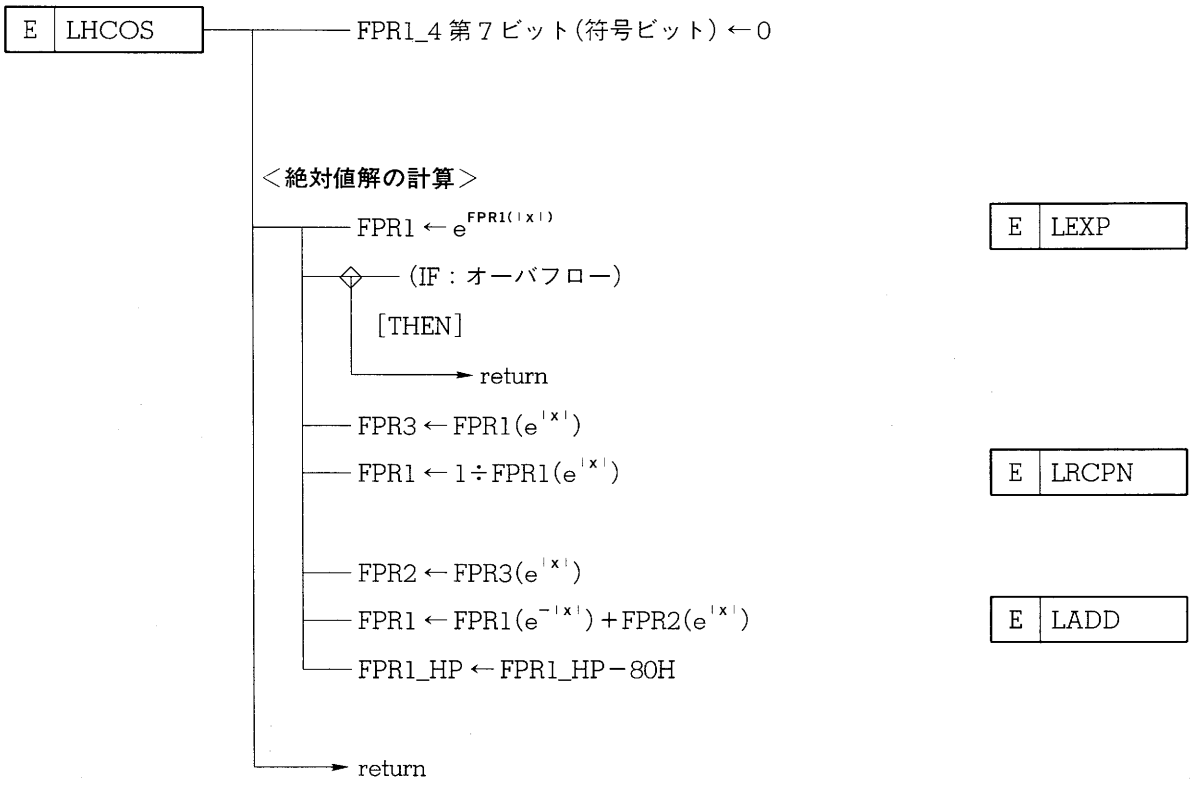
(a) x の絶対値をとります。

(b) $e^{|x|}$ を LEXP 関数により求めます。

(c) $e^{|x|}$ がオーバフローの場合，異常終了します。

(d) $(e^{|x|}+1/e^{|x|})/2$ を求め，解とします。

(9) 処 理 図


E LHCOS
FPR1_4 第7ビット(符号ビット) ←0
＜絶対値解の計算＞
FPR1 ← e^FPR1(|x|)
E LEXP
(IF：オーバフロー)
[THEN]
return
FPR3 ← FPR1(e^|x|)
FPR1 ← 1÷FPR1(e^|x|)
E LRCPN
FPR2 ← FPR3(e^|x|)
FPR1 ← FPR1(e^-|x|) + FPR2(e^|x|)
E LADD
FPR1_HP ← FPR1_HP − 80H
return

4.16　tanh 関数（LHTAN）

(1) 処理内容

FPR1 の値を x として，tanh(x) を FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, LHSIN, LHCOS, LHTAN, LRCPN, FTOL, LTOF

(3) 消費スタック・サイズ

12（LHTAN からの戻り番地２バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR5, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：7792 μs

最大：10021 μs（tanh(9.4484739)）

(7) アルゴリズム

次式により求めます。

$$\tanh(x)=\frac{\sinh(x)}{\cosh(x)}$$

(8) 処理手順

(a) cosh(x) を LHCOS 関数により求めます。

(b) オーバフローの場合，

- x＞0 なら，1 を解として演算を終了します。
- x＜0 なら，−1 を解として演算を終了します。

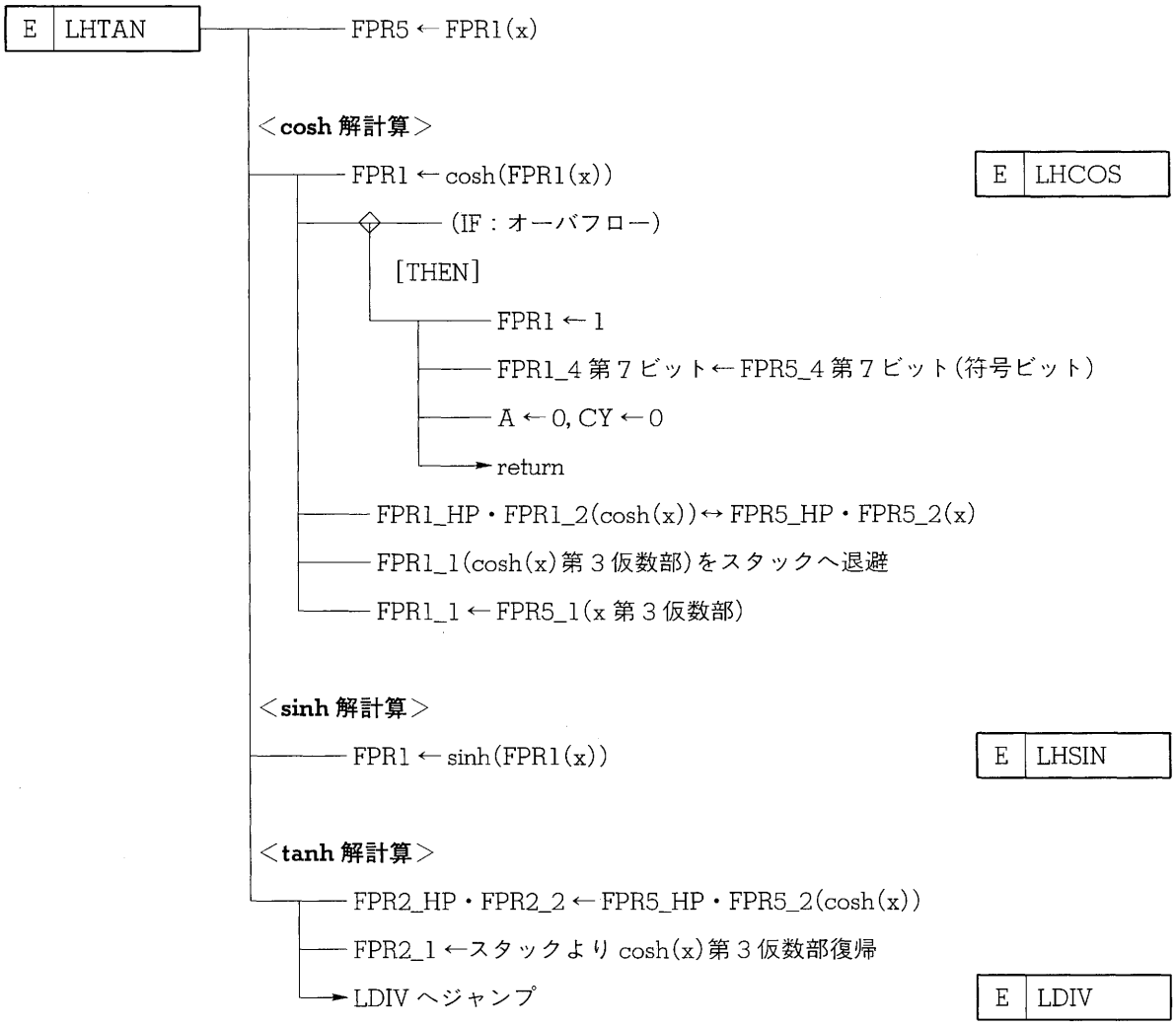
(c) sinh(x) を LHSIN 関数により求めます。

(d) sinh(x)/cosh(x) を求め解とします。

(9) 浮動小数点定数データ

定数データ 1 を使用しています。

(10) 処 理 図


E LHTAN
FPR5 ← FPR1(x)
＜cosh 解計算＞
FPR1 ← cosh(FPR1(x))
E LHCOS
(IF：オーバフロー)
[THEN]
FPR1 ← 1
FPR1_4 第7ビット ← FPR5_4 第7ビット(符号ビット)
A ← 0, CY ← 0
return
FPR1_HP・FPR1_2(cosh(x)) ↔ FPR5_HP・FPR5_2(x)
FPR1_1(cosh(x)第3仮数部)をスタックへ退避
FPR1_1 ← FPR5_1(x 第3仮数部)
＜sinh 解計算＞
FPR1 ← sinh(FPR1(x))
E LHSIN
＜tanh 解計算＞
FPR2_HP・FPR2_2 ← FPR5_HP・FPR5_2(cosh(x))
FPR2_1 ←スタックより cosh(x)第3仮数部復帰
LDIV へジャンプ
E LDIV

4.17　絶対値関数（LABS）

(1) 処理内容

FPR1 の値の絶対値をとり，FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LABS

(3) 消費スタック・サイズ

2（LABS からの戻り番地２バイトのみ）

(4) 使用レジスタ

A

(5) 使用ワーク・エリア

FPR1

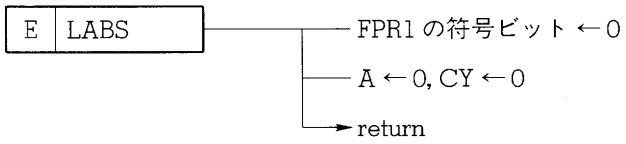
(6) 処理時間（内部システム・クロック＝8.38 MHz）

6.4 μs

(7) 処理手順

FPR1 の符号ビットを０にします。

(8) 処 理 図



4.18　逆数関数（LRCPN）

(1) 処理内容

FPR1 の値の逆数をとり，FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LLD, LRCPN

(3) 消費スタック・サイズ

4（LRCPN からの戻り番地２バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL

(5) 使用ワーク・エリア

FPR1, FPR2, FPR1_X, FPR2_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：539 μs

最大：637 μs（1/（8.5070602e+37））

(7) 処理手順

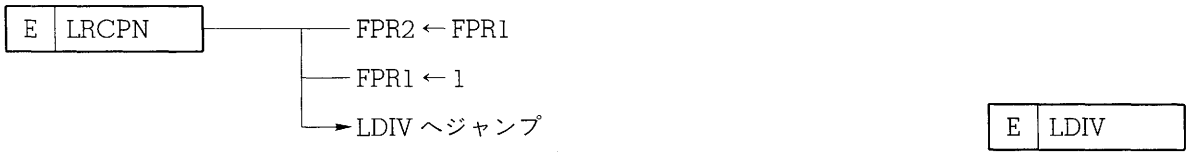
(a) FPR1 の値を FPR2 へ転送し，FPR1 に定数１をセットします。

(b) LDIV 関数へジャンプします。

(8) 浮動小数点定数データ

定数データ１を使用しています。

(9) 処 理 図



第5章　座標変換関数

座標変換関数には，次のものを用意しています。

(1) 極座標 → 直交座標への変換関数（POTORA）

極座標値（r, θ）を直交座標値（x, y）へ変換します。

値の受け渡しは，r, x を FPR1 により，θ, y を FPR2 により行います。

(2) 直交座標 → 極座標への変換関数（RATOPO）

直交座標値（x, y）を極座標値（r, θ）へ変換します。

値の受け渡しは，x, r を FPR1 により，y, θ を FPR2 により行います。

5.1　極座標 → 直交座標への変換関数（POTORA）

(1) 処理内容

FPR1 の値を r，FPR2 の値を θ として，極座標（r，θ）を直交座標（x，y）へ変換し，x を FPR1 に，y を FPR2 に返します。

●θの単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSIN，LCOS，POTORA，FTOL，LTOF

(3) 消費スタック・サイズ

14（POTORA からの戻り番地２バイトを含む）

(4) 使用レジスタ

AX，BC，DE，HL

(5) 使用ワーク・エリア

FPR1，FPR2，FPR3，FPR4，FPR5，FPR1_X，FPR2_X，FPR3_X，FPR4_X，FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：5336 μs

最大：10845 μs（r＝0.5，θ＝6.8056469e＋38）

(7) アルゴリズム

次の式により変換します。

$$x=r\times\cos(|\theta|),\quad y=r\times\sin(\theta)$$

(8) 処理手順

(a) r＝0 の場合，座標（0，0）を返します。

(b) r＜0 の場合，異常終了します。

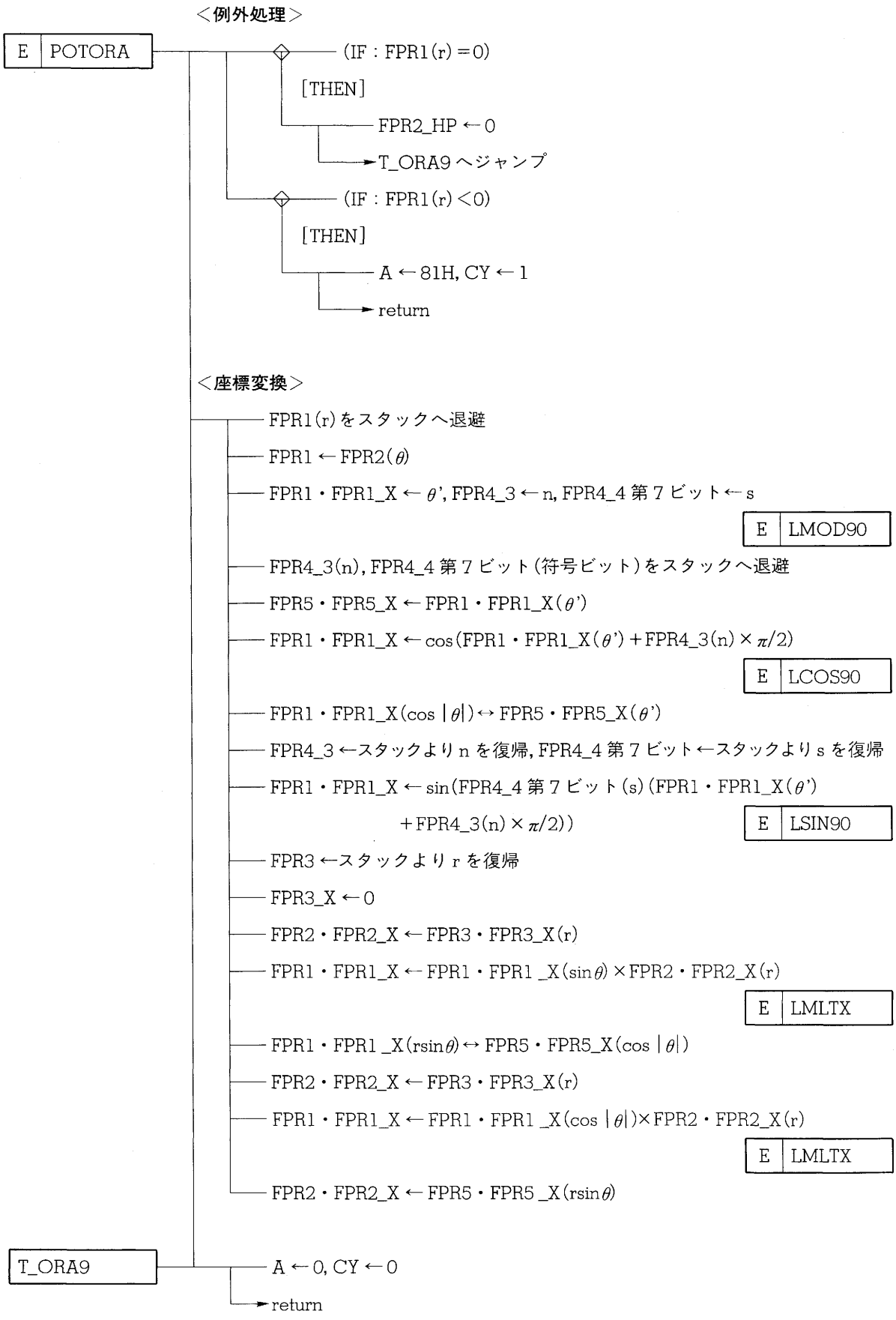
(c) θ を LMOD90 関数により，$\theta=s(\theta'+n\pi/2)$ に変換します。

ここで s は θ の符号，n は整数，$0\leqq\theta'<\pi/2$ を示します。

(d) sin（s（$\theta'+n\pi/2$））を LSIN90 関数により，cos（$\theta'+n\pi/2$）を LCOS90 関数により求めます。

(e) $r\times\cos(\theta'+n\pi/2)$ を x，$r\times\sin(s(\theta'+n\pi/2))$ を y とします。

(9) 処 理 図


＜例外処理＞
E POTORA
(IF : FPR1(r) = 0)
[THEN]
FPR2_HP ← 0
T_ORA9 へジャンプ
(IF : FPR1(r) ＜ 0)
[THEN]
A ← 81H, CY ← 1
return
＜座標変換＞
FPR1(r) をスタックへ退避
FPR1 ← FPR2(θ)
FPR1・FPR1_X ← θ', FPR4_3 ← n, FPR4_4 第7ビット ← s
E LMOD90
FPR4_3(n), FPR4_4 第7ビット(符号ビット)をスタックへ退避
FPR5・FPR5_X ← FPR1・FPR1_X(θ')
FPR1・FPR1_X ← cos(FPR1・FPR1_X(θ') + FPR4_3(n) × π/2)
E LCOS90
FPR1・FPR1_X(cos |θ|) ↔ FPR5・FPR5_X(θ')
FPR4_3 ← スタックより n を復帰, FPR4_4 第7ビット ← スタックより s を復帰
FPR1・FPR1_X ← sin(FPR4_4 第7ビット(s)(FPR1・FPR1_X(θ')
+ FPR4_3(n) × π/2))
E LSIN90
FPR3 ← スタックより r を復帰
FPR3_X ← 0
FPR2・FPR2_X ← FPR3・FPR3_X(r)
FPR1・FPR1_X ← FPR1・FPR1 _X(sinθ) × FPR2・FPR2_X(r)
E LMLTX
FPR1・FPR1 _X(rsinθ) ↔ FPR5・FPR5_X(cos |θ|)
FPR2・FPR2_X ← FPR3・FPR3_X(r)
FPR1・FPR1_X ← FPR1・FPR1 _X(cos |θ|) × FPR2・FPR2_X(r)
E LMLTX
FPR2・FPR2_X ← FPR5・FPR5 _X(rsinθ)
T_ORA9
A ← 0, CY ← 0
return

5.2　直交座標 → 極座標への変換関数（RATOPO）

(1) 処理内容

FPR1の値をx，FPR2の値をyとして，直交座標（x，y）を極座標（r，θ）へ変換し，rをFPR1に，θをFPR2に返します。

●返値θの範囲：$-\pi$～$+\pi$

●単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSQRT，LATAN，LRCPN，RATOPO

(3) 消費スタック・サイズ

10（RATOPOからの戻り番地2バイトを含む）

(4) 使用レジスタ

AX，BC，DE，HL

(5) 使用ワーク・エリア

FPR1，FPR2，FPR3，FPR4，FPR5，FPR1_X，FPR2_X，FPR3_X，FPR4_X，FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：5631 μs

最大：6979 μs（x＝－12.220954，y＝69.662003）

(7) アルゴリズム

次の2式により求めます。

$$r=\sqrt{(x^2+y^2)}$$

$$\theta=\arctan(y/x)$$

(8) 処理手順

(a) x＝y＝0の場合，そのままリターンします。

(b) x^2+y^2を求めます。

(c) x^2+y^2がオーバフローの場合，異常終了します。

(d) y/x を求めます。

(e) y/x がオーバフローの場合，次のようになります。

- y＞0 なら，$\theta=\pi/2$
- y＜0 なら，$\theta=-\pi/2$

(f) y/x が正常終了なら，arctan(y/x) を LATAN 関数により求め，θ とします。

(g) (f) で，x＜0 の場合，次のようになります。

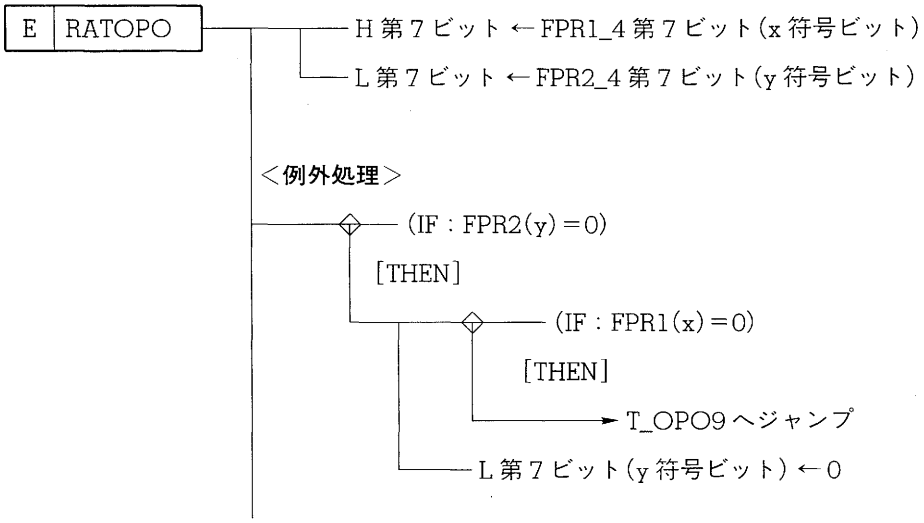
- y≧0 なら，θ に π を加算
- y＜0 なら，θ から π を減算

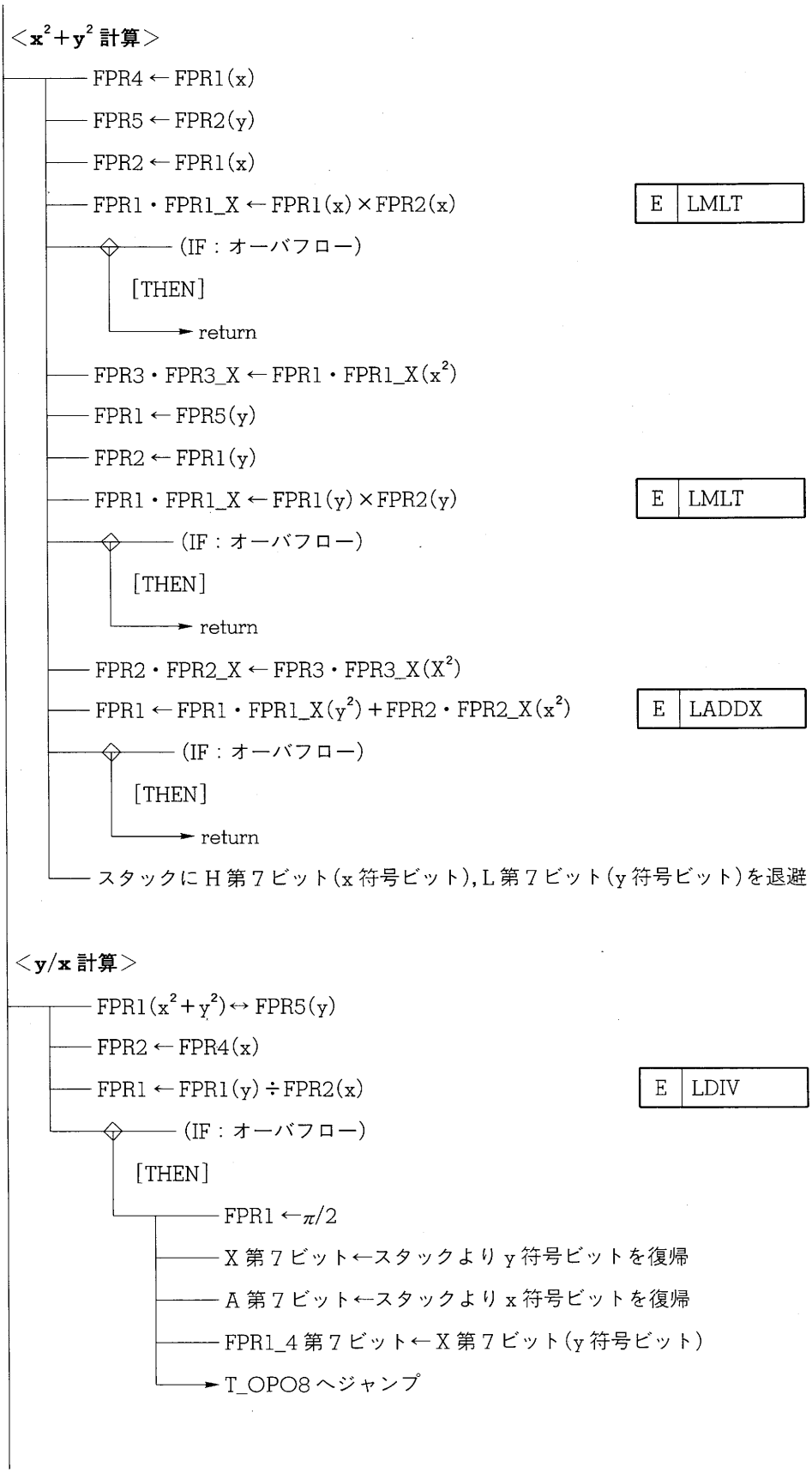
(h) $\sqrt{(x^2+y^2)}$ を求め，r とします。

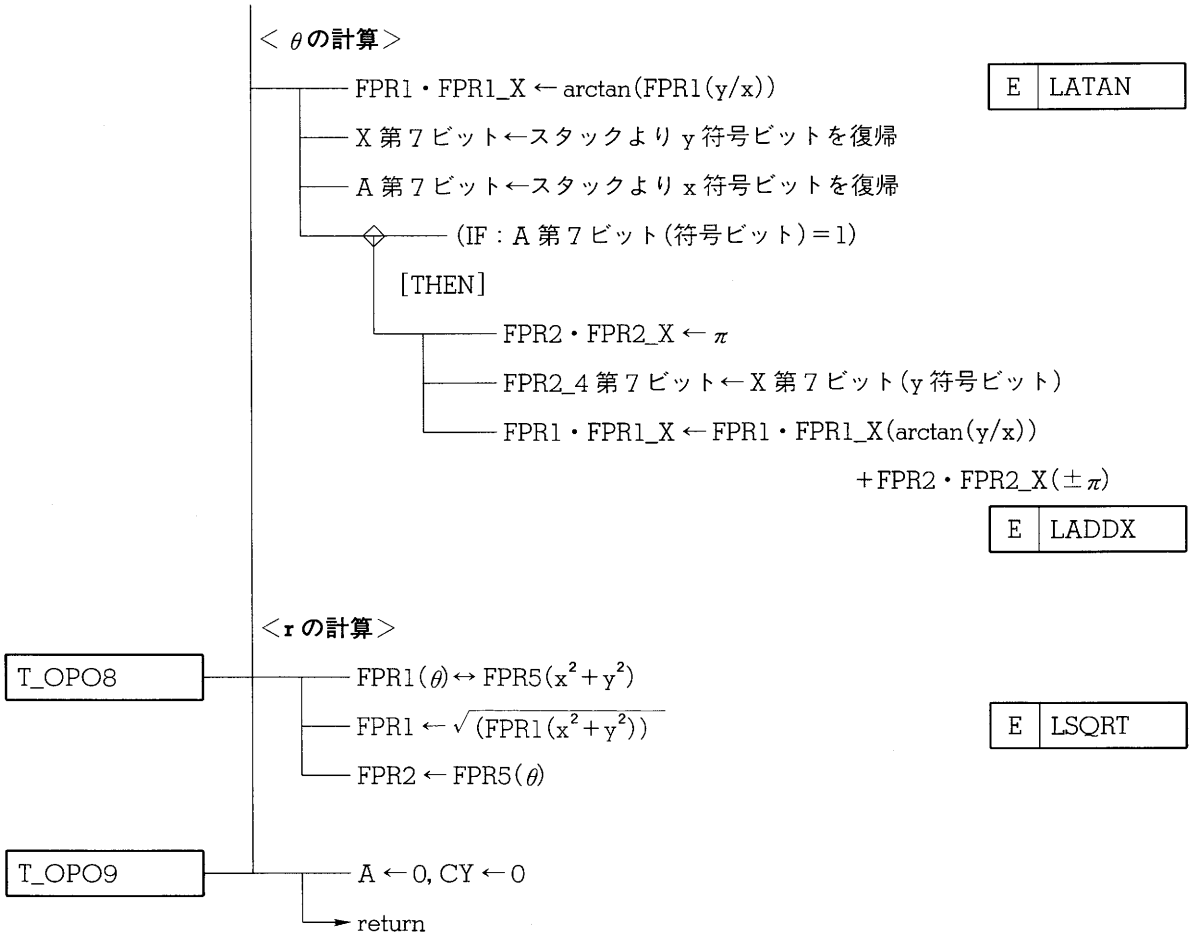
(9) 浮動小数点定数データ

$\pi/2$，および π（π は拡張形式）を定数データとして使用しています。

(10) 処 理 図




＜x^2+y^2 計算＞
FPR4 ← FPR1(x)
FPR5 ← FPR2(y)
FPR2 ← FPR1(x)
FPR1・FPR1_X ← FPR1(x)×FPR2(x)
E LMLT
(IF：オーバフロー)
[THEN]
return
FPR3・FPR3_X ← FPR1・FPR1_X(x^2)
FPR1 ← FPR5(y)
FPR2 ← FPR1(y)
FPR1・FPR1_X ← FPR1(y)×FPR2(y)
E LMLT
(IF：オーバフロー)
[THEN]
return
FPR2・FPR2_X ← FPR3・FPR3_X(X^2)
FPR1 ← FPR1・FPR1_X(y^2)＋FPR2・FPR2_X(x^2)
E LADDX
(IF：オーバフロー)
[THEN]
return
スタックにH第7ビット(x符号ビット),L第7ビット(y符号ビット)を退避
＜y/x 計算＞
FPR1(x^2+y^2)↔ FPR5(y)
FPR2 ← FPR4(x)
FPR1 ← FPR1(y)÷FPR2(x)
E LDIV
(IF：オーバフロー)
[THEN]
FPR1 ←π/2
X第7ビット←スタックよりy符号ビットを復帰
A第7ビット←スタックよりx符号ビットを復帰
FPR1_4第7ビット←X第7ビット(y符号ビット)
T_OPO8へジャンプ


＜θの計算＞
FPR1・FPR1_X ← arctan(FPR1(y/x))
E LATAN
X第7ビット←スタックよりy符号ビットを復帰
A第7ビット←スタックよりx符号ビットを復帰
(IF：A第7ビット(符号ビット)＝1)
[THEN]
FPR2・FPR2_X ← π
FPR2_4第7ビット←X第7ビット(y符号ビット)
FPR1・FPR1_X ← FPR1・FPR1_X(arctan(y/x))
＋FPR2・FPR2_X(±π)
E LADDX
＜r の計算＞
T_OPO8
FPR1(θ) ↔ FPR5(x²＋y²)
FPR1 ← √(FPR1(x²＋y²))
E LSQRT
FPR2 ← FPR5(θ)
T_OPO9
A ← 0, CY ← 0
return

第６章　型変換関数

型変換関数には，次のものを用意しています。

（1）文字列 → 浮動小数点形式への変換関数（ATOL）

先頭アドレスがHLレジスタで示される文字列を浮動小数点形式に変換し，FPR1に格納します。

（2）浮動小数点形式 → 文字列への変換関数（LTOA）

FPR1の値を文字列に変換し，HLレジスタが示すアドレスより格納します。

（3）２バイト整数型 → 浮動小数点形式への変換関数（FTOL）

DEレジスタ内容を符号付２バイト整数型として，浮動小数点形式に変換し，FPR1に格納します。

（4）浮動小数点形式 → ２バイト整数型への変換関数（LTOF）

FPR1の値を符号付２バイト整数型へ変換し，DEレジスタに格納します。

６.１　文字列 → 浮動小数点形式への変換関数（ATOL）

(1) 処理内容

先頭アドレスがHLレジスタで示される文字列を浮動小数点形式に変換し，FPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, ATOL, FTOL, LTOF

(3) 消費スタック・サイズ

14（ATOLからの戻り番地２バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL（HLレジスタの内容は保存されます）

(5) 使用ワーク・エリア

FPR1, FPR2, FPR3, FPR4, FPR5, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：5049 μs

最大：6388 μs（"0.00000000000000000117549428"）

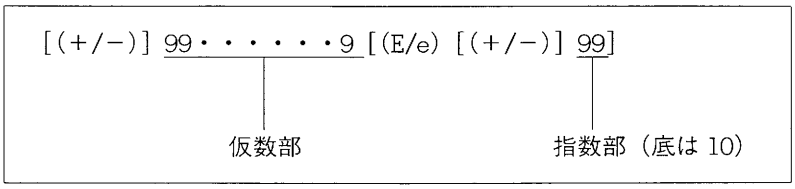
(7) 文字列の構成因子

文字列は次の17種の文字によって構成されます。

文字	アスキー・コード
0-9	30H-39H
＋	2BH
－	2DH
.	2EH
E	45H
e	65H
△（スペース）	20H
NUL	00H

(8) 文字列形式

文字列の形式は，次のようになります。



備考　[　]：省略可

9：0-9

(/)：どちらかを選択

例　“−99.8”　　＝−99.8

“.007e0”　　＝.007

“0998e−03”　＝998×10^{-3}

(9) 文字列の規定

次の規定外の文字列は，この関数ではエラーとなります。

(a) 文字列の終わりは △（スペース）または NUL で判断する。

(b) 文字列構成因子以外の文字が文字列中に含まれてはならない。

(c) 仮数部文字列の最大長は 27 で，1 つ以下の '.' を含んでよい。
ただし，1 つ以上の数字が含まれていなければならない。

(d) 指数部文字列の長さは 2 または 1 でなければならない。

(e) 値が 2^{129} 以上または -2^{129} 以下となった場合はエラー。

備考　仮数部値＝0 になる場合は，例外的に処理されます。
この場合，指数部文字列は無視され，0 が解として返されます。

(10) 処理手順

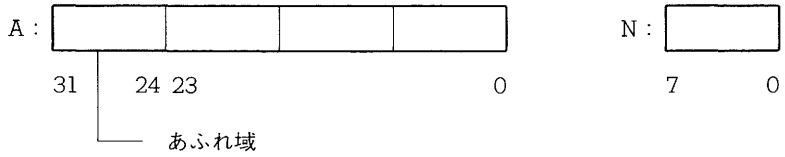
(a) 先頭文字が '−' なら符号を負，他なら正として記憶します。

(b) 仮数部文字列から小数部の桁数を求め，F とします。

(c) 小数点を除去した仮数部の文字列 “A_1，A_2，……，A_n” より，仮数部値 A を 4 バイト整数型で求めます。

$$A=(((A_1\cdot10+A_2)\cdot10+A_3)\cdot10\cdot\cdot\cdot A_{n-1})\cdot10+A_n$$

次の方法で計算します。



(i) 初期値を A=0, N=0 とします。

(ii) A_1 を A に加算します。

A_1 が存在しなければ，異常終了します。

(iii) あふれ域=0 の場合

A に 10 を掛け，A_k を加算します。

(iv) あふれ域≠0 の場合

N に 1 を加算し，A_k を無視します。

(v) (iii) ～ (iv) を k=2～n まで繰り返します。

(vi) n>27 なら異常終了します。

(d) (c) で求めた仮数部値 A を浮動小数点形式に正規化し，記憶していた符号ビットを埋め込み A' とします。

(e) 指数部文字列 "(+/−) B_1B_2" より，指数部値 B を求めます。

(f) 指数部値 B に，小数部の桁数と無視した仮数部の桁数を加えた実質指数部 B' (=B−F+N) を求めます。

(g) A', B' により，解を次の式で計算します。

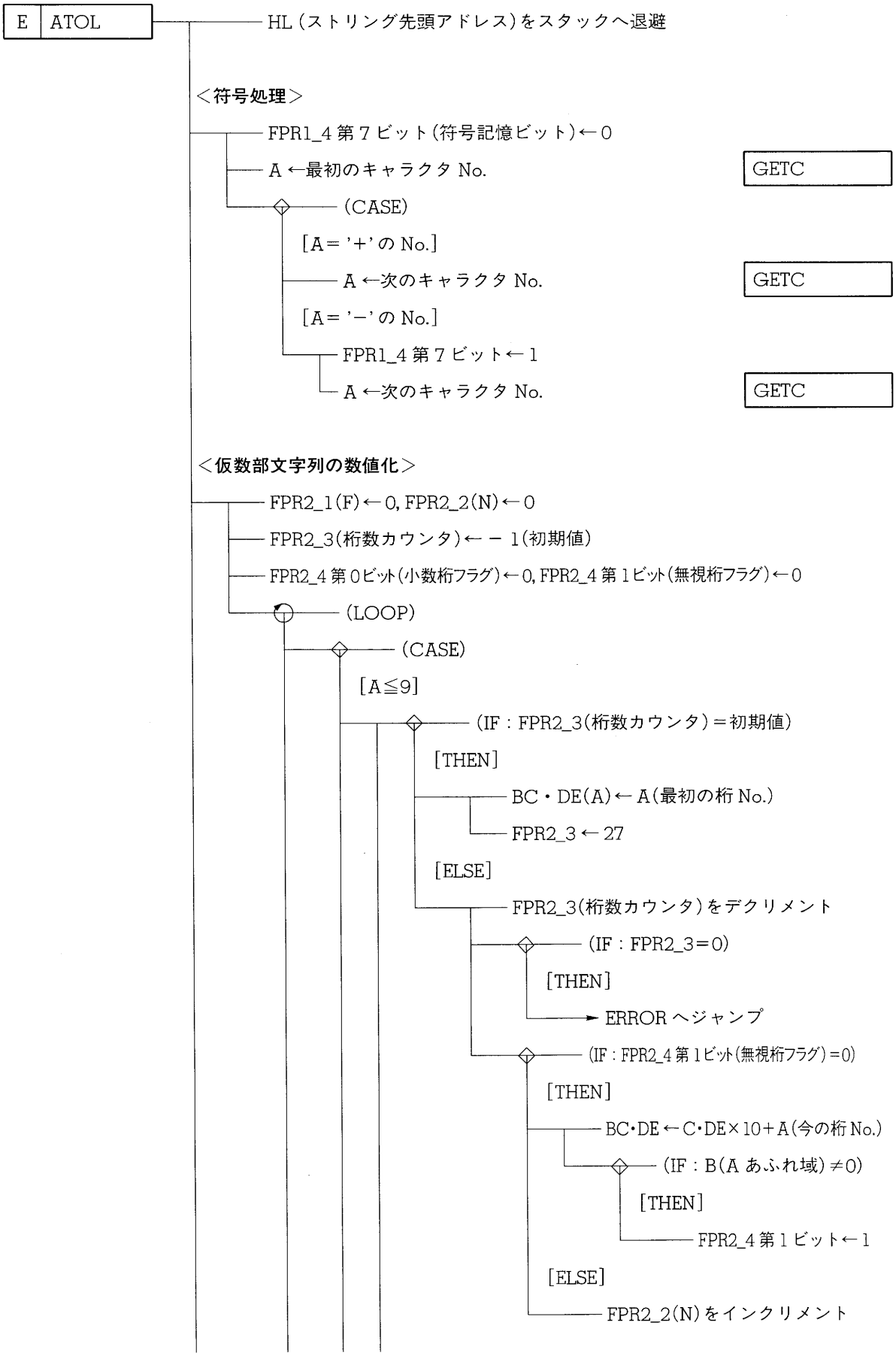
$$
\begin{aligned}
&A' \times 10^{B'} \\
&= A' \times 2^{\log_2 10 \times B'} \\
&= A' \times 2^{dec(\log_2 10 \times B')} \times 2^{int(\log_2 10 \times B')} \\
&= A' \times e^{\log 2 \times dec(\log_2 10 \times B')} \times 2^{int(\log_2 10 \times B')}
\end{aligned}
$$

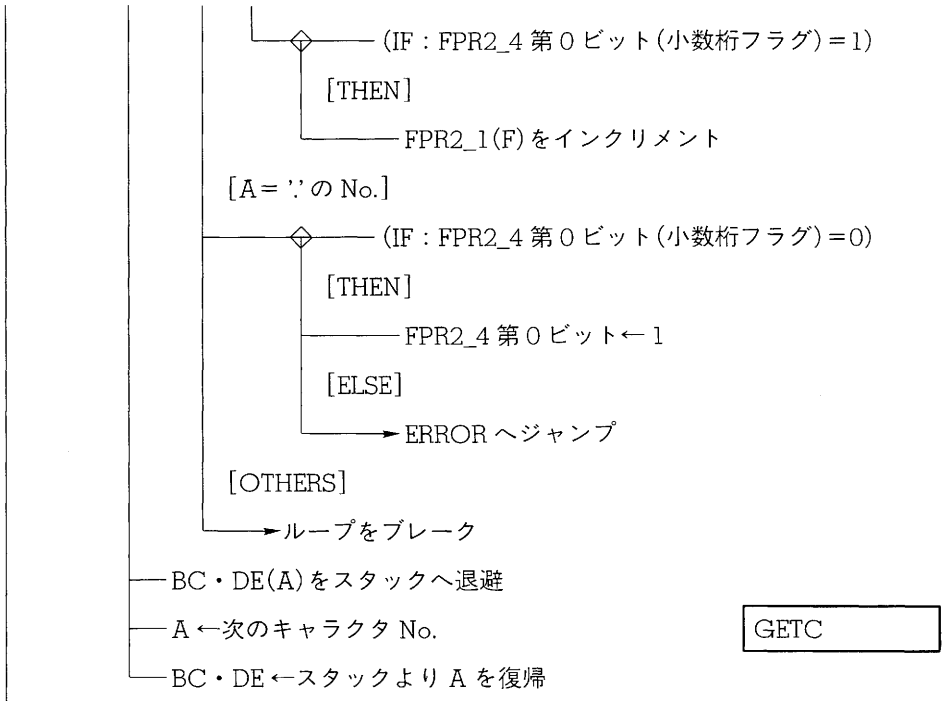
備考　dec(x) は x の小数部，int(x) は x の整数部を表します。

(11) 浮動小数点定数データ

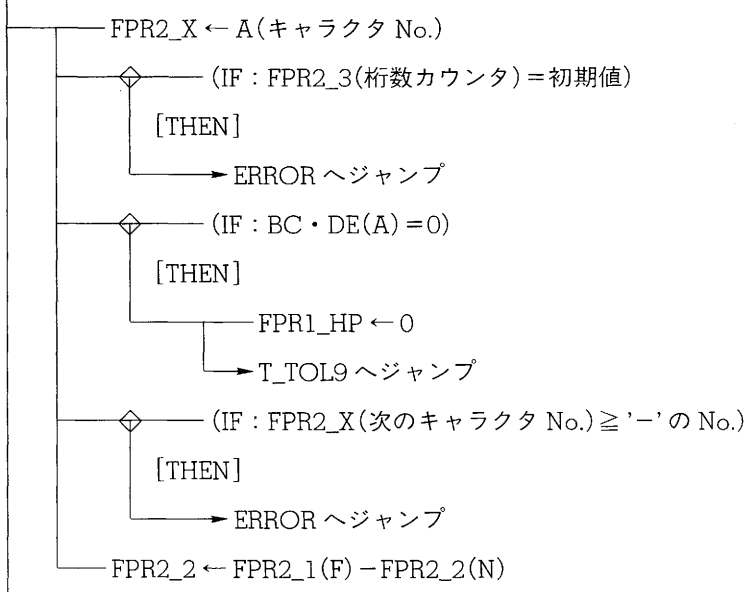
拡張仮数部を持つ定数データ $\log_2 10$ と log2 を使用しています。

(12) 処 理 図

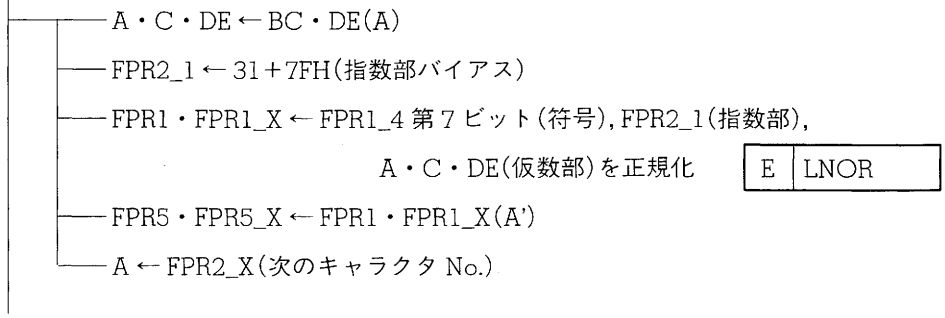

E ATOL
HL (ストリング先頭アドレス)をスタックへ退避
＜符号処理＞
FPR1_4 第７ビット(符号記憶ビット)←0
A ←最初のキャラクタ No.
GETC
(CASE)
[A＝'+'の No.]
A ←次のキャラクタ No.
GETC
[A＝'−'の No.]
FPR1_4 第７ビット←1
A ←次のキャラクタ No.
GETC
＜仮数部文字列の数値化＞
FPR2_1(F)←0, FPR2_2(N)←0
FPR2_3(桁数カウンタ)←−1(初期値)
FPR2_4 第0ビット(小数桁フラグ)←0, FPR2_4 第1ビット(無視桁フラグ)←0
(LOOP)
(CASE)
[A≦9]
(IF : FPR2_3(桁数カウンタ)＝初期値)
[THEN]
BC・DE(A)← A(最初の桁 No.)
FPR2_3 ← 27
[ELSE]
FPR2_3(桁数カウンタ)をデクリメント
(IF : FPR2_3＝0)
[THEN]
ERROR へジャンプ
(IF : FPR2_4 第1ビット(無視桁フラグ)＝0)
[THEN]
BC・DE ← C・DE×10＋A(今の桁 No.)
(IF : B(A あふれ域)≠0)
[THEN]
FPR2_4 第1ビット←1
[ELSE]
FPR2_2(N)をインクリメント

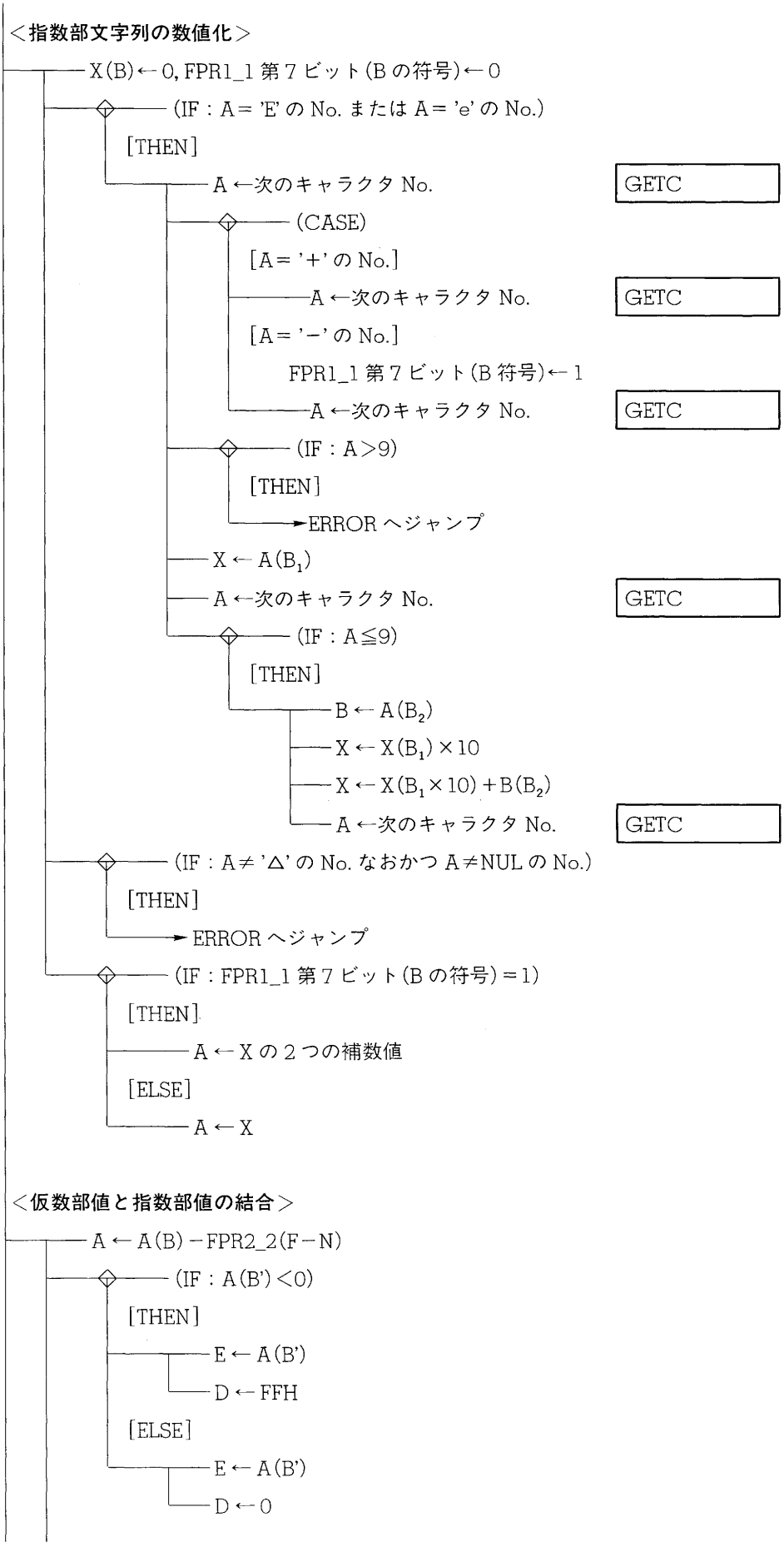


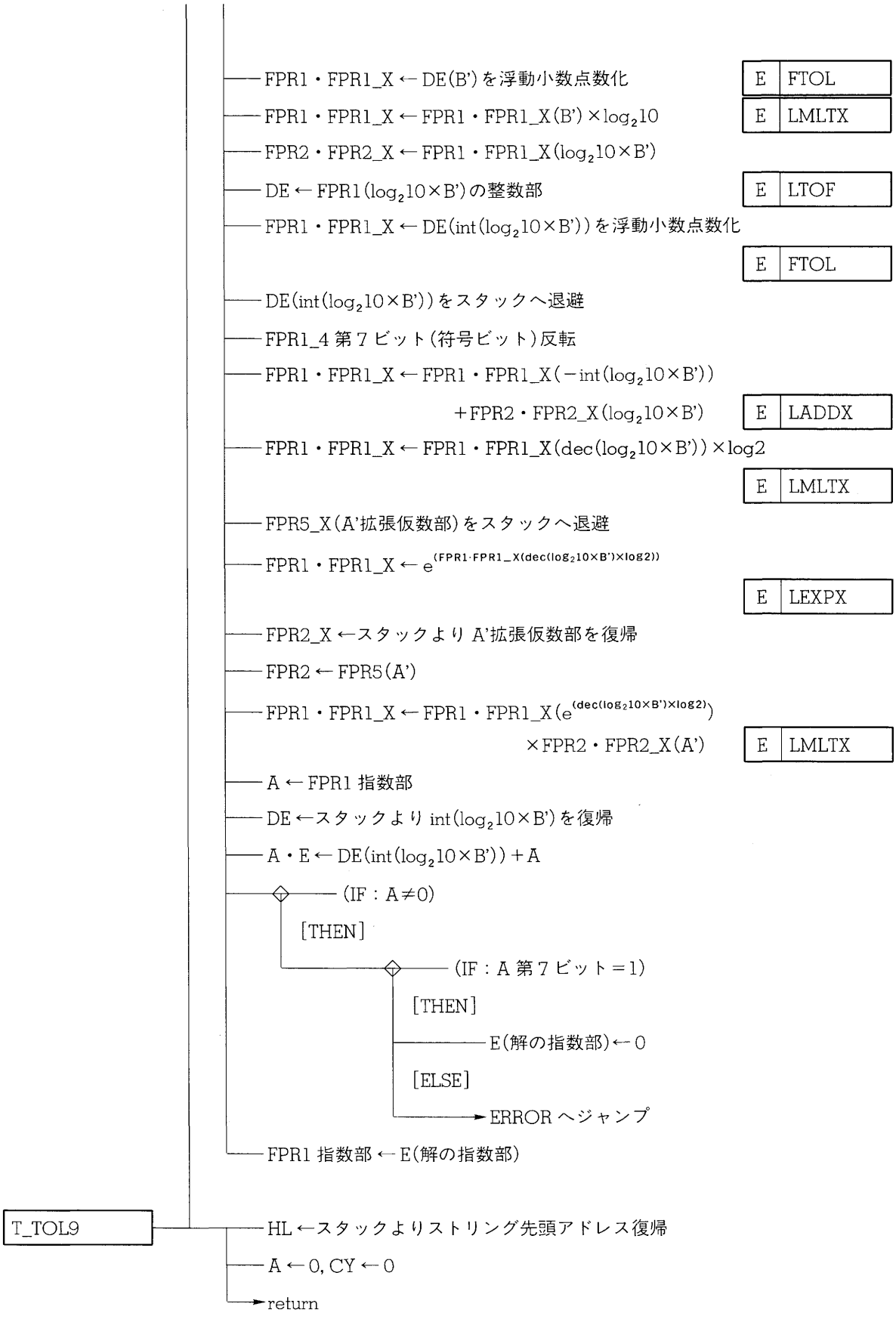
＜例外処理＞

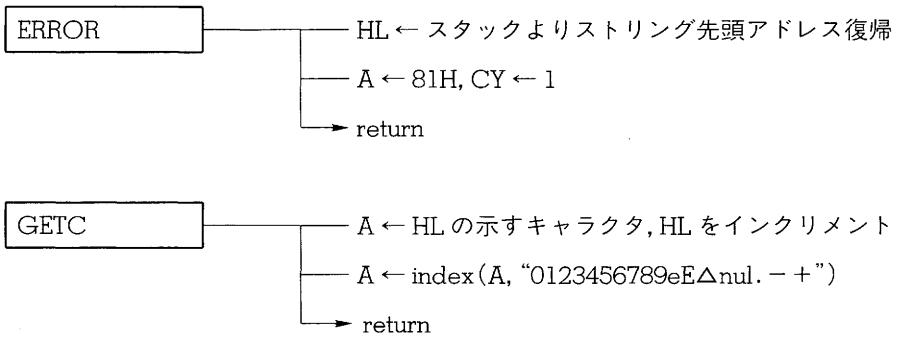


＜仮数部 **A** の正規化＞




＜指数部文字列の数値化＞
X(B)←0, FPR1_1 第7ビット(Bの符号)←0
(IF：A＝'E' の No. または A＝'e' の No.)
[THEN]
A←次のキャラクタ No.
GETC
(CASE)
[A＝'+' の No.]
A←次のキャラクタ No.
GETC
[A＝'−' の No.]
FPR1_1 第7ビット(B符号)←1
A←次のキャラクタ No.
GETC
(IF：A＞9)
[THEN]
ERROR へジャンプ
X←A(B1)
A←次のキャラクタ No.
GETC
(IF：A≦9)
[THEN]
B←A(B2)
X←X(B1)×10
X←X(B1×10)＋B(B2)
A←次のキャラクタ No.
GETC
(IF：A≠'△' の No. なおかつ A≠NUL の No.)
[THEN]
ERROR へジャンプ
(IF：FPR1_1 第7ビット(Bの符号)＝1)
[THEN]
A←X の2つの補数値
[ELSE]
A←X
＜仮数部値と指数部値の結合＞
A←A(B)−FPR2_2(F−N)
(IF：A(B')＜0)
[THEN]
E←A(B')
D←FFH
[ELSE]
E←A(B')
D←0


T_TOL9
HL←スタックよりストリング先頭アドレス復帰
A←0, CY←0
return
FPR1・FPR1_X←DE(B')を浮動小数点数化
E FTOL
FPR1・FPR1_X←FPR1・FPR1_X(B')×log2 10
E LMLTX
FPR2・FPR2_X←FPR1・FPR1_X(log2 10×B')
DE←FPR1(log2 10×B')の整数部
E LTOF
FPR1・FPR1_X←DE(int(log2 10×B'))を浮動小数点数化
E FTOL
DE(int(log2 10×B'))をスタックへ退避
FPR1_4 第7ビット(符号ビット)反転
FPR1・FPR1_X←FPR1・FPR1_X(−int(log2 10×B'))
+FPR2・FPR2_X(log2 10×B')
E LADDX
FPR1・FPR1_X←FPR1・FPR1_X(dec(log2 10×B'))×log2
E LMLTX
FPR5_X(A'拡張仮数部)をスタックへ退避
FPR1・FPR1_X←e^(FPR1・FPR1_X(dec(log2 10×B')×log2))
E LEXPX
FPR2_X←スタックよりA'拡張仮数部を復帰
FPR2←FPR5(A')
FPR1・FPR1_X←FPR1・FPR1_X(e^(dec(log2 10×B')×log2))
×FPR2・FPR2_X(A')
E LMLTX
A←FPR1 指数部
DE←スタックより int(log2 10×B')を復帰
A・E←DE(int(log2 10×B'))+A
(IF：A≠0)
[THEN]
(IF：A 第7ビット＝1)
[THEN]
E(解の指数部)←0
[ELSE]
ERROR へジャンプ
FPR1 指数部←E(解の指数部)



備考　index（キャラクタ，ストリング）はキャラクタのストリング中の位置（0-16）を返します。
キャラクタがストリング中に存在しなかった場合，0FFH が返されます。

6.2　浮動小数点形式 → 文字列への変換関数（LTOA）

(1) 処理内容

FPR1 の値を文字列に変換し，HL レジスタが示すアドレスより格納します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LLOG, LLOG10, LEXP, LEXP10, LTOA, FTOL, LTOF

(3) 消費スタック・サイズ

12（LTOA からの戻り番地２バイトを含む）

(4) 使用レジスタ

AX, BC, DE, HL（HL レジスタの内容は保存されます）

(5) 使用ワーク・エリア

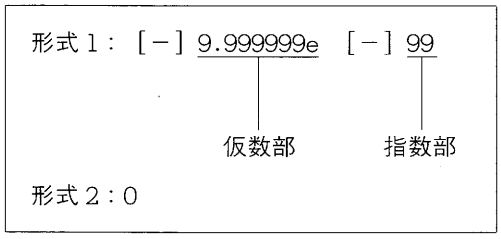
FPR1, FPR2, FPR3, FPR4, FPR5, FPR1_X, FPR2_X, FPR3_X, FPR4_X, FPR5_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：9405 μs

最大：10757 μs（−1.2677555e＋13）

(7) 出力文字列の形式



規定　(a) 0 の場合にのみ形式２を出力し，ほかはすべて形式１を用います。

(b) 文字列の終わりに 'NUL' コードが付加されます。

(c) 形式１の仮数部，および指数部の文字列長は，それぞれ 8，2 で固定となります。

(d) 形式１の仮数部，および指数部のサインは，負の場合にのみ付加されます。

(e) 'NUL' コードを含めて，最大文字列長は 14 となります。

(8) 処理手順

(a) 浮動小数点値 x＝0 なら，文字列 “0NUL” を出力し終了します。

(b) 次式により，x を $a\times10^{b}$ に変換します。

$$b=\mathrm{floor}(\log_{10}(|x|)),\quad b\neq38\rightarrow a=x\times10^{-b}$$
$$b=38\rightarrow a=x/10^{38}$$

ここで，floor(x) は x から負の方向へ向かって一番近い整数を示します。

b＝38 のとき，10^{-b} を直接計算しないのは，10^{-38} が 78K/0 の浮動小数点系ではアンダフローになってしまうからです。

(c) a＜0 なら '－' を出力し，a←|a| とします。

(d) a は数学的には 1≦a＜10 ですが，実際には計算誤差により，a＜1 または a≧10 となり得ます。

この場合，次の補正を行います。

a≧10 なら，a←a/10，b←b＋1
a＜1 なら，a←a×10，b←b－1

(e) 1≦a＜10 のため，小数点位置は固定となります。

下に示す計算結果より，文字列 "A_1，A_2，……，A_7" を出力します。

$a_1=a$
$A_1=\mathrm{int}(a_1)$，$a_2=\mathrm{dec}(a_1)\times10$
$A_2=\mathrm{int}(a_2)$，$a_3=\mathrm{dec}(a_2)\times10$

・・・・・・・・・・・・

$A_7=\mathrm{int}(a_7)$

ここで，int(a) は a の整数部を，dec(a) は a の小数部を示します。

(f) 'e' を出力します。

(g) b＜0 なら '－' を出力し，b←|b| とします。

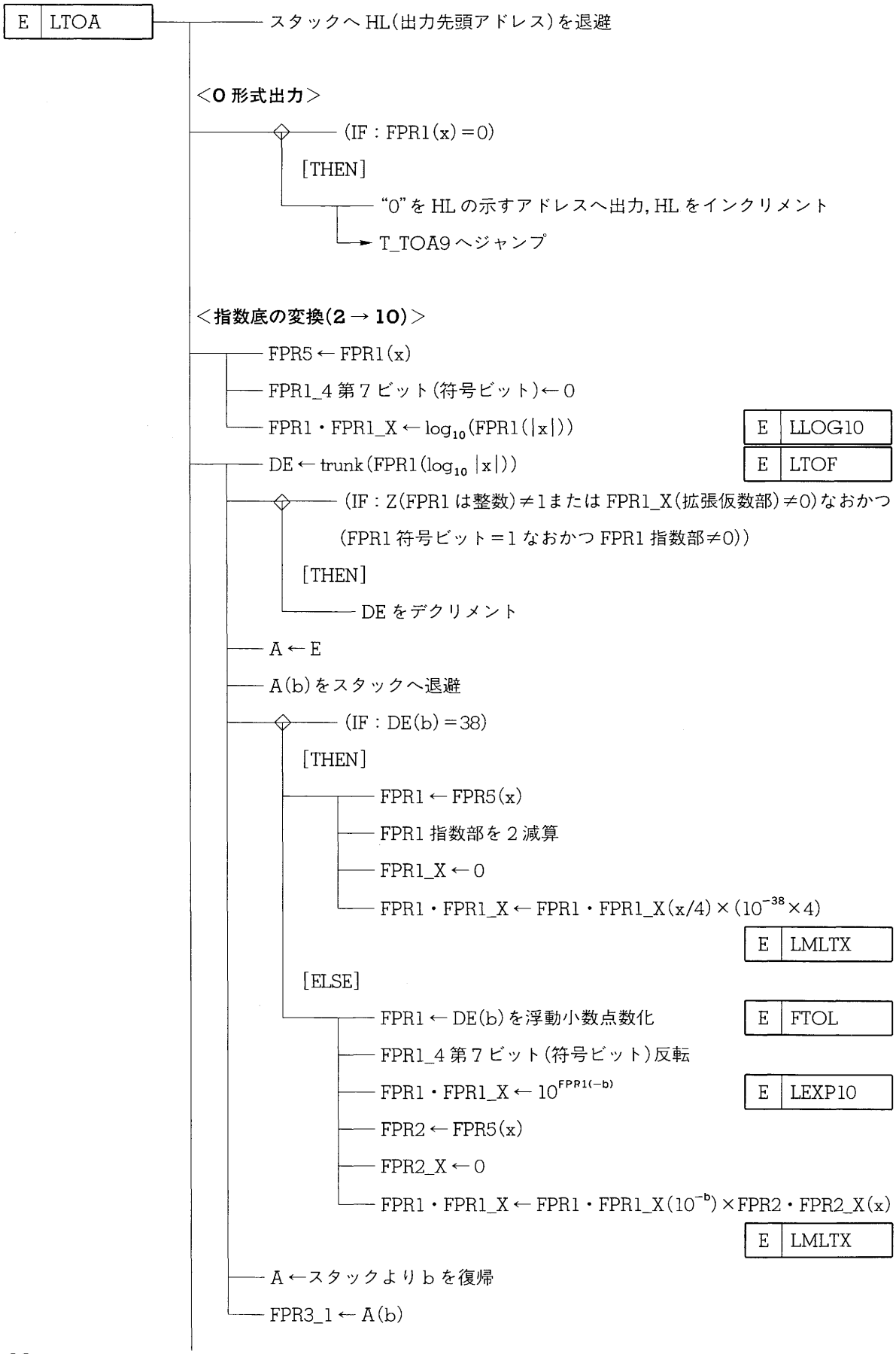
(h) b の文字列 "B_1B_2" ($B_1=b/10$，$B_2=b-B_1\times10$) を出力します。

(i) 'NUL' コードを出力します。

(9) 浮動小数点定数データ

拡張仮数部を持つ定数データ $10^{-38}\times4$，10，1/10 を使用しています。

(10) 処 理 図


E LTOA
スタックへ HL(出力先頭アドレス)を退避
<0 形式出力>
(IF : FPR1(x)=0)
[THEN]
"0"を HL の示すアドレスへ出力, HL をインクリメント
T_TOA9 へジャンプ
<指数底の変換(2 → 10)>
FPR5 ← FPR1(x)
FPR1_4 第7ビット(符号ビット)← 0
FPR1・FPR1_X ← log10(FPR1(|x|))
E LLOG10
DE ← trunk(FPR1(log10 |x|))
E LTOF
(IF : Z(FPR1 は整数)≠1 または FPR1_X(拡張仮数部)≠0)なおかつ
(FPR1 符号ビット=1 なおかつ FPR1 指数部≠0))
[THEN]
DE をデクリメント
A ← E
A(b)をスタックへ退避
(IF : DE(b)=38)
[THEN]
FPR1 ← FPR5(x)
FPR1 指数部を2減算
FPR1_X ← 0
FPR1・FPR1_X ← FPR1・FPR1_X(x/4)×(10^-38×4)
E LMLTX
[ELSE]
FPR1 ← DE(b)を浮動小数点数化
E FTOL
FPR1_4 第7ビット(符号ビット)反転
FPR1・FPR1_X ← 10^FPR1(-b)
E LEXP10
FPR2 ← FPR5(x)
FPR2_X ← 0
FPR1・FPR1_X ← FPR1・FPR1_X(10^-b)×FPR2・FPR2_X(x)
E LMLTX
A ←スタックより b を復帰
FPR3_1 ← A(b)

＜仮数部文字列出力＞

- HL ←スタックより出力先頭アドレス復帰
- HL をスタックへ退避
- ◇ (IF：FPR1_4 第７ビット(a の符号)＝1)
  - [THEN]
    - '−' を HL の示すアドレスへ出力, HL をインクリメント
    - FPR1_4 第７ビット←0
- HL をスタックへ退避
- ◇ (IF：FPR1・FPR1_X(|a|)≧10)
  - [THEN]
    - FPR1・FPR1_X ← FPR1・FPR1_X(|a|)×1/10

E	LMLTX
    - FPR3_1(b)をインクリメント
- ◇ (IF：FPR1・FPR1_X(|a|)＜1)
  - [THEN]
    - FPR1・FPR1_X ← FPR1・FPR1_X(|a|)×10

E	LMLTX
    - FPR3_1 をデクリメント
- DE ← int(FPR1(a_1))

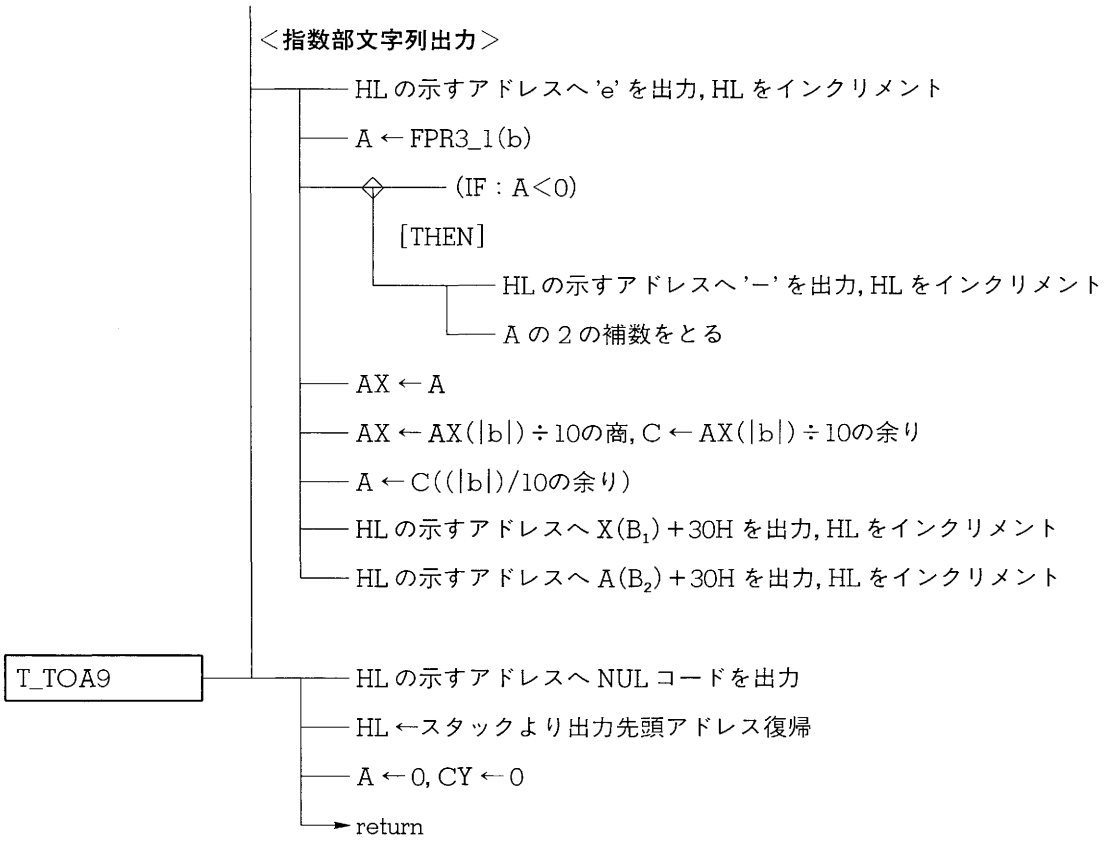
E	LTOF
- HL ←スタックより出力アドレス復帰
- HL の示すアドレスへ E(A_1)＋30H を出力, HL をインクリメント
- HL の示すアドレスへ '.' を出力, HL をインクリメント
- ↻ (FOR：i＝1 TO 6)
  - HL をスタックへ退避
  - FPR2・FPR2_X ← FPR1・FPR1_X(a_i)
  - FPR1・FPR1_X ← DE(int(a_i))を浮動小数点数化

E	FTOL
  - FPR1_4 第７ビット(符号ビット)← 1
  - FPR1・FPR1_X ← FPR1・FPR1_X(−int(a_i))
    ＋FPR2・FPR2_X(a_i)

E	LADDX
  - FPR1・FPR1_X ← FPR1・FPR1_X(dec(a_i))×10

E	LMLTX
  - DE ← int(FPR1(a_{i+1}))

E	LTOF
  - HL ←スタックより出力アドレス復帰
  - HL の示すアドレスへ E(A_{i+1})＋30H を出力, HL をインクリメント



備考　trunk（x）はxの小数部を0に向かって丸めた整数を示します。

$$\begin{bmatrix} x \geqq 0 \text{ なら}, & \text{trunk}(x) \leqq x < \text{trunk}(x)+1 \\ x < 0 \text{ なら}, & \text{trunk}(x)-1 < x \leqq \text{trunk}(x) \end{bmatrix}$$

6.3　２バイト整数型 → 浮動小数点形式への変換関数（FTOL）

(1) 処理内容

DE レジスタ内容を，符号付き２バイト整数型として浮動小数点形式に変換し，FPR1 に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，FTOL

(3) 消費スタック・サイズ

2（FTOL からの戻り番地２バイトのみ）

(4) 使用レジスタ

AX，C，DE（DE レジスタの内容は保存されます）

(5) 使用ワーク・エリア

FPR1，FPR1_X

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：40.1 μs

最大：72.3 μs（－1）

(7) ２バイト整数型形式

最上位ビットが符号ビットで，負数は２の補数表現とします。

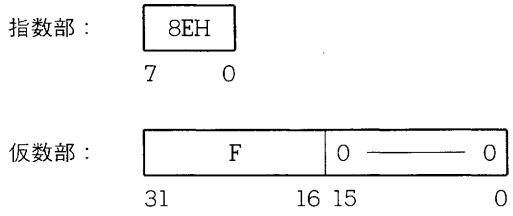
２バイト整数型 F は，－8000H～＋7FFFH の範囲の整数値をとります。

(8) 処理手順

(a) ２バイト整数値 F＝0 なら，0 を返します。

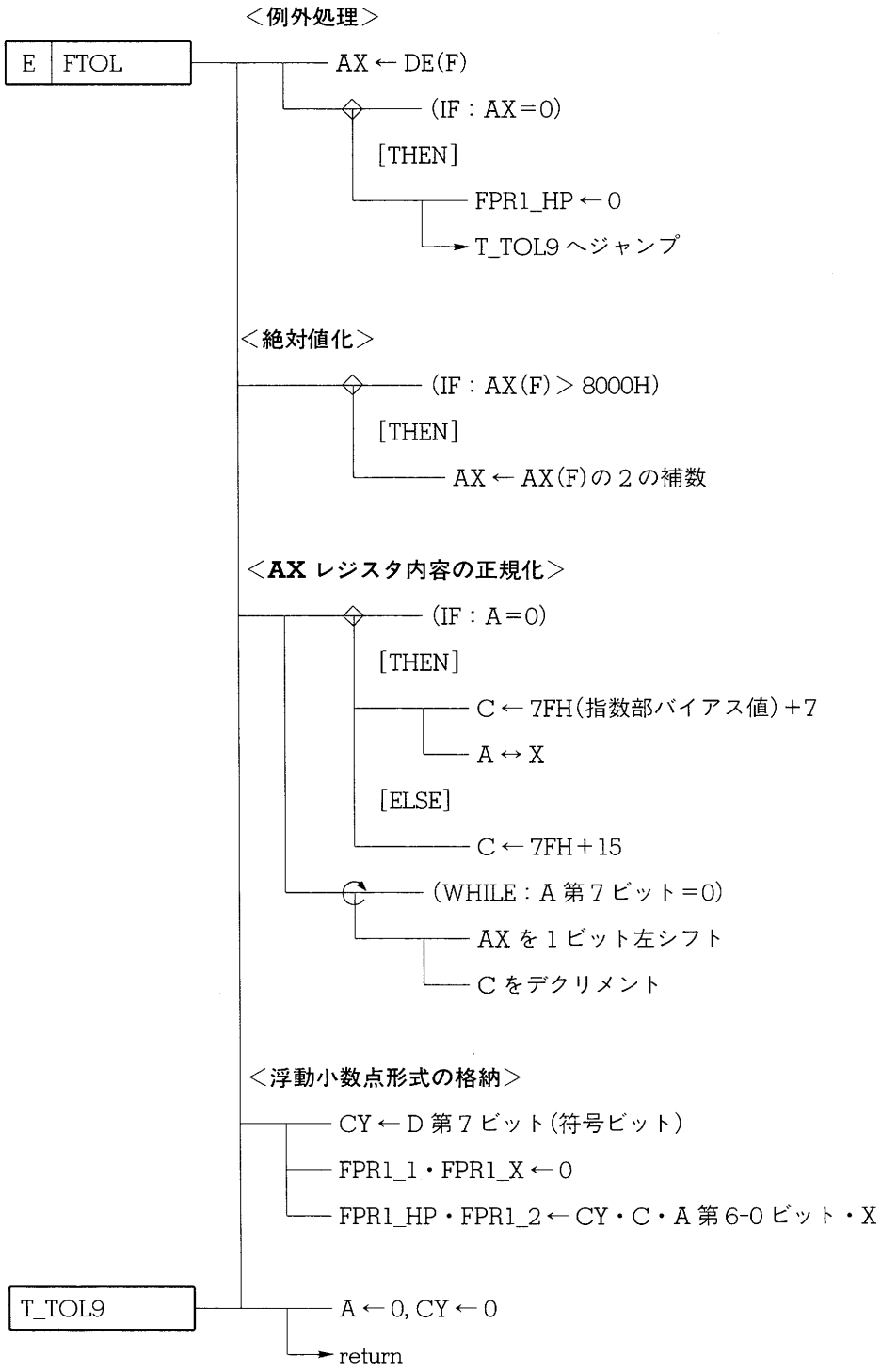
(b) －7FFFH≦F＜0 なら，F の２の補数をとります。

(c) 指数部および仮数部の状態を次に示す初期状態と考え，正規化を行うことにより，浮動小数点形式へ変換します。



(d) (c) で得た指数部，仮数部，および F の符号ビットを FPR1 へ格納します。

(9) 処 理 図


＜例外処理＞
E
FTOL
AX ← DE(F)
(IF : AX＝0)
[THEN]
FPR1_HP ← 0
T_TOL9 へジャンプ
＜絶対値化＞
(IF : AX(F)＞8000H)
[THEN]
AX ← AX(F)の２の補数
＜AX レジスタ内容の正規化＞
(IF : A＝0)
[THEN]
C ← 7FH(指数部バイアス値)＋7
A ↔ X
[ELSE]
C ← 7FH＋15
(WHILE : A 第７ビット＝0)
AX を１ビット左シフト
C をデクリメント
＜浮動小数点形式の格納＞
CY ← D 第７ビット(符号ビット)
FPR1_1・FPR1_X ← 0
FPR1_HP・FPR1_2 ← CY・C・A 第6-0 ビット・X
T_TOL9
A ← 0, CY ← 0
return

6.4　浮動小数点形式 → 2バイト整数型への変換関数（LTOF）

(1) 処理内容

FPR1の値を符号付き2バイト整数型へ変換し，DEレジスタに返します。

また，FPR1に小数部が含まれなかったならZフラグ＝1を，変換過程で丸め（小数部の切り捨て）を行った場合，Zフラグ＝0を返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LTOF

(3) 消費スタック・サイズ

2（LTOFからの戻り番地2バイトのみ）

(4) 使用レジスタ

AX，C，DE

(5) 使用ワーク・エリア

FPR1，FPR1_X（FPR1，FPR1_Xの内容は保存されます）

(6) 処理時間（内部システム・クロック＝8.38 MHz）

平均：62.3 μs

最大：88.5 μs（−1）

(7) 処理手順

(a) 指数部＝0なら，整数値0とZフラグ＝1を返します。

指数部＜7FHなら，整数値0とZフラグ＝0を返します。

指数部≧8FHなら，エラーを返します。

(b) 整数部をunsigned Integer形式で取り出します。

MSBをセットしておきます。

指数部＜87Hなら，第1仮数部を（86H−指数部）ビット右シフトし，整数値を得ます。

指数部≧87Hなら，第1，第2仮数部を（8EH−指数部）ビット右シフトし，整数値を得ます。

(c) unsigned Integer → Integer へ変換します。

(b) で求めた整数値と FPR1 の符号について，次に示す条件により変換します。

負かつ 8000H より大きい → エラー

負かつ 8000H 以下 → ２の補数をとる

正かつ 8000H 以上 → エラー

正かつ 8000H 未満 → そのまま

(d) (b) において，

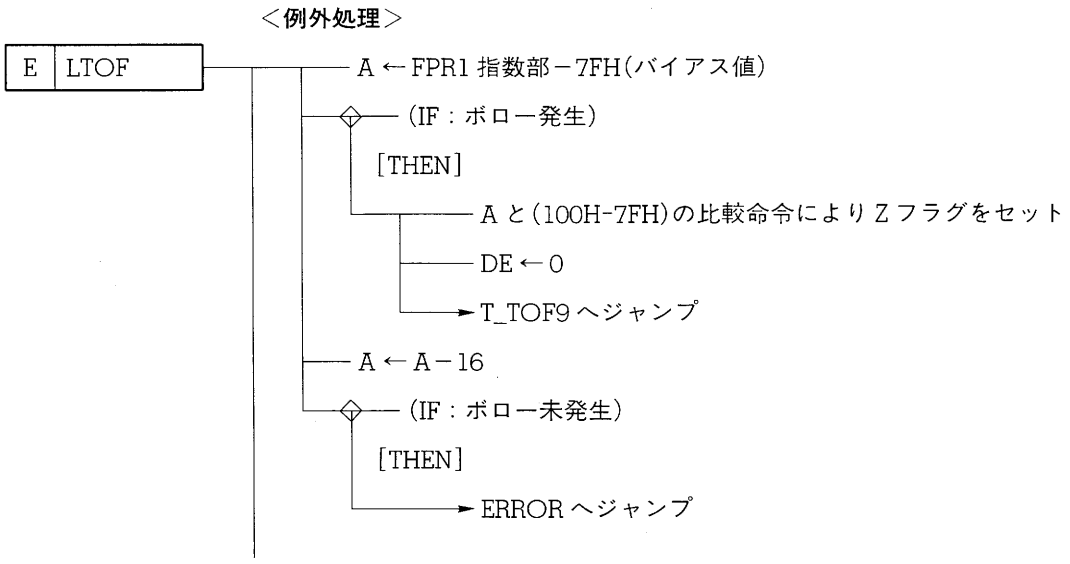
●Z フラグ＝1 を返す場合

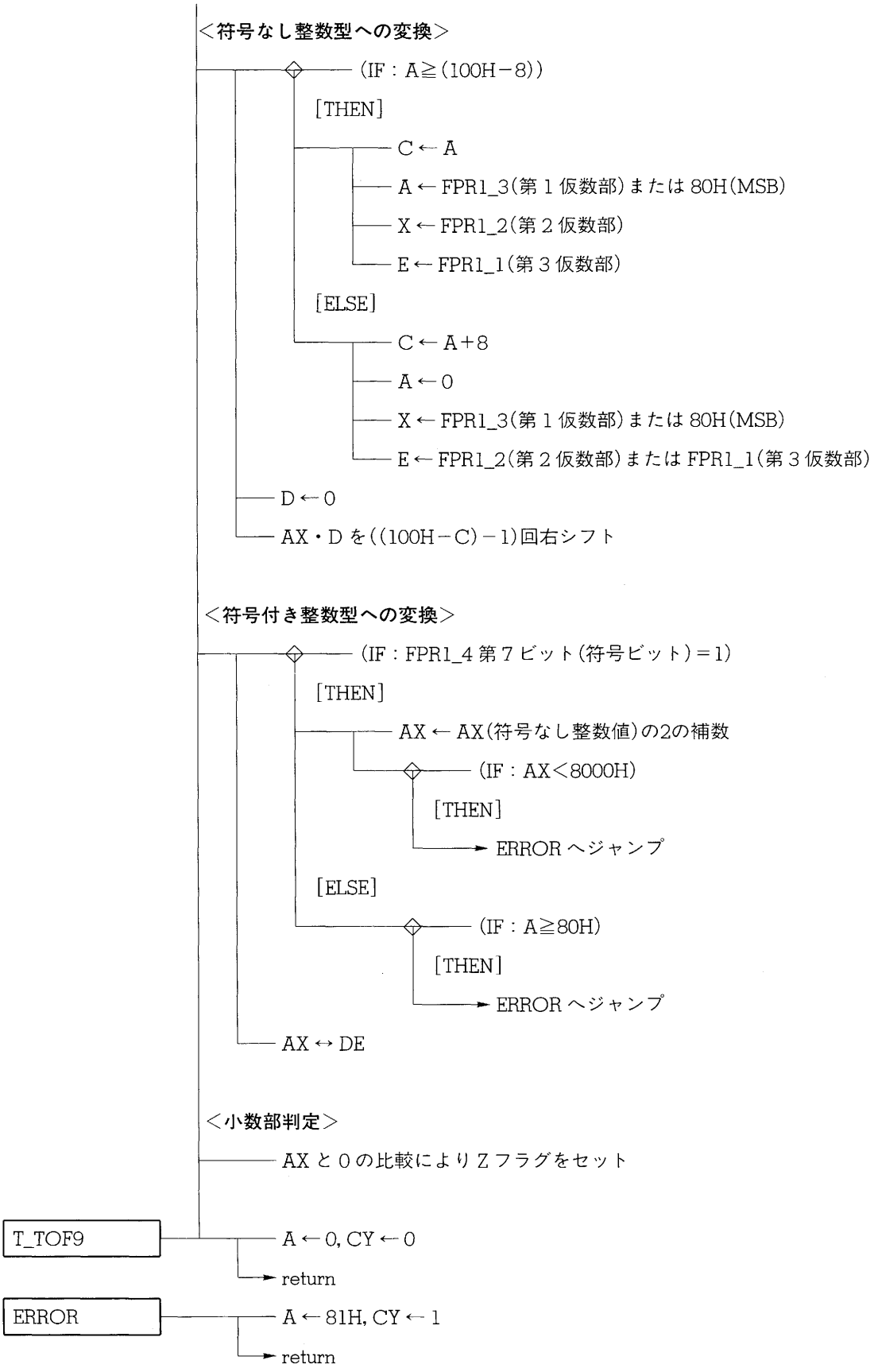
- 指数部＜87H のケースで，第 2，第 3 仮数部の全ビットが 0，かつ第 1 仮数部の右シフト過程でキャリーがでなかったとき
- 指数部≧87H のケースで，第 3 仮数部の全ビットが 0，かつ第 1，第 2 仮数部の右シフト過程でキャリーがでなかったとき

●Z フラグ＝0 を返す場合

上記以外

(8) 処 理 図




＜符号なし整数型への変換＞
(IF：A≧(100H－8))
[THEN]
C ← A
A ← FPR1_3(第１仮数部)または80H(MSB)
X ← FPR1_2(第２仮数部)
E ← FPR1_1(第３仮数部)
[ELSE]
C ← A+8
A ← 0
X ← FPR1_3(第１仮数部)または80H(MSB)
E ← FPR1_2(第２仮数部)またはFPR1_1(第３仮数部)
D ← 0
AX・Dを((100H－C)－1)回右シフト
＜符号付き整数型への変換＞
(IF：FPR1_4第７ビット(符号ビット)＝1)
[THEN]
AX ← AX(符号なし整数値)の2の補数
(IF：AX＜8000H)
[THEN]
ERRORへジャンプ
[ELSE]
(IF：A≧80H)
[THEN]
ERRORへジャンプ
AX ↔ DE
＜小数部判定＞
AXと0の比較によりZフラグをセット
T_TOF9
A ← 0, CY ← 0
return
ERROR
A ← 81H, CY ← 1
return

第7章　実行結果

この章では，各関数の演算結果と処理時間を示します。

掲載した処理時間は，次の条件による測定数値です。

CPU：μPD78P014（IE-78014-R-EM）

動作クロック：メイン・システム・クロック（内部システム・クロック ＝8.38 MHz）

コード・エリア：外部代替メモリ

ワーク・エリア：内部 RAM

スタック・エリア：0FE00H-0FE1FH

プログラマブル・ウエイト：全域ノー・ウエイト

処理時間の測定には，μPD78P014 のタイマ 0 を使用しており，数値にはタイマ値の読み出しに消費される時間（約 10 クロック）が含まれます。

演算結果には，内部形式（浮動小数点形式）と 10 進表現間の変換時の丸め誤差が含まれます。

なお，参考に PC-9801（MSC ver5.1）による演算結果を示します。

7.1　浮動小数点加算（LADD）

0+0=0（13.6 μs）

1+0=1（13.6 μs）

0+1=1（36.5 μs）

1234+98765=99999（112 μs）

4.5567831e+09+2.1447790e+06=4.5589279e+09（141 μs）

1.3e+20+4e-30=1.3e+20（29.4 μs）

223+0.111111=223.11111（141 μs）

2.1474836e+09+0.5=2.1474836e+09（29.4 μs）

3.4028237e+38+3.4028237e+38=異常終了（48.4 μs）

0.5+（-0.5）=0（53.7 μs）

1.7632415e-38+（-1.1754944e-38）=0（63.7 μs）

1.0737418e+09+0.5=1.0737418e+09（293 μs）

0.5+（-0.50000006）=-5.9604645e-08（332 μs）

7.2　浮動小数点減算（LSUB）

0-0=0（16.5 μs）

1-0=1（16.5 μs）

0-1=-1（39.4 μs）

1.7014110e+38-1e+32=1.7014le+38（223 μs）

2.3352e-05-9.99999e-21=2.3352e-05（32.2 μs）

3.7634668e-24-1.2=-1.2（42.2 μs）

9.8999999e-38-2.2000001e-38=7.6999998e-38（104 μs）

12356565-45876=12310689（131 μs）

1.2345679e+08-1.2345679e+08=0（56.6 μs）

125654-988656=-863002（103 μs）

0.5-0.50000006=-5.9604645e-08（335 μs）

7.3　浮動小数点乗算（LMLT）

0×0=0（16.9 μs）

1×0=0（16.9 μs）

0×1=0（22.2 μs）

1.701411e+38×0.1=1.701411e+37（132 μs）

6.5436653e+08×12345=8.0781548e+12（138 μs）

1.2345679e+08×1.2345679e+08=1.5241579e+16（132 μs）

1e+30×0=0（16.9 μs）

1e+30×1e-10=1e+20（132 μs）

1.234e+20×2.34e+02=2.8875600e+22（132 μs）

5.1042355e+38×1.5=異常終了（121 μs）

2.5521178e+38×1.5=3.8281766e+38（132 μs）

1×1.1754944e-38=1.1754944e-38（136 μs）

2×2=4（138 μs）

7.4　浮動小数点除算（LDIV）

0÷1=0（16.9 μs）

1.701411e+38÷2=8.5070551e+37（602 μs）

1÷0=異常終了　（11.7 μs）

9.9999997e+37÷1e+08=9.9999994e+29（514 μs）

12÷21=0.57142854（524 μs）

1.1754944e-38÷2=0（25.1 μs）

3.4028237e+38÷0.25=異常終了（21.7 μs）

（-2.3509887e-38）÷2=-1.1754944e-38（479 μs）

1÷8.5070592e+37=1.1754944e-38（479 μs）

1.9999999÷1=1.9999999（620 μs）

7.5　sin 関数（LSIN）

	μPD78P014	PC-9801
sin(3.1415927)	−8.7544322e−08 (1.34 ms)	−8.7422780e−08
sin(1.5707964)	0.99999995 (3.70 ms)	1
sin(100)	−0.50636563 (3.34 ms)	−0.50636564
sin(9999999)	0.98960368 (4.01 ms)	0.99066465
sin(−100)	0.50636563 (3.34 ms)	0.50636564
sin(0)	0 (204 μs)	0
sin(0.2)	0.19866933 (2.18 ms)	0.19866933
sin(−32.967228)	−0.99980993 (3.19 ms)	−0.99980998
sin(33.333332)	0.94053001 (3.37 ms)	0.94053001
sin(6.2831593)	−2.6050024e−05 (1.41 ms)	−2.6051198e−05
sin(9.424778)	−2.6077032e−08 (1.38 ms)	−2.3849761e−08
sin(6.8056469e+38)	0.93973852 (8.01 ms)	桁落ちによるエラー

7.6　cos 関数（LCOS）

	μPD78P014	PC-9801
cos(0)	0.99999995 (2.69 ms)	1
cos(3.1415927)	−0.99999995 (3.68 ms)	−1
cos(1.5707964)	−4.3772161e−08 (1.35 ms)	−4.3711390e−08
cos(−10000)	−0.95215377 (3.40 ms)	−0.95215537
cos(8.3775806)	−0.5000002 (3.30 ms)	−0.5000002
cos(−9424.7783)	0.99999988 (3.57 ms)	0.99999994
cos(162.31561)	0.49999341 (3.25 ms)	0.49999338
cos(6.8056469e+38)	0.34189401 (7.80 ms)	桁落ちによるエラー
cos(4.712389)	1.3038516e−08 (1.39 ms)	1.1924880e−08

7.7　tan 関数（LTAN）

	μPD78P014		PC-9801
tan(0)	0	(2.94 ms)	0
tan(3.1415927)	8.7544322e−08	(4.76 ms)	8.7422780e−08
tan(1.5707964)	−22845568	(4.81 ms)	−22877332
tan(−1.0471976)	−1.7320508	(6.08 ms)	−1.7320509
tan(2.0999999)	−1.7098470	(6.71 ms)	−1.7098469
tan(1000)	1.4703251	(6.71 ms)	1.4703242
tan(500)	0.52924407	(6.69 ms)	0.52924386
tan(157.07964)	2.9802322e−06	(4.69 ms)	2.9406275e−06
tan(6.8056469e+38)	2.7486253	(11.04 ms)	桁落ちによるエラー
tan(4.712389)	−7.6695840e+07	(4.87 ms)	−8.3858283e+07

7.8　自然対数関数（LLOG）

	μPD78P014		PC-9801
log(2.7182817)	0.99999996	(3.41 ms)	0.99999997
log(9.9999996e+35)	82.893063	(3.07 ms)	82.893063
log(1)	0	(495 μs)	0
log(0)	異常終了	(11.7 μs)	引数不正
log(−0.1)	異常終了	(10.3 μs)	引数不正
log(12345.679)	9.4210614	(3.66 ms)	9.4210614
log(59874.141)	11	(2.87 ms)	11
log(20.085537)	3	(3.29 ms)	3
log(4.5399931e−05)	−9.9999999	(3.57 ms)	−10
log(6.8056469e+38)	89.415986	(2.06 ms)	89.415986
log(1.1754944e−38)	−87.336545	(632 μs)	−87.336545
log(1.4397301e−25)	−57.200172	(3.66 ms)	−57.200172

7.9　常用対数関数（LLOG10）

	μPD78P014		PC-9801
$\log_{10}(0)$	異常終了	(16.2 μs)	引数不正
$\log_{10}(-1)$	異常終了	(14.8 μs)	引数不正
$\log_{10}(1)$	0	(538 μs)	0
$\log_{10}(10)$	0.99999999	(3.43 ms)	1
$\log_{10}(1.2345679e+08)$	8.091515	(2.67 ms)	8.091515
$\log_{10}(9.8765434e+08)$	8.994605	(2.67 ms)	8.994605
$\log_{10}(0.44400001)$	−0.35261702	(3.15 ms)	−0.35261702
$\log_{10}(100000)$	5	(3.53 ms)	5
$\log_{10}(6.8056469e+38)$	38.832869	(2.21 ms)	38.832869
$\log_{10}(1.1754944e-38)$	−37.929779	(784 μs)	−37.929779
$\log_{10}(2.4001264e-18)$	−17.619766	(3.80 ms)	−17.619766

7.10　指数関数（底＝e）（LEXP）

	μPD78P014		PC-9801
$e^{(0)}$	1	(342 μs)	1
$e^{(1)}$	2.7182818	(3.67 ms)	2.7182818
$e^{(-1)}$	0.36787944	(3.87 ms)	0.36787944
$e^{(0.98765433)}$	2.6849291	(3.67 ms)	2.6849291
$e^{(20)}$	4.8516519e+08	(3.85 ms)	4.851652e+08
$e^{(11)}$	59874.142	(3.88 ms)	59874.142
$e^{(89.415993)}$	異常終了	(214 μs)	オーバフロー
$e^{(89.415985)}$	6.8056386+38	(2.11 ms)	6.8056393+38
$e^{(-87.336548)}$	0	(2.09 ms)	1.1754907e−38
$e^{(-87.33654)}$	1.1754998e−38	(1.92 ms)	1.1754997e−38
$e^{(-0.82129019)}$	0.43986378	(4.08 ms)	0.43986378

7.11　指数関数（底＝10）（LEXP10）

	μPD78P014	PC-9801
$10^{(38.832863)}$	6.8055425e+38 (2.16 ms)	6.8055441e+38
$10^{(-37.929775)}$	1.1755061e-38 (2.06 ms)	1.1755058e-38
$10^{(89.415993)}$	異常終了 (415 μs)	オーバフロー
$10^{(-87.336548)}$	0 (424 μs)	4.60736e-88
$10^{(0)}$	1 (387 μs)	1
$10^{(0.56666666)}$	3.686945 (3.68 ms)	3.686945
$10^{(0.96666664)}$	9.2611866 (4.01 ms)	9.2611867
$10^{(1.7333332)}$	54.116939 (4.13 ms)	54.11694
$10^{(0.34659675)}$	2.2212464 (3.84 ms)	2.2212465
$10^{(-0.35975304)}$	0.43676412 (4.24 ms)	0.43676412
$10^{(38.83287)}$	異常終了 (374 μs)	オーバフロー

7.12　べき乗関数（LPOW）

	μPD78P014	PC-9801
$(0)^{(0)}$	異常終了 (17.9 μs)	オーバフロー
$(0)^{(1)}$	0 (20.3 μs)	0
$(1)^{(0)}$	1 (892 μs)	1
$(1)^{(1)}$	1 (898 μs)	1
$(1)^{(-1)}$	1 (898 μs)	1
$(2)^{(-2)}$	0.25 (2.70 ms)	0.25
$(50)^{(2)}$	2500 (7.55 ms)	2500
$(2050)^{(2)}$	4202499.9 (4.24 ms)	4202500
$(-1)^{(2.5669999)}$	異常終了 (41.8 μs)	引数不正
$(0)^{(-9.8765001)}$	異常終了 (20.3 μs)	オーバフロー
$(1.3038405e+19)^{(2)}$	1.6999996e+38 (6.21 ms)	1.7e+38
$(9.876543)^{(1.2345679)}$	16.900803 (6.91 ms)	16.900803
$(9)^{(1.2345001)}$	15.066501 (6.48 ms)	15.066502
$(2.1900001)^{(-9.1199999)}$	7.8550622e-04 (6.90 ms)	7.8550618e-04
$(4)^{(6.8056469e+38)}$	異常終了 (819 μs)	オーバフロー
$(1.50487e+12)^{(-0.20180109)}$	3.4879273e-03 (7.84 ms)	3.4879272e-03
$(2.7182817)^{(89.415985)}$	6.8056125e+38 (5.63 ms)	6.8056208e+38

7.13　平方根関数（LSQRT）

	μPD78P014		PC-9801
√(0)	0	(11.7 μs)	0
√(1)	1	(1.86 ms)	1
√(2)	1.4142135	(1.96 ms)	1.4142136
√(121)	11	(1.99 ms)	11
√(2500)	50	(1.98 ms)	50
√(1e−06)	9.9999993e−04	(2.03 ms)	1e−03
√(−9.9999998e−03)	異常終了	(11.2 μs)	引数不正
√(30.863079)	5.5554547	(1.99 ms)	5.5554549
√(11.111111)	3.3333333	(1.94 ms)	3.3333333
√(5.1001101e−35)	7.1415052e−18	(2.08 ms)	7.1415055e−18

7.14　arcsin 関数（LASIN）

	μPD78P014		PC-9801
arcsin(0)	0	(2.28 ms)	0
arcsin(1)	1.5707963	(45.8 μs)	1.5707963
arcsin(−1)	−1.5707963	(46.8 μs)	−1.5707963
arcsin(−0.5)	−0.52359877	(5.70 ms)	−0.52359878
arcsin(3.1415927)	異常終了	(21.0 μs)	引数不正
arcsin(0.78539819)	0.90333916	(6.35 ms)	0.90333915
arcsin(−0.86602539)	−1.0471976	(6.31 ms)	−1.0471975
arcsin(0.98437494)	1.3937883	(6.67 ms)	1.3937883
arcsin(0.99999994)	1.5704504	(5.16 ms)	1.5704511

7.15 arccos 関数（LACOS）

	μPD78P014		PC-9801
arccos(0)	1.5707963	(2.34 ms)	1.5707963
arccos(1)	0	(121 μs)	0
arccos(−1)	3.1415927	(132 μs)	3.1415927
arccos(0.52359879)	1.0197267	(5.25 ms)	1.0197267
arccos(−0.5)	2.0943951	(5.81 ms)	2.0943951
arccos(−0.86602539)	2.6179939	(6.40 ms)	2.6179938
arccos(0.1)	1.4706289	(4.92 ms)	1.4706289
arccos(−0.1)	1.6709638	(4.91 ms)	1.6709637
arccos(0.98437494)	0.17700801	(6.79 ms)	0.17700802
arccos(0.99999994)	3.4594024e−04	(5.38 ms)	3.4526698e−04

7.16 arctan 関数（LATAN）

	μPD78P014		PC-9801
arctan(0)	0	(289 μs)	0
arctan(1)	0.78539817	(3.70 ms)	0.78539816
arctan(−1)	−0.78539817	(3.70 ms)	−0.78539816
arctan(3.1415927)	1.2626273	(3.80 ms)	1.2626273
arctan(1.5707964)	1.0038848	(3.04 ms)	1.0038848
arctan(1.7014110e+38)	1.5707963	(838 μs)	1.5707963
arctan(10000000)	1.5707962	(1.60 ms)	1.5707962
arctan(0.001)	9.9999971e−04	(1.36 ms)	9.9999971e−04
arctan(10)	1.4711277	(2.65 ms)	1.4711277
arctan(−10)	−1.4711277	(2.65 ms)	−1.4711277
arctan(1.0000001)	0.78539823	(3.84 ms)	0.78539822

7.17　sinh 関数（LHSIN）

	μPD78P014		PC-9801
sinh(89.415993)	異常終了	(225 μs)	3.4028456e+38
sinh(−89.415993)	異常終了	(225 μs)	−3.4028456e+38
sinh(89.415985)	3.4028192e+38	(2.23 ms)	3.4028196e+38
sinh(−89.415985)	−3.4028192e+38	(2.23 ms)	−3.4028196e+38
sinh(0)	0	(187 μs)	0
sinh(0.4998779)	0.52095762	(1.65 ms)	0.52095763
sinh(0.125)	0.12532578	(1.86 ms)	0.12532578
sinh(1.0842022e−19)	1.0842022e−19	(317 μs)	1.0842022e−19
sinh(0.76666665)	0.84400989	(4.10 ms)	0.84400998
sinh(1.0666666)	1.2807617	(4.50 ms)	1.2807619
sinh(1.8666666)	3.1560328	(4.62 ms)	3.1560329
sinh(3.351413)	14.254	(4.78 ms)	14.254001

7.18　cosh 関数（LHCOS）

	μPD78P014		PC-9801
cosh(−89.415993)	異常終了	(220 μs)	3.4028456e+38
cosh(−89.415985)	3.4028192e+38	(2.22 ms)	3.4028196e+38
cosh(0)	1	(935 μs)	1
cosh(1.0842022e−19)	1	(1.31 ms)	1
cosh(0.76666665)	1.3085689	(4.07 ms)	1.308569
cosh(1.0666666)	1.6249156	(4.48 ms)	1.6249157
cosh(1.8666666)	3.3106711	(4.59 ms)	3.3106712
cosh(4.0319099)	28.193103	(4.77 ms)	28.193105

7.19　tanh 関数（LHTAN）

	μPD78P014		PC-9801
tanh(89.415993)	1.0000001	(254 μs)	1
tanh(−89.415993)	−1.0000001	(255 μs)	−1
tanh(0)	0	(1.18 ms)	0
tanh(0.4998779)	0.46202111	(6.63 ms)	0.46202113
tanh(0.125)	0.124353	(6.84 ms)	0.124353
tanh(1.0842022e−19)	1.0842022e−19	(2.14 ms)	1.0842022e−19
tanh(0.76666665)	0.64498699	(8.74 ms)	0.64498699
tanh(1.0666666)	0.78820205	(9.53 ms)	0.78820205
tanh(1.8666666)	0.953291	(9.74 ms)	0.95329096
tanh(9.4484739)	0.99999988	(10.02 ms)	0.99999999
tanh(7.8125e−03)	7.8123417e−03	(4.92 ms)	7.8123411e−03

7.20　絶対値関数（LABS）

|0|	=0	(6.4 μs)
|−2.1290744e−19|	=2.1290744e−19	(6.4 μs)
|1.6415355e+27|	=1.6415355e+27	(6.4 μs)

7.21　逆数関数（LRCPN）

1/0	=異常終了	(38.9 μs)
1/(−1.1754944e−38)	=−8.5070592e+37	(505 μs)
1/0.99999994	= 1	(490 μs)
1/(−1.0000001)	=−0.99999988	(636 μs)
1/(8.5070602e+37)	= 0	(637 μs)
1/(8.5070592e+37)	= 1.1754944e−38	(506 μs)
1/(2.8545976e−18)	= 3.5031207e+17	(541 μs)
1/(−2.9092885e+21)	=−3.4372664e−22	(549 μs)

7.22　極座標 → 直交座標への変換関数（POTORA）

(r, θ)	(x, y)	
(1, 1.5707964)	(−4.3772161e−08, 0.99999995)	(4.63 ms)
	(−4.3711390e−08, 1)	
(1, 0.52359879)	(0.8660254, 0.50000001)	(5.91 ms)
	(0.8660254, 0.50000001)	
(7, 2.6179938)	(−6.0621777, 3.5000003)	(6.48 ms)
	(−6.0621777, 3.5000003)	
(99, −2.0943952)	(−49.500005, −85.736512)	(6.51 ms)
	(−49.500005, −85.736512)	
(8.8888798, 2.3561945)	(−6.2853872, 6.2853871)	(6.74 ms)
	(−6.2853872, 6.2853871)	
(4.4443998, 3.1415927)	(−4.4443996, −3.8908197e−07)	(4.61 ms)
	(−4.4443998, −3.8854179e−07)	
(0.5, 6.8056469e+38)	(0.17094701, 0.46986926)	(10.84 ms)
	桁落ちによるエラー	
(0.5, 4.712389)	(6.5192580e−09, −0.49999997)	(4.68 ms)
	(5.9624402e−09, −0.5)	
(0.5, 6.283185)	(0.49999997, −1.5040860e−07)	(4.48 ms)
	(0.5, −1.5099580e−07)	

備考　上段に μPD78P014 による演算結果を，下段に PC-9801 による演算結果を示しています。

7.23　直交座標 → 極座標への変換関数（RATOPO）

(x, y)	(r, θ)	
(0, 0)	(0, 0)	(17.9 μs)
	(0, 0)	
(1, 1)	(1.4142135, 0.78539813)	(6.60 ms)
	(1.4142136, 0.78539816)	
(0, 1)	(1, 1.5707963)	(2.19 ms)
	(1, 1.5707963)	
(1, −1)	(1.4142135, −0.78539813)	(6.60 ms)
	(1.4142136, −0.78539816)	
(−1, 1)	(1.4142135, 2.3561943)	(6.71 ms)
	(1.4142136, 2.3561945)	
(−1, −1)	(1.4142135, −2.3561943)	(6.71 ms)
	(1.4142136, −2.3561945)	
(0, −1)	(1, −1.5707963)	(2.19 ms)
	(1, −1.5707963)	
(1, 0)	(1, 0)	(2.49 ms)
	(1, 0)	
(−1, 0)	(1, 3.1415925)	(2.54 ms)
	(1, 3.1415927)	
(11111, 11111)	(15713.326, 0.78539813)	(6.58 ms)
	(15713.327, 0.78539816)	
(−12.220954, 69.662003)	(70.725845, 1.7444612)	(6.98 ms)
	(70.725853, 1.7444613)	
(0.92719781, 0.23236816)	(0.95587164, 0.24555582)	(5.90 ms)
	(0.95587172, 0.24555586)	

備考　上段に μPD78P014 による演算結果を，下段に PC-9801 による演算結果を示しています。

7.24　文字列 → 浮動小数点形式への変換関数（ATOL）

"1234567.890123456789012345678"	=	異常終了	(1.11 ms)
"0Q"	=	0	(126 μs)
"E12"	=	異常終了	(78.8 μs)
"1e"	=	異常終了	(562 μs)
"1E+123"	=	異常終了	(581 μs)
"1.17549427E−38"	=	0	(5.39 ms)
"1.17549428E−38"	=	1.1754944e−38	(5.39 ms)
"6.8056476E+38"	=	異常終了	(4.43 ms)
"6.8056475E+38"	=	6.8056473e+38	(4.44 ms)
"655361"	=	655361	(1.20 ms)
"1e−20"	=	1e−20	(5.18 ms)
"12345678901234567890123456 7E−32"	=	1.2345679e−06	(5.62 ms)
"1.00000000000000000000000000E−9"	=	1e−09	(5.88 ms)
"+1.2030646E+22"	=	1.2030646e+22	(5.42 ms)
"−4.6231684E−18"	=	−4.6231685e−18	(4.95 ms)
"0.00000000000000000117549428E−20"	=	1.1754944e−38	(6.39 ms)

7.25　浮動小数点形式 → 文字列への変換関数（LTOA）

0	=	"0"	(18.4 μs)
1.0000002e−37	=	"1.000000e−37"	(8.32 ms)
9.9999999e−38	=	"9.999998e−38"	(9.04 ms)
1.0000001e−05	=	"1.000000e−05"	(9.73 ms)
1	=	"1.000000e00"	(1.91 ms)
100000	=	"1.000000e05"	(9.61 ms)
9.9999993e+19	=	"1.000000e20"	(9.50 ms)
9.9999987e+37	=	"9.999999e37"	(9.28 ms)
1.0000001e +38	=	"1.000000e38"	(8.85 ms)
6.8056469e+38	=	"6.805646e38"	(4.81 ms)
−6.8056469e+38	=	"−6.805646e38"	(4.81 ms)
−1.2677555e+13	=	"−1.267755e13"	(10.76 ms)
−1.1907760e−29	=	"−1.190775e−29"	(9.02 ms)

7.26　2バイト整数型 → 浮動小数点形式への変換関数（FTOL）

```
  0          (12.2 μs)
 -1          (72.3 μs)
-32768       (26.5 μs)
  1          (68.5 μs)
  511        (66.6 μs)
 -511        (70.4 μs)
  255        (28.4 μs)
  32767      (32.2 μs)
```

7.27　浮動小数点形式 → 2バイト整数型への変換関数（LTOF）

```
-0.99999994=  0 (Z フラグ=0)        (14.6 μs)
  0         =  0 (Z フラグ=1)        (14.6 μs)
  65536     =  異常終了              (14.6 μs)
-32769      =  異常終了              (35.1 μs)
-32768      = -32768 (Z フラグ=1)    (37.5 μs)
  32767.5   =  32767 (Z フラグ=0)    (38.9 μs)
  1         =  1 (Z フラグ=1)        (83.3 μs)
  1.5       =  1 (Z フラグ=0)        (83.3 μs)
-1          = -1 (Z フラグ=1)        (88.5 μs)
```

第8章　プログラム・リスト

(1) EQU. INC

```
$       NOLIST
;*******************************************************
;*
;*  78K0 COMMON NAME DEFINE
;*
;*
;*
;*******************************************************
SHORT   EQU     4       ;size of real type
INTEGR  EQU     2       ;size of integer type
BYTE    EQU     8       ;bit figures of byte
ZEROEX  EQU     7FH     ;exponent bias for 0
R_OK    EQU     0       ;normal return code
R_ERR   EQU     81H     ;abnormal return code
$       LIST
```

(2) REF1. INC

```
$       NOLIST
;*******************************************************
;*
;*  78K0 FLOATING POINT REGISTER REFFERENCE DEFINE
;*
;*
;*
;*******************************************************
        EXTRN   FPR1
        EXTRN   FPR1_LP,FPR1_HP
        EXTRN   FPR1_1,FPR1_2,FPR1_3,FPR1_4

        EXTRN   FPR2
        EXTRN   FPR2_LP,FPR2_HP
        EXTRN   FPR2_1,FPR2_2,FPR2_3,FPR2_4

        EXTRN   FPR3
        EXTRN   FPR3_LP,FPR3_HP
        EXTRN   FPR3_1,FPR3_2,FPR3_3,FPR3_4
        EXTRN   FPR4
        EXTRN   FPR4_LP,FPR4_HP
        EXTRN   FPR4_1,FPR4_2,FPR4_3,FPR4_4
        EXTRN   FPR5
        EXTRN   FPR5_LP,FPR5_HP
        EXTRN   FPR5_1,FPR5_2,FPR5_3,FPR5_4

        EXTRN   FPRE_XP
        EXTRN   FPR1_X,FPR2_X
        EXTRN   FPR3_X,FPR4_X,FPR5_X
$       LIST
```

(3) REF2. INC

```
$       NOLIST
;*************************************************************
;*
;*  78K0 FLOATING POINT REGISTER LOAD FUNCTION REFFERENCE
;*
;*
;*
;*************************************************************
        EXTRN   LLD21,LLD21X
        EXTRN   LLD31,LLD31X
        EXTRN   LLD41,LLD41X
        EXTRN   LLD51,LLD51X
        EXTRN   LLD32
        EXTRN   LLD52
        EXTRN   LLD13
        EXTRN   LLD23,LLD23X
        EXTRN   LLD24,LLD24X
        EXTRN   LLD15
        EXTRN   LLD25,LLD25X
        EXTRN   LLD1C,LLD1CX
        EXTRN   LLD2C,LLD2CX
        EXTRN   LXC13,LXC13X
        EXTRN   LXC14,LXC14X
        EXTRN   LXC15,LXC15X
$       LIST
```

(4) ASCII. INC

```
$       NOLIST
;*********************************************************************
;
; 78K0 ASCII CODE DEFINE
;
;
;
;*********************************************************************
A_PL    EQU     02BH        ;'+'
A_MN    EQU     02DH        ;'-'
A_PD    EQU     02EH        ;'.'
A_NL    EQU     000H        ;nul
A_BL    EQU     020H        ;blank
A_E     EQU     045H        ;'E'
A_E2    EQU     065H        ;'e'
A_0     EQU     030H        ;'0'
A_9     EQU     039H        ;'9'

N_PL    EQU     16
N_MN    EQU     15
N_PD    EQU     14
N_NL    EQU     13
N_BL    EQU     12
N_E     EQU     11
N_9     EQU     9

S_INDX  EQU     7

@_INDX  MACRO
        DB A_PL,A_MN,A_PD,A_NL
        DB A_BL,A_E ,A_E2
        ENDM
$       LIST
```

(5) DFLT. SRC

```
$       TITLE   ('FLOATING POINT REGISTERS')
        NAME    M_DFLT
;*******************************************************
;*
;*  78K0 FLOATING POINT REGISTERS
;*
;*
;*
;*******************************************************

        PUBLIC  FPR1
        PUBLIC  FPR1_LP,FPR1_HP
        PUBLIC  FPR1_1,FPR1_2,FPR1_3,FPR1_4
        PUBLIC  FPR2
        PUBLIC  FPR2_LP,FPR2_HP
        PUBLIC  FPR2_1,FPR2_2,FPR2_3,FPR2_4

        PUBLIC  FPR3
        PUBLIC  FPR3_LP,FPR3_HP
        PUBLIC  FPR3_1,FPR3_2,FPR3_3,FPR3_4
        PUBLIC  FPR4
        PUBLIC  FPR4_LP,FPR4_HP
        PUBLIC  FPR4_1,FPR4_2,FPR4_3,FPR4_4
        PUBLIC  FPR5
        PUBLIC  FPR5_LP,FPR5_HP
        PUBLIC  FPR5_1,FPR5_2,FPR5_3,FPR5_4

        PUBLIC  FPRE_XP
        PUBLIC  FPR1_X,FPR2_X

        PUBLIC  FPR3_X,FPR4_X,FPR5_X

        DSEG    SADDRP
```

```
;******** FLOATING POINT REGISTER 1 **
FPR1:
FPR1_LP:
FPR1_1:
        DS      1
FPR1_2:
        DS      1
FPR1_HP:
FPR1_3:
        DS      1
FPR1_4:
        DS      1

;******** FLOATING POINT REGISTER 2 **
FPR2:
FPR2_LP:
FPR2_1:
        DS      1
FPR2_2:
        DS      1
FPR2_HP:
FPR2_3:
        DS      1
FPR2_4:
        DS      1

;******** FLOATING POINT REGISTER 3 **
FPR3:
FPR3_LP:
FPR3_1:
        DS      1
FPR3_2:
        DS      1
FPR3_HP:
FPR3_3:
        DS      1
FPR3_4:
        DS      1
```

```
;******** FLOATING POINT REGISTER 4 **
FPR4:
FPR4_LP:
FPR4_1:
        DS      1
FPR4_2:
        DS      1
FPR4_HP:
FPR4_3:
        DS      1
FPR4_4:
        DS      1

;******** FLOATING POINT REGISTER 5 **
FPR5:
FPR5_LP:
FPR5_1:
        DS      1
FPR5_2:
        DS      1
FPR5_HP:
FPR5_3:
        DS      1
FPR5_4:
        DS      1

;******** FLOATING POINT REGISTER 4th MANTISSA **
FPRE_XP:
FPR1_X:
        DS      1
FPR2_X:
        DS      1

FPR3_X:
        DS      1
FPR4_X:
        DS      1
FPR5_X:
        DS      1

        END
```

(6) LFLT1. SRC

```
$       TITLE   ('THE 4 RULES FUNCTIONS')
        NAME    M_LFLT1

#include "EQU.INC"
#include "REF1.INC"

        PUBLIC  LADD,LSUB,LMLT,LDIV
        PUBLIC  LADDX,LSUBX,LMLTX
        PUBLIC  LNOR

        CSEG
;*************************************************
;*
;*  78K0 FLOATING POINT ADDITION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 += FPR2
;*             ERROR then set CY
;*
;*************************************************

LADD:
        FPRE_XP = #0            ;clear 4th mantissa
LADDX:
        CY  = FPR2_3.7
        A = FPR2_4
        ADDC A,A
        if_bit (Z)
            goto T_RET
        endif
        D = A                   ;FPR2 exponent

        CY  = FPR1_3.7
        A = FPR1_4
        ROLC A,1
        C = A                   ;FPR1 exponent

;******* CHECK EXPONENT & LOAD **
                                ;d : FPR1･FPR1_X <- one of higher exp.
                                ;s : A･DE･C <- mantissa (lower exp. one)
                                ;    FPR2_X <- difference of exp.
                                ;    FPR2_4.7 <- sign (lower exp. one)
```

```
A -= D                  ;difference of exp.

if_bit (!CY)
  A <-> FPR2_X
  A <-> C
  A <-> FPR2_1
  E = A
  D = FPR2_2 (A)
  A = FPR2_3

else
  A ^= #0FFH
  A++                   ;|difference of exp.|
  A <-> FPR2_X
  A <-> FPR1_X
  A <-> C
  A <-> D
  A <-> FPR2_1
  A <-> FPR1_1
  E = A
  A = FPR2_4
  A <-> FPR1_4
  FPR2_4 = A
  A = FPR2_2
  A <-> FPR1_2
  A <-> D
  A <-> FPR2_3
  A <-> FPR1_3

  if (FPR2_3 == #0)     ;exp. of destination == 0?
    goto T_RET
  endif

endif

if (FPR2_X >= #SHORT*BYTE)
  goto T_RET            ;neglect lower value
endif

FPR1_3 |= #80H          ;(set mantissa MSB)
A |= #80H               ;(set mantissa MSB)
```

```
;******* BE AGREED MANTISSA POTENTIAL **

        if (FPR2_X != #0)       ;exp. agree?
          repeat
            CLR1 CY
            RORC A,1
            A <-> D
            RORC A,1
            A <-> E
            RORC A,1
            A <-> C
            RORC A,1
            A <-> C
            A <-> E
            A <-> D             ;s':A･DE･C <- mantissa be agreed potential
            FPR2_X--
          until_bit (Z)
        endif

;******* CALC. MANTISSA(set result to A･C･DE) **

        CY = FPR2_4.7
        CY ^=  FPR1_4.7

        if_bit (!CY)            ;sign agree?
          A <-> C               ;s' += d
          ADD A,FPR1_X
          A <-> E
          ADDC A,FPR1_1
          A <-> D
          ADDC A,FPR1_2
          A <-> C
          ADDC A,FPR1_3

          if_bit (CY)
            FPR2_1++            ;normalize mantissa overflow
            if_bit (Z)
              goto ERROR
            endif
            RORC A,1
            A <-> C
            RORC A,1
            A <-> D
            RORC A,1
            A <-> E
            RORC A,1
            A <-> E
            A <-> D
            A <-> C
          endif
```

```
else
  X = A
  if (A == FPR1_3)
    if (D == FPR1_2) (A)
      if (E == FPR1_1) (A)
        if (C == FPR1_X) (A)     ;if(s' == d)
          goto ZERO
        endif
      endif
    endif
  endif

  if_bit (!CY)              ;if(s' > d)
    FPR1_4 ^= #80H          ;turn sign bit
    A = X
  else
    A = C
    A <-> FPR1_X
    C = A
    A = E
    A <-> FPR1_1
    E = A
    A = D
    A <-> FPR1_2
    D = A
    A = X
    A <-> FPR1_3
  endif

  A <-> C                   ;|s' -= d|
  SUB A,FPR1_X
  A <-> E
  SUBC A,FPR1_1
  A <-> D
  SUBC A,FPR1_2
  A <-> C
  SUBC A,FPR1_3
```

```
LNOR:
          while_bit (!A.7)      ;normalize catastrophic cancellation
            FPR2_1--
            if_bit (Z)
              goto ZERO
            endif
            A <-> E
            ROLC A,1
            A <-> D
            ROLC A,1
            A <-> C
            ROLC A,1
            A <-> E
            ROLC A,1
            A <-> E
            A <-> C
            A <-> D
            A <-> E
          endw
        endif

;******* STORE FPR1 **

        FPR1_3 = A              ;1st mantissa
        A = FPR2_1
T_STOR:
        CY = FPR1_4.7
        RORC A,1
        FPR1_4 = A              ;sign,exponent
        FPR1_3.7 = CY           ;exponent LSB
        FPR1_2 = C (A)          ;2nd mantissa
        FPR1_1 = D (A)          ;3rd mantissa
        FPR1_X = E (A)          ;4th mantissa

T_RET:
        A = #R_OK
        CLR1 CY
        RET
ZERO:
        FPR1_HP = #0
        goto T_RET
ERROR:
        A = #R_ERR
        SET1 CY
        RET
```

```
;*****************************************************
;*
;*  78K0 FLOATING POINT SUBTRACTION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 -= FPR2
;*             ERROR then set CY
;*
;*****************************************************

LSUB:
        FPRE_XP = #0            ;clear 4th mantissa
LSUBX:
        FPR2_4 ^= #80H
        goto LADDX


;*****************************************************
;*
;*  78K0 FLOATING POINT MULTIPLICATION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 *= FPR2
;*             ERROR then set CY
;*
;*****************************************************

LMLT:
        FPRE_XP = #0            ;clear 4th mantissa

;******* ZERO EXCEPTION **
LMLTX:
        CY = FPR2_3.7
        A = FPR2_4
        ADDC A,A
        if_bit (Z)
          goto ZERO
        endif
        C = A                   ;FPR2 exp.

        CY  = FPR1_3.7
        A = FPR1_4
        ADDC A,A                ;FPR1 exp.
        if_bit (Z)
          goto ZERO
        endif
```

```
;******* MULTIPLE EXPONENT **

        A += C
        if_bit (CY)
          A -= #ZEROEX
          if_bit (!CY)          ;exp. >= 100H
            goto ERROR
          endif
        else
          A -= #ZEROEX
          if_bit (CY)           ;exp. < 0
            goto ZERO
          endif
        endif

        A <-> FPR2_4            ;FPR2_4 <- exp.
        A ^=  FPR1_4
        FPR1_4 = A              ;FPR1_4.7 <- sign

;******* CALC.MANTISSA (set result to A·C·DE) **
                                ;d: FPR1 mantissa
                                ;s: FPR2 mantissa

        SET1 FPR1_3.7           ;(set mantissa MSB)
        SET1 FPR2_3.7           ;(set mantissa MSB)

        X = FPR1_X (A)
        A = FPR2_3
        MULU X                  ;d(0) * s(3)

        A <-> FPR1_1
        X = A
        E = A
        A = FPR2_2
        MULU X                  ;d(1) * s(2)
        A += FPR1_1             ;->CY

        A <-> E
        X = A
        A = FPR2_3
        MULU X                  ;d(1) * s(3)
        ADDC A,#0               ;<-CY
        A <-> X
        E += A                  ;->CY
        A = X
        ADDC A,#0               ;<-CY
```

```
A <-> FPR1_2
X = A
D = A
A = FPR2_1
MULU X                  ;d(2) * s(1)
E += A                  ;->CY

X = D (A)
A = FPR2_2
MULU X                  ;d(2) * s(2)
ADDC A,#0               ;<-CY
A <-> X
E += A                  ;->CY
A = X
ADDC A,FPR1_2           ;<-CY, ->CY

A <-> D
X = A
A = FPR2_3
MULU X                  ;d(2) * s(3)
ADDC A,#0               ;<-CY
A <-> X
D += A                  ;->CY
A = X
ADDC A,#0               ;<-CY

A <-> FPR1_3
X = A
C = A
A = FPR2_X
MULU X                  ;d(3) * s(0)
E += A                  ;->CY

X = C (A)
A = FPR2_1
MULU X                  ;d(3) * s(1)
ADDC A,#0               ;<-CY
A <-> X
E += A                  ;->CY
A = X
ADDC D,A                ;<-CY,->CY

X = C (A)
A = FPR2_2
MULU X                  ;d(3) * s(2)
ADDC A,#0               ;<-CY
A <-> X
D += A                  ;->CY
A = X
ADDC A,FPR1_3           ;<-CY,->CY
```

```
        A <-> C
        X = A
        A = FPR2_3
        MULU X                  ;d(3) * s(3)
        ADDC A,#0               ;<-CY
        A <-> X
        C += A                  ;->CY
        A = X
        ADDC A,#0               ;<-CY

;******* NORMALIZE **

        if_bit (A.7)
          FPR2_4++              ;2 <= mantissa < 4
          if_bit (Z)
            goto ERROR          ;exp. = 100H
          endif
        else
          if (FPR2_4 == #0)     ;1 <= mantissa < 2
            goto ZERO
          endif
          CLR1 CY
          A <-> E
          ROLC A,1
          A <-> D
          ROLC A,1
          A <-> C
          ROLC A,1
          A <-> E
          ROLC A,1
          A <-> D
          A <-> E
          A <-> C
          A <-> D
        endif

        FPR1_3 = A              ;1st mantissa
        A = FPR2_4

        goto T_STOR
```

```
;*****************************************************
;*
;*  78K0 FLOATING POINT DIVISION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 /= FPR2
;*             ERROR then set CY
;*
;*****************************************************

LDIV:

;******* ZERO EXCEPTION **

        CY = FPR2_3.7
        A = FPR2_4
        ADDC A,A
        if_bit (Z)
          goto ERROR
        endif
        B = A                   ;FPR2 exp.

        CY  = FPR1_3.7
        A = FPR1_4
        ADDC A,A                ;FPR1 exp.
        if_bit (Z)
          goto T_RET
        endif

;******* DIVIDE EXPONENT **

        A -= B

        if_bit (CY)
          A += #ZEROEX-1
          if_bit (!CY)          ;exp. <= 0
            goto ZERO
          endif

        else
          A += #ZEROEX-1
          if_bit (CY)           ;exp. > 100H
            goto ERROR
          endif
        endif
```

```
        A <-> FPR2_4            ;STORE:FPR2_4 <- (exp.-1)
        A ^=  FPR1_4
        FPR1_4 = A              ;FPR1_4.7 <- sign

;******* LOAD MANTISSA **
                                ;d: CY･E･HL <- FPR1 mantissa
                                ;s: FPR2_3･FPR2_LP
        B = #(SHORT-1)*BYTE+1   ;loop counter
        HL = FPR1_LP (AX)
        A = FPR1_3
        A |= #80H               ;(set mantissa MSB)
        E = A
        CLR1 CY

        FPR2_3 |= #80H          ;(set mantissa MSB)

;******* DIVIDE MANTISSA (set quotient to CY･X･C･D) **

        goto T_DIV1

        repeat
          A = L                 ;d * 2
          ADD L,A
          A = H
          ADDC H,A
          A = E
          ADDC E,A
T_DIV1:
          if_bit (!CY)
            if (E == FPR2_3) (A)
              if (H == FPR2_2) (A)
                A = L
                CMP A,FPR2_1
              endif
            endif
            NOT1 CY
          endif

          if_bit (CY)           ;if(d >= s)
            A = L               ;d -= s
            SUB A,FPR2_1
            L = A
            A = H
            SUBC A,FPR2_2
            H = A
            A = E
            SUBC A,FPR2_3
            E = A
            SET1 CY             ;quotient digit
          endif
```

```
          A = D                ;shift in quotient digit
          ADDC D,A
          A = C
          ADDC C,A
          A = X
          ADDC X,A
          B--
        until_bit(Z)

;******* NORMALIZE **

        if_bit (CY)            ;1 <= mantissa < 2
          FPR2_4++
          if_bit (Z)
            goto ERROR
          endif
          A = X
          RORC A,1
          X = A
          A = C
          RORC A,1
          C = A
          A = D
          RORC A,1
          D = A
        endif

        FPR1_3 = X (A)         ;1st mantissa
        E = #0                 ;4th mantissa
        A = FPR2_4

        goto T_STOR


        END
```

(7) LFLT2. SRC

```
$       TITLE   ('FLOATING POINT COMMON FUNCTIONS 1')
        NAME    M_LFLT2

#include "EQU. INC"
#include "REF1. INC"
#include "REF2. INC"

        EXTRN   LADDX, LMLTX

        PUBLIC  LPLY, LPLY2

        CSEG
;*****************************************************************
;*
;*  78K0 FLOATING POINT FUNCTION THAT
;*
;*     CALC. A POLYNOMIAL EXPRESSION by EXTENDED FORMAT
;*
;*                                  2                   n
;*  polynomial.1: x + k1xy + k1k2xy  + ... + k1k2..knxy
;*
;*     input  conditions:
;*        FPR1·FPR1_X <- x , FPR4·FPR4_X <- y
;*        HL <- head address of coefficient array (k1,,kn)
;*        B  <- n
;*
;*                                  2                   n
;*  polynomial.2: z + k1xy + k1k2xy  + ... + k1k2..knxy
;*
;*     input  conditions:
;*        FPR3·FPR3_X <- x , FPR4·FPR4_X <- y, FPR1·FPR1_X <- z
;*        HL <- head address of coefficient array (k1,,kn)
;*        B  <- n
;*
;*     output conditions(common to both):
;*        FPR1·FPR1_X <- result of polynomial expression
;*        FPR4·FPR4_X : keep
;*
;*****************************************************************

LPLY:
        CALL !LLD31X
        goto T_PLY1
```

```
LPLY2:
        repeat

          CALL !LXC13X
T_PLY1:
          CALL !LLD24X
          CALL !LMLTX

          CALL !LLD2CX
          CALL !LMLTX

          CALL !LXC13X

          CALL !LLD23X
          CALL !LADDX

          CY = FPR3_3.7
          A = FPR3_4
          ADDC A,A
          if_bit (Z)
            RET
          endif
          C = A

          CY = FPR1_3.7
          A = FPR1_4
          ROLC A,1

          A -= C
          if_bit (!CY)
            if (A >= #(SHORT-1)*BYTE+4)
              RET
            endif
          endif

          B--
        until_bit (Z)
        RET

        END
```

(8) LLD. SRC

```
$       TITLE   ('FPR LOAD FUNCTIONS')
        NAME    M_LLD

#include "EQU.INC"
#include "REF1.INC"

        PUBLIC  LLD21,LLD21X
        PUBLIC  LLD31,LLD31X
        PUBLIC  LLD41,LLD41X
        PUBLIC  LLD51,LLD51X
        PUBLIC  LLD32
        PUBLIC  LLD52
        PUBLIC  LLD13
        PUBLIC  LLD23,LLD23X
        PUBLIC  LLD24,LLD24X
        PUBLIC  LLD15
        PUBLIC  LLD25,LLD25X
        PUBLIC  LLD1C,LLD1CX
        PUBLIC  LLD2C,LLD2CX
        PUBLIC  LXC13,LXC13X
        PUBLIC  LXC14,LXC14X
        PUBLIC  LXC15,LXC15X

        CSEG
;*******************************************************
;*
;*  78K0 FLOATING POINT REGISTER LOAD FUNCTIONS
;*
;*
;*
;*******************************************************

;******* LOAD FPR2,FPR1

LLD21X:
        FPR2_X  = FPR1_X (A)
LLD21:
        FPR2_LP = FPR1_LP (AX)
        FPR2_HP = FPR1_HP (AX)
        RET
```

```
;******* LOAD FPR3,FPR1

LLD31X:
        FPR3_X  = FPR1_X (A)
LLD31:
        FPR3_LP = FPR1_LP (AX)
        FPR3_HP = FPR1_HP (AX)
        RET

;******* LOAD FPR4,FPR1

LLD41X:
        FPR4_X  = FPR1_X (A)
LLD41:
        FPR4_LP = FPR1_LP (AX)
        FPR4_HP = FPR1_HP (AX)
        RET

;******* LOAD FPR5,FPR1

LLD51X:
        FPR5_X  = FPR1_X (A)
LLD51:
        FPR5_LP = FPR1_LP (AX)
        FPR5_HP = FPR1_HP (AX)
        RET

;******* LOAD FPR3,FPR2

LLD32:
        FPR3_LP = FPR2_LP (AX)
        FPR3_HP = FPR2_HP (AX)
        RET

;******* LOAD FPR5,FPR2

LLD52:
        FPR5_LP = FPR2_LP (AX)
        FPR5_HP = FPR2_HP (AX)
        RET

;******* LOAD FPR1,FPR3

LLD13:
        FPR1_LP = FPR3_LP (AX)
        FPR1_HP = FPR3_HP (AX)
        RET
```

```
;******* LOAD FPR2,FPR3

LLD23X:
        FPR2_X  = FPR3_X (A)
LLD23:
        FPR2_LP = FPR3_LP (AX)
        FPR2_HP = FPR3_HP (AX)
        RET

;******* LOAD FPR2,FPR4

LLD24X:
        FPR2_X  = FPR4_X (A)
LLD24:
        FPR2_LP = FPR4_LP (AX)
        FPR2_HP = FPR4_HP (AX)
        RET

;******* LOAD FPR1,FPR5

LLD15:
        FPR1_LP = FPR5_LP (AX)
        FPR1_HP = FPR5_HP (AX)
        RET

;******* LOAD FPR2,FPR5

LLD25X:
        FPR2_X  = FPR5_X (A)
LLD25:
        FPR2_LP = FPR5_LP (AX)
        FPR2_HP = FPR5_HP (AX)
        RET

;******* LOAD FPR1,constant

LLD1CX:
        FPR1_X = [HL] (A)
        HL++
LLD1C:
        FPR1_1 = [HL] (A)
        HL++
        FPR1_2 = [HL] (A)
        HL++
        FPR1_3 = [HL] (A)
        HL++
        FPR1_4 = [HL] (A)
        HL++
        RET
```

```
;******* LOAD FPR2,constant

LLD2CX:
        FPR2_X = [HL] (A)
        HL++
LLD2C:
        FPR2_1 = [HL] (A)
        HL++
        FPR2_2 = [HL] (A)
        HL++
        FPR2_3 = [HL] (A)
        HL++
        FPR2_4 = [HL] (A)
        HL++
        RET

;******* XCHANGE FPR1,FPR3

LXC13X:
        A       =  FPR3_X
        A      <-> FPR1_X
        FPR3_X  =  A
LXC13:
        AX      =  FPR3_LP
        A      <-> FPR1_2
        A      <-> X
        A      <-> FPR1_1
        A      <-> X
        FPR3_LP =  AX
        AX      =  FPR3_HP
        A      <-> FPR1_4
        A      <-> X
        A      <-> FPR1_3
        A      <-> X
        FPR3_HP =  AX
        RET

;******* XCHANGE FPR1,FPR4

LXC14X:
        A       =  FPR4_X
        A      <-> FPR1_X
        FPR4_X  =  A
```

```
LXC14:
        AX      =  FPR4_LP
        A       <-> FPR1_2
        A       <-> X
        A       <-> FPR1_1
        A       <-> X
        FPR4_LP =  AX
        AX      =  FPR4_HP
        A       <-> FPR1_4
        A       <-> X
        A       <-> FPR1_3
        A       <-> X
        FPR4_HP =  AX
        RET

;******* XCHANGE FPR1,FPR5

LXC15X:
        A       =  FPR5_X
        A       <-> FPR1_X
        FPR5_X  =  A
LXC15:
        AX      =  FPR5_LP
        A       <-> FPR1_2
        A       <-> X
        A       <-> FPR1_1
        A       <-> X
        FPR5_LP =  AX
        AX      =  FPR5_HP
        A       <-> FPR1_4
        A       <-> X
        A       <-> FPR1_3
        A       <-> X
        FPR5_HP =  AX
        RET

        END
```

(9) LSIN. SRC

```
$       TITLE   ('SINE FUNCTION')
        NAME    M_LSIN

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LPLY
        EXTRN   LADDX,LMLTX

        EXTRN   FTOL,LTOF

        PUBLIC  LSIN
        PUBLIC  LMOD90,LSIN90

S_PLY   EQU     5

C1_X    EQU     0A2H
C1_1    EQU     0DAH
C1_2    EQU     00FH
C1_3    EQU     0C9H
C1_4    EQU     03FH

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT SINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- sin(x)
;*
;*********************************************************
LSIN:

;******* TRANS. sin(x) to (sign)sin(x' +nπ/2) : 0<=x'<π/2 **
                                              ; n = 0,1,2 or 3
        CALL !LMOD90
```

```
;******* TRANS. to (sign')sin(x'') by n : 0<=x''<π/2 **
LSIN90:
        if_bit (FPR4_3.0)       ;(n == odd)?
          SET1 FPR1_4.7
          HL = #C1
          CALL !LLD2CX
          CALL !LADDX   ;π/2 -x'
        endif

        CY = FPR4_3.1           ;(n/2 = odd)?
        CY ^= FPR4_4.7          ;  turn sign bit

        FPR1_4.7 = CY           ;set sign bit

;******* CALC. POLYNOMIAL EXPRESSION **

        CALL !LLD41X            ;set x'' to FPR4･FPR4_X

        CALL !LLD21X
        CALL !LMLTX             ;x''*x''

        CALL !LXC14X            ;set x''*x'' to FPR4･FPR4_X
                                ;set x'' to FPR1･FPR1_X
        HL = #CK
        B = #S_PLY
        CALL !LPLY

        A = #R_OK
        CLR1 CY
        RET

;***********************************************
;*       GET MOD by π/2
;*
;* input  conditions: FPR1 <- x
;* output conditions: FPR1･FPR1_X = x % π/2
;*                    FPR4_3.(0,1bit) <- quotient
;*                    FPR4_4.7   <- sign of x
;***********************************************
LMOD90:
        FPR1_X = #0             ;clear 4th mantissa
        FPR4_3 = #0
        FPR4_4 = FPR1_4 (A)     ;set sign bit
        CLR1 FPR1_4.7           ;|x|
```

```
        while (forever)
          if (FPR1_HP == #C1_4*100H+C1_3) (AX)
            if (FPR1_LP == #C1_2*100H+C1_1) (AX)
               A = FPR1_X
               CMP A,#C1_X
            endif
          endif

          if_bit (CY)
            RET
          endif

          CALL !LLD31X

          HL = #C2
          CALL !LLD2CX
          CALL !LMLTX           ;x / (π/2) : quotient
          FPR1_X  = #0
          FPR1_1  = #0
          FPR1_2 &= #0FCH       ;valid digit → 14bit

          CALL !LTOF
          if_bit (!CY)
            if_bit (!Z)         ;if (include decimal digit)
              CALL !FTOL        ;  cut decimal digit
            endif
            A = E
            A += FPR4_3
            FPR4_3 = A          ;add last 2bit of quotient
          endif

          HL = #C1
          CALL !LLD2CX
          CALL !LMLTX           ;int(x/(π/2)) * π/2

          SET1 FPR1_4.7
          CALL !LLD23X
          CALL !LADDX           ;x - int(x/(π/2))*π/2
        endw

C1:
        DB C1_X,C1_1,C1_2,C1_3,C1_4 ; const π/2
C2:
        DB 06EH,083H,0F9H,022H,03FH ;       2/π
```

```
CK:                                     ;coefficient array of LPLY
        DB 0AAH,0AAH,0AAH,02AH,0BEH ;const -1/6
        DB 0CCH,0CCH,0CCH,04CH,0BDH ;       -1/20
        DB 0C3H,030H,00CH,0C3H,0BCH ;       -1/42
        DB 0E3H,038H,08EH,063H,0BCH ;       -1/72
        DB 04FH,009H,0F2H,014H,0BCH ;       -1/110

        END
```

(10) **LCOS. SRC**

```
$       TITLE   ('COSINE FUNCTION')
        NAME    M_LCOS

#include "EQU.INC"
#include "REF1.INC"

        EXTRN   LMOD90,LSIN90

        PUBLIC  LCOS,LCOS90

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT COSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- cos(x)
;*
;*********************************************************
LCOS:

;******* TRANS. cos(x) to cos(x' +nπ/2) : 0<=x'<π/2 **
                                        ; n = 0,1,2 or 3
        CALL !LMOD90

;******* TRANS. to sin(x' +(n+1)π/2) **

LCOS90:
        CLR1 FPR4_4.7           ;clear sign bit
        FPR4_3++                ;n++

        goto LSIN90

        END
```

(11) **LTAN. SRC**

```
$       TITLE   ('TANGENT FUNCTION')
        NAME    M_LTAN

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LDIV
        EXTRN   LMOD90,LSIN90,LCOS90

        PUBLIC  LTAN

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT TANGENT FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- tan(x)
;*                        ERROR then set CY
;*
;*********************************************************
LTAN:

;******* TRANS. sin(x) to (sign)sin(x' +nπ/2) : 0<=x'<π/2 **
;              cos(x) to       cos(x' +nπ/2) : n = 0,1,2 or 3

        CALL !LMOD90

        AX = FPR4_HP
        PUSH AX                 ;esc. n & sign
        CALL !LLD51X            ;esc. x'

;******* GET (sign)sin(x' +nπ/2) / cos(x' +nπ/2) **

        CALL !LCOS90
        CALL !LXC15X

        POP AX
        FPR4_HP = AX
        CALL !LSIN90

        CALL !LLD25
        goto LDIV

        END
```

(12) **LLOG. SRC**

```
$       TITLE   ('LOGARITHMIC FUNCTION')
        NAME    M_LLOG

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LADD,LSUB,LDIV
        EXTRN   LADDX,LMLTX
        EXTRN   LPLY2
        EXTRN   FTOL

        PUBLIC  LLOG

C0_3    EQU     0B5H    ;1st mantissa of √2

S_PLY   EQU     4

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT LOGARITHMIC FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- log(x)
;*                        ERROR then set CY
;*
;*********************************************************
LLOG:
        CY = FPR1_3.7
        A = FPR1_4
        ADDC A,A                ;x exponent(xe + exponent bias)

;******* EXCEPTION **

        if_bit (CY || Z)
          A = #R_ERR            ;zero,negative exception
          SET1 CY
          RET
        endif
```

```
;******* CALC. EXP.PART LOG **

        X = A
        A = #0
        AX -= #ZEROEX
        DE  = AX                ;exponent value (:xe)

        FPR3_LP = FPR1_LP (AX)  ;set mantissa value (:xf) to FPR3
        AX      = FPR1_HP

        A  = #ZEROEX/2
        A <-> X
        A |= #80H
        if (A >= #C0_3)
          A &= #7FH             ;xf/2 (:xf')
          DE++                  ;xe+1 (:xe')
        endif
        A <-> X
        FPR3_HP = AX

        CALL !FTOL              ;real value of xe'

        HL = #C1
        CALL !LLD2CX
        CALL !LMLTX             ;exponent part log (xe'*log2)
        CALL !LLD41X            ;STORE xe'*log2 to FPR4･FPR4_X

;******* TRANS. MANTISSA FOR TAYLOR APPROXIMATE **

        CALL !LLD13
        HL = #C2
        CALL !LLD2C
        CALL !LADD              ;xf' +1

        CALL !LXC13
        HL = #C2
        CALL !LLD2C
        CALL !LSUB              ;xf' -1

        CALL !LLD23
        CALL !LDIV              ;(xf' -1)/(xf' +1)  :x'
```

```
;******* CALC. MANTISSA LOG **

        CALL !LLD31X

        A = FPR3_4
        ADD A,A
        if_bit (!Z)             ;if(x' != 0) set 2x' to FPR3･FPR3_X
          ADD FPR3_3,#80H       ;else        set 0
          ADDC FPR3_4,#0
        endif

        CALL !LLD21X
        CALL !LMLTX             ;x'*x'
        CALL !LXC14X            ;set x'*x' to FPR4･FPR4_X

        CALL !LLD23X
        CALL !LADDX             ;set xe'*log2+2x' to FPR1･FPR1_X

        HL = #CK
        B  = #S_PLY
        CALL !LPLY2

        A = #R_OK
        CLR1 CY
        RET
C1:
        DB 0F7H,017H,072H,031H,03FH ; const log2
C2:
        DB      000H,000H,080H,03FH ;       1

CK:                                 ; coefficient array of LPLY
        DB 0AAH,0AAH,0AAH,0AAH,03EH ; const 1/3
        DB 099H,099H,099H,019H,03FH ;       3/5
        DB 0B6H,06DH,0DBH,036H,03FH ;       5/7
        DB 0C7H,071H,01CH,047H,03FH ;       7/9

        END
```

(13) **LLOG10. SRC**

```
$       TITLE   ('LOGARITHMIC FUNCTION 2')
        NAME    M_LLOG10

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LMLTX
        EXTRN   LLOG

        PUBLIC  LLOG10

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT LOGARITHMIC FUNCTION 2
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- log10(x)
;*                        ERROR then set CY
;*
;*************************************************************
LLOG10:

;******* log(x) / log10 **

        CALL !LLOG
        if_bit (CY)
          RET
        endif

        HL = #C1
        CALL !LLD2CX
        goto LMLTX

C1:
        DB 0A9H,0D8H,05BH,0DEH,03EH ; const 1/log10

        END
```

(14) **LEXP. SRC**

```
$       TITLE   ('EXPONENTIAL FUNCTION')
        NAME    M_LEXP

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LADDX,LSUBX,LMLTX
        EXTRN   LPLY
        EXTRN   LTOF,FTOL

        PUBLIC  LEXP,LEXPX

S_PLY   EQU     6

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT EXPONENTIAL FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- e^x
;*                        ERROR then set CY
;*
;*********************************************************
LEXP:
        FPR1_X = #0
LEXPX:
        FPR3_X = FPR1_4 (A)     ;esc sign bit

;******* TRANS. to 2^(x/log2) **

        HL = #C1
        CALL !LLD2CX
        CALL !LMLTX             ;1/log2 * x
```

```
;******* CALC. EXP **

        if_bit (!CY)
          CALL !LTOF            ;trunk(x/log2)
        endif
        if_bit (CY)
          goto T_FLOW
        endif

        if_bit (Z)
          CMP FPR1_X,#0
        endif
        if_bit (!Z)                     ;if ((dec(x/log2) != 0) &&
          if (FPR1_HP >= #8080H) (AX)   ;    (negative))
            DE--                        ;  floor(x/log2) = trunk(x/log2)-1
          endif
        endif

        AX = DE
        AX += #ZEROEX

        if (A != #0)
          goto T_FLOW
        endif

        FPR5_X = X (A)          ;esc exp.
;******* CALC. MANTISSA **

        CALL !LLD21X
        CALL !FTOL              ;floor(x/log2)
        FPR1_4 ^= #80H
        CALL !LADDX             ;x' : x/log2 -floor(x/log2)

        if (FPR1_4 == #3FH)     ;(1/2<=x'<1)
          HL = #C2+1
          CALL !LLD2C
          FPR2_X = #02H         ;(power adjust to (x'<1) for boundary)
          CALL !LSUBX           ;x' :  decimal(x/log2)-1
        endif

        CALL !LLD41X            ;set x'

        HL = #CK
        CALL !LLD2CX
        CALL !LMLTX             ;set log2*x'
```

```
        B = #S_PLY
        CALL !LPLY

        HL = #C2
        CALL !LLD2CX
        CALL !LADDX             ;2^(x')

;******* RETURN EXP.PART **

        A = FPR5_X
        RORC A,1
        FPR1_3.7 = CY
        FPR1_4 = A
T_EXP9:
        A = #R_OK
        CLR1 CY
        RET
T_FLOW:
        if_bit (FPR3_X.7)
          FPR1_HP = #0
          goto T_EXP9
        endif

        A = #R_ERR
        SET1 CY
        RET

C1:
        DB 029H,03BH,0AAH,0B8H,03FH  ; const  1/log2
C2:
        DB 000H,000H,000H,080H,03FH  ;        1

CK:                                  ; coefficient array of LPLY2
        DB 0F7H,017H,072H,031H,03FH  ; const.log2
        DB 0F7H,017H,072H,0B1H,03EH  ;       log2/2
        DB 0F5H,01FH,098H,06CH,03EH  ;       log2/3
        DB 0F7H,017H,072H,031H,03EH  ;       log2/4
        DB 0F9H,0DFH,0F4H,00DH,03EH  ;       log2/5
        DB 0F5H,01FH,098H,0ECH,03DH  ;       log2/6
        DB 01BH,089H,0CBH,0CAH,03DH  ;       log2/7

        END
```

(15) **LEXP10. SRC**

```
$       TITLE   ('EXPONENTIAL FUNCTION 2')
        NAME    M_LEXP10

#include "EQU. INC"
#include "REF1. INC"
#include "REF2. INC"

        EXTRN   LMLTX
        EXTRN   LEXPX

        PUBLIC  LEXP10

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT EXPONENTIAL FUNCTION 2
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- 10^x
;*                        ERROR then set CY
;*
;*********************************************************
LEXP10:
        FPR1_X = #0             ;clear 4th mantissa

;******* TRANSLATE TO e^(log10*x) **

        FPR3_X = FPR1_4 (A)

        HL = #C1
        CALL !LLD2CX
        CALL !LMLTX
        if_bit (CY)             ;overflow
          if_bit (FPR3_X.7)     ;x<0
            FPR1_HP = #0
            A = #R_OK
            CLR1 CY
          endif
          RET
        endif

        goto LEXPX

C1:
        DB 0DDH,08DH,05DH,013H,040H ;const log10

        END
```

(16) **LPOW. SRC**

```
$       TITLE   ('POWER FUNCTION')
        NAME    M_LPOW

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LMLTX
        EXTRN   LLOG,LEXPX

        PUBLIC  LPOW

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT POWER FUNCTION
;*
;*     input  condition : FPR1 <- a , FPR2 <- b
;*
;*     output conditions: FPR1 <- a^b
;*                        ERROR then set CY
;*
;*************************************************************
LPOW:
        CY = FPR2_3.7
        A = FPR2_4
        ROLC A,1                ;exp. of b
        C = A

        CY = FPR1_3.7
        A  = FPR1_4
        ADDC A,A                ;exp. of a
        A <-> C

;******* a=0 EXCEPTION **

        if_bit (Z)
          CMP A,#0
          if_bit (Z || FPR2_4.7)
            goto ERROR          ;0^0,0^(negative)=overflow
          endif
          goto T_POW9           ;0^(positive)=0
        endif
```

```
;******* a<0 EXCEPTION **

        CLR1 FPR5_X.0           ;FPR5_X.0 :sign of result
        if_bit (CY)
          if (A == #0)
            HL = #C1
            CALL !LLD1C
            goto T_POW9         ;x^0=1
          endif

;**   ** b: DECIMAL PART = 0 ? **

            if (A < #ZEROEX)
              goto ERROR        ;(negative)^(decimal)=error
            endif

            A -= #ZEROEX+BYTE*(SHORT-1)-1
            C = A
            B = FPR2_1 (A)
            X = FPR2_2 (A)
            A = FPR2_3
            SET1 A.7

            if_bit (CY)
              repeat
                RORC A,1
                A <-> X
                RORC A,1
                A <-> B
                RORC A,1
                A <-> B
                A <-> X
                if_bit (CY)
                  goto ERROR    ;include decimal digit
                endif
                C++
              until_bit (Z)
            endif
            if_bit (Z)          ;if (exp.of b <= 23)
              FPR5_X = B (A)    ;FPR5_X.0 = UNIT1
            endif
          endif
```

```
;***** CALC. e^(b * log|a|) **

        CALL !LLD52             ;esc. b to FPR5
        CLR1 FPR1_4.7
        CALL !LLOG              ;log|a|
        CALL !LLD25             ;ret. b to FPR2
        FPR2_X = #0
        CALL !LMLTX             ;b * log|a|

        if_bit (CY)             ;overflow
          if_bit (FPR5_4.7)
            FPR1_HP = #0
            goto T_POW9
          endif
          goto ERROR
        endif

        FPR5_1 = FPR5_X (A)
        CALL !LEXPX

;******* RETURN SIGN BIT **

        if_bit (CY)
          RET
        endif

        if_bit (FPR5_1.0)       ;if (b == odd integer && a<0)
          SET1 FPR1_4.7         ;  {set sign bit}
        endif
T_POW9:
        A = #R_OK
        CLR1 CY
        RET
ERROR:
        A = #R_ERR
        SET1 CY
        RET
C1:
        DB 000H,000H,080H,03FH ;const 1

        END
```

(17) **LSQRT. SRC**

```
$       TITLE   ('SQUARE ROOT FUNCTION')
        NAME    M_LSQRT

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LADD,LDIV

        PUBLIC  LSQRT

C_LIM   EQU     5       ;limiter of approximate

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT SQUARE ROOT FUNCTION
;*
;*   input  condition : FPR1 <- x
;*
;*   output conditions: FPR1 <- √(x)
;*                      ERROR then set CY
;*
;*************************************************************
LSQRT:
        CY = FPR1_3.7
        A = FPR1_4
        ADDC A,A

;******* EXCEPTION **

        if_bit (Z)
          goto T_QRT9   ;zero
        endif
        if_bit (CY)
          A = #R_ERR    ;negative
          RET
        endif
```

```
;******* TRANS. √(a) TO √(r)*2^n (1≦r<4) **

        RORC A,1
        if_bit (CY)
          FPR1_4 = #(ZEROEX-1)/2        ;r'/2 (r'=r)
        else
          FPR1_4 = #(ZEROEX-2)/2        ;r'/2 (r'=r/4)
        endif
        ADDC A,#ZEROEX/2
        FPR1_3 ^= #80H
        FPR4_X = A              ;escape exp.part root (n)

;******* CALC. VIRT.PART ROOT **

        CALL !LLD31     ;esc. r'/2 to FPR3

        HL = #C1
        CALL !LLD2C
        CALL !LADD      ; r'/2 + .5  : 2ndary approximate (R2)

        FPR3_X = #C_LIM-2
        repeat
          CALL !LLD41   ; esc.previous approximate(Ri) to FPR4
          CALL !LLD21
          CALL !LLD13
          CALL !LDIV    ; (r'/2) / Ri

          CALL !LLD24
          SUB FPR2_3,#80H       ; Ri/2
          SUBC FPR2_4,#0

          CALL !LADD    ; Ri/2 + r'/(2Ri) : next approximate

          FPR3_X--
        until_bit (Z)

        A = FPR4_X
        RORC A,1
        FPR1_4 = A              ;ret. exp.part root (n)
        FPR1_3.7 = CY
        FPR1_X = #0
T_QRT9:
        A = #R_OK
        CLR1 CY
        RET
C1:
        DB 000H,000H,000H,03FH ;const .5

        END
```

(18) **LASIN. SRC**

```
$       TITLE   ('ARCSINE FUNCTION')
        NAME    M_LASIN

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LMLT,LDIV
        EXTRN   LADDX
        EXTRN   LSQRT,LATAN

        PUBLIC  LASIN

C1_4    EQU     03FH
C1_3    EQU     080H
C1_2    EQU     000H
C1_1    EQU     000H

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT ARCSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- arcsin(x)
;*                        ERROR then set CY
;*
;*********************************************************
LASIN:

;******** EXCEPTION **

        CALL !LLD51             ;store x to FPR5

        CLR1 FPR1_4.7           ;x <- |x|

        if (FPR1_HP == #C1_4*100H+C1_3) (AX)
          if (FPR1_LP == #C1_2*100H+C1_1) (AX)
            HL = #C2            ;|x|=1 exception
            CALL !LLD1CX
```

```
            if_bit (FPR5_4.7)
              SET1 FPR1_4.7      ;± π /2
            endif
            A = #R_OK
            RET
          endif
        endif

        if_bit (!CY)             ;|x|>1 exception
          A = #R_ERR
          SET1 CY
          RET
        endif

;******** TRANS. to ARCTAN  **

        CALL !LLD21
        CALL !LMLT               ;x*x
        SET1 FPR1_4.7

        HL = #C1
        CALL !LLD2CX
        CALL !LADDX              ;1-x*x
        CALL !LSQRT              ;√(1-x*x)

        CALL !LLD21
        CALL !LLD15
        CALL !LDIV               ;x/√(1-x*x)

        goto LATAN

C1:
        DB 000H,C1_1,C1_2,C1_3,C1_4 ;const 1
C2:
        DB 0A2H,0DAH,00FH,0C9H,03FH ;       π /2

        END
```

(19) **LACOS. SRC**

```
$       TITLE   ('ARCCOSINE FUNCTION')
        NAME    M_LACOS

#include "EQU. INC"
#include "REF1. INC"
#include "REF2. INC"

        EXTRN   LADDX
        EXTRN   LASIN

        PUBLIC  LACOS

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT ARCCOSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- arccos(x)
;*                        ERROR then set CY
;*
;*************************************************************
LACOS:

        CALL !LASIN
        if_bit (CY)
          RET
        endif

        FPR1_4 ^= #80H

        HL = #C1
        CALL !LLD2CX
        goto LADDX      ;π/2 -arcsin(x)

C1:
        DB 0A2H, 0DAH, 00FH, 0C9H, 03FH ;const π/2

        END
```

(20) **LATAN. SRC**

```
$       TITLE   ('ARCTANGENT FUNCTION')
        NAME    M_LATAN

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LADD,LSUB,LMLT,LDIV
        EXTRN   LADDX,LMLTX
        EXTRN   LPLY2
        EXTRN   LRCPN

        PUBLIC  LATAN

S_PLY   EQU     3

        CSEG
;************************************************************
;*
;*  78K0 FLOATING POINT ARCTANGENT FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- arctan(x)
;*
;************************************************************
LATAN:
        AX = FPR1_HP
        FPR5_X = A      ;esc. sign bit

;******* TRANS. arctan(x) to (sign)(arctan(x'))   or
;                           (sign)(π/2 -arctan(x')) **

                        ;x' = |x|       case |x|< 1
                        ;x' = 1/|x|     case |x|>=1

        CLR1 FPR1_4.7   ;|x|

        CLR1 A.7
        CMPW AX,#ZEROEX SHL 7

        FPR5_X.6 = CY   ;(|x|<1)

        if_bit (!CY)
          CALL !LRCPN   ;1/|x|
        endif
```

```
;******* TRANS. arctan(x') to arctan(W)+arctan(V) case if x'>=1/8 **

                    ;W : 1/8,3/8,5/8 or 7/8
                    ;   (-1/8 <= x'-W <= 1/8)
                    ;V = (x'-W)/(1+x'*W)

        FPR3_X = #0
        if (FPR1_4 >= #3EH)     ;1/8

          CALL !LLD41
          AX = FPR1_HP
          if (AX >= #3F40H)                             ;6/8
                FPR2_HP = #060H+(SHORT+1)*400H          ;W=7/8
          elseif (A >= #3FH)                            ;4/8
                FPR2_HP = #020H+(SHORT+1)*300H          ;W=5/8
          elseif (FPR1_3 >= #80H)                       ;2/8
                FPR2_HP = #0C0H+(SHORT+1)*200H          ;W=3/8
          else
                FPR2_HP = #000H+(SHORT+1)*100H          ;W=1/8
          endif

          A <-> FPR2_4
          FPR3_X = A    ;store data pointer of arctan(W)
          FPR2_LP = #0

          CALL !LLD32

          CALL !LMLT    ;x'*W

          HL = #C1
          CALL !LLD2C
          CALL !LADD    ;x'*W +1

          CALL !LXC14
          CALL !LLD23
          CALL !LSUB    ;x' -W

          CALL !LLD24
          CALL !LDIV    ;(x'-W)/(1+x'*W)
        endif
```

```
;******* CALC. APPROXIMATE POLINOMIAL FUNCTION **

        CALL !LLD41

        CALL !LLD21
        CALL !LMLT      ;V*V

        CALL !LXC14X    ;set V*V to FPR4･FPR4_X
        FPR1_X = #0

        HL = #CK0
        CALL !LLD2CX
        CALL !LMLTX     ;4a0*V

        X = FPR3_X (A)
        A = #0
        AX += #CW
        HL = AX
        CALL !LLD2CX    ;arctan(W)

        CALL !LLD31X    ;set 4a0*V to FPR3･FPR3_X

        CALL !LADDX     ;4a0*V +arctan(W)

        HL = #CK1
        B  = #S_PLY
        CALL !LPLY2     ;arctan(x')

        if_bit (!FPR5_X.6)      ;|x|>=1?
          FPR1_4 ^= #80H
          HL = #C2
          CALL !LLD2CX
          CALL !LADDX   ;π/2 -arctan(x')
        endif

;******* RETURN SIGN BIT **

        FPR1_4.7 = FPR5_X.7 (CY)

        A = #R_OK
        CLR1 CY
        RET
C1:
        DB      000H,000H,080H,03FH ;const 1
C2:
        DB 0A2H,0DAH,00FH,0C9H,03FH ;       π/2
```

```
CW:
        DB 000H,000H,000H,000H,000H ;       0
        DB 0D5H,0D4H,0ADH,0FEH,03DH ;       arctan(1/8)
        DB 00FH,0CAH,0B0H,0B7H,03EH ;       arctan(3/8)
        DB 05FH,05DH,000H,00FH,03FH ;       arctan(5/8)
        DB 02CH,03EH,005H,038H,03FH ;       arctan(7/8)

                                    ;coefficient array(an)
                                    ;     of approximate
CK0:
        DB 0FEH,0FFH,0FFH,07FH,03FH ;cof.  a0     * 4
CK1:    DB 032H,0A4H,0AAH,0AAH,0BEH ;      a1/a0 *16
        DB 05CH,0DAH,090H,019H,0BFH ;      a2/a1 *16
        DB 001H,058H,0FEH,031H,0BFH ;      a3/a2 *16

        END
```

(21) **LHSIN. SRC**

```
$       TITLE   ('HYPERBOLICSINE FUNCTION')
        NAME    M_LHSIN

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LMLT,LSUB
        EXTRN   LPLY
        EXTRN   LEXP
        EXTRN   LRCPN

        PUBLIC  LHSIN

S_PLY   EQU     3

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT HYPERBOLICSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- sinh(x)
;*                        ERROR then set CY
;*
;*************************************************************
LHSIN:

        FPR5_1 = FPR1_4 (A)

;******* CALC. (e^|x|-e^(-|x|))/2  case if |x| >= 0.5 **

        A &= #7FH
        if (A >= #ZEROEX/2)     ;|x| >= 0.5

          CLR1 FPR1_4.7

          CALL !LEXP            ;e^|x|
          if_bit (CY)
            RET                 ;overflow
          endif
```

```
          CALL !LLD31
          CALL !LRCPN           ;e^(-|x|)
          CALL !LLD23

          CALL !LSUB            ;e^(-|x|)-e^|x|
          SUB FPR1_3,#80H
          SUBC FPR1_4,#0        ;(e^(-|x|)-e^|x|)/2

          FPR1_4.7 = FPR5_1.7 (CY)      ;sign bit

;******** CALC. DIRECT APPROXIMATE  case if |x| <  0.5 **

        else                    ;|x| < 0.5

          CALL !LLD41

          CALL !LLD21
          CALL !LMLT            ;x*x

          CALL !LXC14X
          FPR1_X = #0

          HL = #CK
          B  = #S_PLY
          CALL !LPLY
        endif

        A = #R_OK
        CLR1 CY
        RET
CK:                                 ; coefficient array of LPLY
        DB 0AAH,0AAH,0AAH,02AH,03EH ; const 1/6
        DB 0CCH,0CCH,0CCH,04CH,03DH ;       1/20
        DB 0C3H,030H,00CH,0C3H,03CH ;       1/42

        END
```

(22) **LHCOS. SRC**

```
$       TITLE   ('HYPERBOLICCOSINE FUNCTION')
        NAME    M_LHCOS

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LADD
        EXTRN   LRCPN
        EXTRN   LEXP

        PUBLIC  LHCOS

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT HYPERBOLICCOSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- cosh(x)
;*                        ERROR then set CY
;*
;*************************************************************
LHCOS:

        CLR1 FPR1_4.7

;******** CALC. (e^|X|+e^(-|X|))/2 **

        CALL !LEXP              ;e^|x|
        if_bit (CY)
          RET                   ;overflow
        endif

        CALL !LLD31
        CALL !LRCPN             ;e^(-|x|)
        CALL !LLD23

        CALL !LADD              ;e^|x| +e^(-|x|)
        SUB FPR1_3,#80H
        SUBC FPR1_4,#0          ;(e^|x| +e^(-|x|))/2
        RET

        END
```

(23) **LHTAN. SRC**

```
$       TITLE   ('HYPERBOLICTANGENT FUNCTION')
        NAME    M_LHTAN

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LDIV
        EXTRN   LHSIN,LHCOS

        PUBLIC  LHTAN

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT HYPERBOLICTANGENT FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- tanh(x)
;*
;*********************************************************
LHTAN:

        CALL !LLD51             ;esc. x to FPR5

;******* CALC. LHCOS **

        CALL !LHCOS
        if_bit (CY)
          HL = #C1
          CALL !LLD1C           ;tanh(positive infinity)=1
          if_bit (FPR5_4.7)
            SET1 FPR1_4.7       ;tanh(negative infinity)=-1
          endif
          A = #R_OK
          CLR1 CY
          RET
        endif

        CALL !LXC15             ;esc. cosh(x)
        A = FPR5_1
        PUSH AX
```

```
;******* CALC. LHSIN **

        CALL !LHSIN             ;sinh(x)

;******* CALC. LHTAN **

        POP AX
        FPR5_1 = A
        CALL !LLD25             ;ret. cosh(x)
        goto  LDIV              ;sinh(x)/cosh(x)
C1:
        DB 000H,000H,080H,03FH ; const 1

        END
```

(24) **LABS. SRC**

```
$       TITLE   ('ABSOLUTE FUNCTION')
        NAME    M_LABS

#include "EQU.INC"
#include "REF1.INC"

        PUBLIC  LABS

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT ABSOLUTE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <-  |x|
;*
;*************************************************************
LABS:
        CLR1 FPR1_4.7

        A = #R_OK
        CLR1 CY
        RET

        END
```

(25) **LRCPN. SRC**

```
$       TITLE   ('RECIPROCAL NUMBER FUNCTION')
        NAME    M_LRCPN

#include "EQU. INC"
#include "REF1. INC"
#include "REF2. INC"

        EXTRN   LDIV

        PUBLIC  LRCPN

        CSEG
;*************************************************
;*
;*  78K0 FLOATING POINT FUNCTION
;*                      GET RECIPROCAL NUMBER
;*
;*    input condition  : FPR1 <- x
;*
;*    output conditions: FPR1 <- 1/x
;*              ERROR then set CY
;*
;*************************************************
LRCPN:
        CALL !LLD21

        HL = #C1
        CALL !LLD1C

        goto LDIV


C1:
        DB 000H, 000H, 080H, 03FH

        END
```

(26) **POTORA. SRC**

```
$       TITLE   ('TRANS. TO RIGHT ANGLE COORDINATES')
        NAME    M_POTORA

#include "EQU. INC"
#include "REF1. INC"
#include "REF2. INC"

        EXTRN   LMLTX
        EXTRN   LMOD90, LSIN90, LCOS90

        PUBLIC  POTORA

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT FUNCTION THAT
;*         TRANS. COORDINATES FROM POLE TO RIGHT ANGLE
;*
;*     input  condition : FPR1 <- r, FPR2 <- θ
;*
;*     output conditions: FPR1 <- x, FPR2 <- y
;*                        ERROR then set CY
;*
;*********************************************************
POTORA:
        A = FPR1_4
        CY = FPR1_3.7
        ADDC A, A

;******* EXCEPTION **

        if_bit (Z)              ;r == 0
          FPR2_HP = #0
          goto T_ORA9
        endif

        if_bit (CY)             ;r < 0
          A = #R_ERR
          RET
        endif
```

```
;******* TRANSLATE  **

        AX = FPR1_LP
        PUSH AX
        AX = FPR1_HP
        PUSH AX                 ;esc. r

        FPR1_HP = FPR2_HP (AX)
        FPR1_LP = FPR2_LP (AX)

        CALL !LMOD90            ;θ -> (sign)(θ'+ nπ/2) (0≦θ'<π/2)

        AX = FPR4_HP
        PUSH AX                 ;esc. n & sign
        CALL !LLD51X            ;esc. θ'

        CALL !LCOS90            ;cos(θ'+nπ/2)
        CALL !LXC15X

        POP AX
        FPR4_HP = AX            ;ret. n & sign
        CALL !LSIN90            ;sin((sign)(θ'+nπ/2))

        POP AX
        FPR3_HP = AX
        POP AX
        FPR3_LP = AX
        FPR3_X = #0             ;ret. r

        CALL !LLD23X
        CALL !LMLTX             ;rsinθ

        CALL !LXC15X

        CALL !LLD23X
        CALL !LMLTX             ;rcosθ

        CALL !LLD25X

T_ORA9:
        A = #R_OK
        CLR1 CY
        RET

        END
```

(27) **RATOPO. SRC**

```
$       TITLE   ('TRANS. TO POLE COORDINATES')
        NAME    M_RATOPO

#include "EQU.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   LADDX,LMLT,LDIV
        EXTRN   LSQRT,LATAN

        PUBLIC  RATOPO

        CSEG
;*************************************************************
;*
;*  78K0 FLOATING POINT FUNCTION THAT
;*         TRANS. COORDINATES FROM RIGHT ANGLE TO POLE
;*
;*     input  condition : FPR1 <- x, FPR2 <- y
;*
;*     output conditions: FPR1 <- r, FPR2 <- θ
;*                        ERROR then set CY
;*
;*************************************************************
RATOPO:
        A = FPR2_4
        L = A                   ;L.7 <- sign of y
        CY = FPR2_3.7
        ADDC A,A

        A = FPR1_4
        H = A                   ;H.7 <- sign of x
        CY = FPR1_3.7
        ROLC A,1

;******** EXCEPTION **

        if_bit (Z)
          if (A == #0)          ;if (x==0 && y==0)
            goto T_OPO9         ;  {(r,θ) = (0,0)}
          endif
          L = #0                ;if (y==0) clear sign bit of y
        endif
```

```
;******** CALC. x*x+y*y  **

        CALL !LLD41             ;FPR4 <- x
        CALL !LLD52             ;FPR5 <- y

        CALL !LLD21
        CALL !LMLT              ;x*x
        if_bit (CY)
          RET
        endif

        CALL !LLD31X

        CALL !LLD15
        CALL !LLD21
        CALL !LMLT              ;y*y
        if_bit (CY)
          RET
        endif

        CALL !LLD23X
        CALL !LADDX             ;x*x + y*y
        if_bit (CY)
          RET
        endif

        PUSH HL                 ;sign bit of x,y

;******** CALC. y/x  **

        CALL !LXC15
        CALL !LLD24
        CALL !LDIV
        if_bit (CY)
          HL = #C1
          CALL !LLD1C           ;arctan(infinity)=π/2 or -π/2
          POP AX
          A = X
          ROLC A,1
          FPR1_4.7 = CY         ;sign of θ <- sign of y
          goto T_OP08
        endif
```

```
;******** CALC. θ **

        CALL !LATAN
        POP AX
        if_bit (A.7)
          HL = #C2
          CALL !LLD2CX          ;x＜0, y≧0 : θ=arctan(y/x)+π
          A = X
          ROLC A,1
          FPR2_4.7 = CY         ;x＜0, y＜0 : θ=arctan(y/x)-π
          CALL !LADDX
        endif

;******** CALC. r  **

T_OP08:
        CALL !LXC15
        CALL !LSQRT             ;√(x*x+y*y)

        CALL !LLD25             ;θ

T_OP09:
        A = #R_OK
        CLR1 CY
        RET
C1:
        DB      0DAH,00FH,0C9H,03FH ;const π/2
C2:
        DB 0A2H,0DAH,00FH,049H,040H ;       π

        END
```

(28) **ATOL. SRC**

```
$       TITLE   ('TRANSLATE ASCII STRING')
        NAME    M_ATOL

#include "EQU.INC"
#include "ASCII.INC"
#include "REF1.INC"
#include "REF2.INC"

        EXTRN   FTOL,LTOF
        EXTRN   LMLT
        EXTRN   LADDX,LMLTX
        EXTRN   LNOR
        EXTRN   LEXPX

        PUBLIC  ATOL

S_VIRT  EQU     27      ;maximum length of mantissa

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT FUNCTION THAT
;*            TRANSLATE ASCII STRING
;*
;*     input  condition : HL <- HEAD ADDRESS of STRING
;*
;*     output conditions: FPR1 <- (REAL VALUE MEANING STRING)
;*                        ERROR then set CY
;*                        HL keep
;*********************************************************
ATOL:
        PUSH HL

;******* TRANS. SIGN **

        CLR1 FPR1_4.7           ;sign keeper

        CALL !GETC
        if     (A == #N_PL)
          CALL !GETC
        elseif (A == #N_MN)
          SET1 FPR1_4.7
          CALL !GETC
        endif
```

```
;******* TRANS. MANTISSA TO BINARY **

        FPR2_LP = #0    ;work FPR2_1 : decimal digit counter (F)
                        ;     FPR2_2 : neglect digit counter (N)
        FPR2_HP = #80H  ;     FPR2_3 : mantissa digit counter
                        ;     FPR2_4.0 : decimal digit flag
                        ;     FPR2_4.1 : neglect digit flag
        while (forever)
          if (A < #N_9+1)
            if_bit (FPR2_3.7)
              E = A             ;initial digit
              D = #0
              BC = #0
              FPR2_3 = #S_VIRT-1+1

            else

              FPR2_3--
              if_bit (Z)
                goto ERROR      ;mantissa length over
              endif

              if_bit (!FPR2_4.1)        ;(not neglect) ?
                A <-> E
                X = #10
                MULU X

                A <-> X
                E += A
                A = X
                A <-> D
                X = #10
                MULU X

                A <-> X
                ADDC D,A
                A = X
                A <-> C
                X = #10
                MULU X

                A <-> X
                ADDC C,A
                A = X
                ADDC A,#0
                B = A           ;BC･DE <- mantissa val.
```

```
            if_bit (!Z)      ;if (work area fill)
              SET1 FPR2_4.1 ;  (neglect forword digit)
            endif
          else
            FPR2_2++         ;neglect digit count up
          endif
        endif

        if_bit (FPR2_4.0)
          FPR2_1++           ;decimal digit count up
        endif

      elseif (A == #N_PD)
        if_bit (!FPR2_4.0)
          SET1 FPR2_4.0      ;begin count of decimal digit
        else
          goto ERROR         ;duplicate
        endif
      else
        break
      endif

      PUSH BC
      PUSH DE
      CALL !GETC
      POP DE
      POP BC
    endw

;******* EXCEPTION **

    FPR2_X = A

    if_bit (FPR2_3.7)        ;no mantissa digit
      goto ERROR
    endif

    if (BC == #0) (AX)
      if (DE == #0) (AX)
        FPR1_HP = #0         ;ZERO EXCEPTION
        goto T_TOL9
      endif
    endif

    if (FPR2_X >= #N_MN)
      goto ERROR
    endif
```

```
        A = FPR2_1      ;decimal digit - neglect digit (F-N)
        A -= FPR2_2
        FPR2_2 = A

;******* NORMALIZE MANTISSA val. **

        A = B

        FPR2_1 = #ZEROEX+SHORT*BYTE-1
        CALL !LNOR              ;normalize with sign bit
        CALL !LLD51X            ;esc. mantissa val(A')

        A = FPR2_X

;******* TRANS. EXP_PART **

        X = #0                  ;work exp val
        CLR1 FPR1_1.7           ;     sign of exp

        if (A < #N_BL)          ;'E' or 'e'
          CALL !GETC
          if     (A == #N_PL)
            CALL !GETC
          elseif (A == #N_MN)
            SET1 FPR1_1.7
            CALL !GETC
          endif
          if (A >= #N_9+1)
            goto ERROR
          endif

          X = A                 ;1st. digit
          CALL !GETC
          if (A < #N_9+1)
            B = A
            A = #10
            MULU X
            A = B
            X += A
            CALL !GETC
          endif
        endif

        if (A != #N_NL && A != #N_BL)
          goto ERROR
        endif
```

```
        if_bit (FPR1_1.7)
          A = #0
          A -= X
        else
          A = X                 ;exp.part value (:B)
        endif

;******* UNITE MANTISSA.val & EXP.val **

        A -= FPR2_2             ; B - (F-N) (:B')
        E = A
        if_bit (A.7)
          D = #0FFH
        else
          D = #0
        endif

        CALL !FTOL              ; B' → real

        HL = #C1
        CALL !LLD2CX
        CALL !LMLTX             ;log2(10) * B'
        CALL !LLD21X

        CALL !LTOF
        CALL !FTOL
        PUSH DE                 ;int(log2(10)*B')

        FPR1_4 ^= #80H
        CALL !LADDX             ;dec(log2(10)*B')

        HL = #C2
        CALL !LLD2CX
        CALL !LMLTX             ;dec(log2(10)*B')*log2

        A = FPR5_X
        PUSH AX                 ;esc. A' 4th mantissa
        CALL !LEXPX

        CALL !LLD25             ;ret. A'
        POP AX
        FPR2_X = A              ;ret. A' 4th mantissa
        CALL !LMLTX             ;A' * e^(dec(log2(10)*B')*log2)
```

```
        A = FPR1_4
        CY = FPR1_3.7
        ROLC A,1

        POP DE
        ADD E,A                 ;exp.part RESULT
        A = D
        ADDC A,#0
        if_bit (A.7)
          E = #0                ;underflow
        elseif_bit (!Z)
          goto ERROR            ;overflow
        endif

        CY = FPR1_4.7
        A = E
        RORC A,1
        FPR1_4 = A
        FPR1_3.7 = CY
T_TOL9:
        POP HL
        A = #R_OK
        CLR1 CY
        RET
ERROR:
        POP HL
        A = #R_ERR
        SET1 CY
        RET

GETC:
        A = [HL]
        HL++
        if (A >= #A_0 && A < #A_9+1)
          A -= #A_0
          RET
        endif
        C = A

        DE = #INDEX
        B = #S_INDX
```

```
        repeat
          A = [DE]
          DE++
          if (A == C)
            A = B
            A += #N_9
            RET
          endif
          B--
        until_bit (Z)

        A = #0FFH
        RET

INDEX:
        @_INDX
C1:
        DB 04BH,078H,09AH,054H,040H ;const. log2(10)
C2:
        DB 0F7H,017H,072H,031H,03FH ;       log2

        END
```

(29) **LTOA. SRC**

```
$       TITLE   ('TRANSLATE TO ASCII STRING')
        NAME    M_LTOA

#include "EQU. INC"
#include "ASCII. INC"
#include "REF1. INC"
#include "REF2. INC"

        EXTRN   LADDX, LMLTX
        EXTRN   LLOG10, LEXP10
        EXTRN   LTOF, FTOL

        PUBLIC  LTOA

S_VIRT  EQU     7       ;string length of mantissa

C2_4    EQU     41H
C2_3    EQU     20H

        CSEG
;*********************************************************
;*
;*  78K0 FLOATING POINT FUNCTION THAT
;*            TRANSLATE TO ASCII STRING
;*
;*     input  condition : FPR1 <- x
;*                       HL <- STORE ADDRESS of STRING
;*
;*     output conditions: STRING, HEAD IS APPOINTED TO HL
;*                       HL keep
;*********************************************************
LTOA:
        CY = FPR1_3.7
        A = FPR1_4
        ADDC A, A

        PUSH HL

;******* ZERO FORMAT **

        if_bit (Z)
          [HL]   = #A_0 (A)
          HL++
          goto T_TOA9
        endif
```

```
;******* TRANS. to a * 10^b (1<= a <10) **

        CALL !LLD51             ;esc. x to FPR5
        CLR1 FPR1_4.7

        CALL !LLOG10            ;log10(|x|)

        CALL !LTOF              ;trunc integer(log10(|x|))
        if_bit (Z)
          CMP FPR1_X,#0
        endif
        if_bit (!Z)                     ;include decimal digit &&
          if (FPR1_HP >= #8080H) (AX)   ;negative ?
            DE--                        ;floor integer(log10(|x|))  : b
          endif
        endif

        A = E
        PUSH AX
        if (A == #38)
          CALL !LLD15
          FPR1_4--
          FPR1_X = #0
          HL = #C1
          CALL !LLD2CX
          CALL !LMLTX           ;x/4 * ((10^-38)*4)
        else
          CALL !FTOL
          FPR1_4 ^= #80H
          CALL !LEXP10          ;10^(-b)
          CALL !LLD25
          FPR2_X = #0
          CALL !LMLTX           ;x * (10^(-b))
        endif
        POP AX
        FPR3_1 = A              ;esc. b

;******* OUTPUT MANTISSA **

        POP HL
        PUSH HL
        if_bit (FPR1_4.7)       ; a<0 ?
          [HL] = #A_MN (A)
          HL++
          CLR1 FPR1_4.7         ; a <- |a|
        endif
        PUSH HL
```

```
if (FPR1_HP >= #C2_4*100H+C2_3)  (AX)    ;if (limit |a|<10 over)
  HL = #C3                               ;  {normalize}
  CALL !LLD2CX
  CALL !LMLTX
  FPR3_1++
endif
if (FPR1_HP < #3F80H) (AX)       ;if (limit |a|>=1 over)
  HL = #C2                       ;  {normalize}
  CALL !LLD2CX
  CALL !LMLTX
  FPR3_1--
endif

CALL !LTOF       ;integer(a)
A = E
A += #A_0
POP HL
[HL] = A
HL++
[HL] = #A_PD (A)
HL++

B = #S_VIRT-1
repeat
  PUSH HL
  CALL !LLD21X
  CALL !FTOL
  SET1 FPR1_4.7
  CALL !LADDX            ;a - integer(a)
  HL = #C2
  CALL !LLD2CX
  CALL !LMLTX            ;(a - integer(a))*10

  CALL !LTOF
  POP HL
  A = E
  A += #A_0
  [HL] = A
  HL++

  B--
until_bit (Z)
```

```
;******* OUTPUT EXP.PART **

        [HL] = #A_E2 (A)
        HL++

        A = FPR3_1
        if_bit (A.7)
          [HL] = #A_MN (A)
          HL++
          A = #0
          A -= FPR3_1
        endif
        X = A
        A = #0

        C = #10
        DIVUW C
        A = C

        AX += #A_0*100H+A_0
        A <-> X
        [HL] = A
        HL++
        A = X
        [HL] = A
        HL++

T_TOA9:
        [HL] = #A_NL (A)
        POP HL
        A = #R_OK
        CLR1 CY
        RET

C1:
        DB 0EDH,0DCH,0C7H,059H,001H ;const (10^-38)*4
C2:
        DB 000H,000H,000H,C2_3,C2_4 ;const 10
C3:
        DB 0CDH,0CCH,0CCH,0CCH,03DH ;const 1/10

        END
```

(30) **FTOL. SRC**

```
$       TITLE    ('TRANS. FIXED TO REAL')
        NAME     M_FTOL

#include "EQU.INC"
#include "REF1.INC"

        PUBLIC   FTOL

        CSEG
;*************************************************************
;*
;*  78K0 FUNCTION THAT TRANSLATE FIXED TO REAL
;*
;*    input  condition : DE <- (integer with sign bit)
;*
;*    output condition : FPR1 <- (real value meaning DE)
;*                       DE keep
;*
;*************************************************************
FTOL:

;******* ZERO EXCEPTION **

        AX = DE
        if (AX == #0)
          FPR1_HP = AX
          goto T_TOL9
        endif

;******* GET ABSOLUTE VALUE **

        if (AX >= #8000H+1)
          A ^= #0FFH
          A <-> X
          A ^= #0FFH
          A <-> X
          AX++
        endif
```

```
;******* TRANSLATE **

        if (A == #0)
          C = #ZEROEX+BYTE-1
          A <-> X
        else
          C = #ZEROEX+BYTE*INTEGR-1
        endif

        while_bit (!A.7)
          A <-> X
          ROLC A,1
          A <-> X
          ROLC A,1
          C--
        endw

;******* STORE **

        FPR1_X = #0     ;4th mantissa
        FPR1_1 = #0     ;3rd mantissa
        A <-> X
        FPR1_2 = A      ;2nd mantissa

        A = D
        ROL A,1         ;CY <- sign
        A = C
        RORC A,1
        FPR1_HP = AX    ;sign,exponent,1st mantissa

        FPR1_3.7 = CY   ;exponent LSB

T_TOL9:
        A = #R_OK
        CLR1 CY
        RET

        END
```

(31) **LTOF. SRC**

```
$       TITLE   ('TRANS. REAL TO FIXED')
        NAME    M_LTOF

#include "EQU.INC"
#include "REF1.INC"

        PUBLIC  LTOF

        CSEG
;*********************************************************
;*
;*  78K0 FUNCTION THAT TRANSLATE REAL TO FIXED
;*
;*    input  condition : FPR1 <- (real value)
;*
;*    output condition : DE <- (integer value meaning FPR1)
;*                       ERROR then set CY
;*                       not INCLUDE DECIMAL PART then set Z
;*               keep : FPR1
;*
;*********************************************************
LTOF:
        CY = FPR1_3.7
        A = FPR1_4
        ROLC A,1
        A -= #ZEROEX

;******* EXCEPTION **

        if_bit (CY)             ;integer(FPR1)=0 ?
          CMP A,#LOW(-ZEROEX)
          DE = #0
          goto T_TOF9
        endif

        A -= #BYTE*INTEGR
        if_bit (!CY)
          goto ERROR            ;overflow
        endif
```

```
;******* GET UNSIGNED INTEGER **

        C  = A
        DE = FPR1_LP (AX)
        A  = FPR1_3
        A |= #80H               ;(set mantissa MSB)
        A <-> C

        if_bit (!A.3)
          SET1 A.3
          A <-> D
          E |=  A
          A  = #0
          A <-> C
          A <-> D
        endif

        A <-> C
        A <-> D
        X  =  A
        A  =  D
        D  = #0

        C++
        CLR1 CY
        while_bit (!Z)
          RORC A,1
          A <-> X
          RORC A,1
          A <-> D
          RORC A,1
          A <-> D
          A <-> X
          C++
        endw

;******* UNSIGNED -> SIGNED INTEGER **

        if_bit (FPR1_4.7)
          A ^= #0FFH
          A <-> X
          A ^= #0FFH
          A <-> X
          AX++
          if_bit (!A.7)
            goto ERROR
          endif
```

```
        else
          if_bit (A.7)
            goto ERROR
          endif
        endif

;******* DECIMAL PART JUDGE **

        XCHW AX,DE

        CMPW AX,#0
T_TOF9:
        A = #R_OK
        CLR1 CY
        RET
ERROR:
        A = #R_ERR
        SET1 CY
        RET

        END
```

付録　SPD チャートの説明 ★

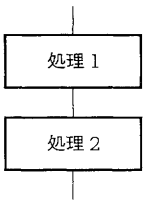
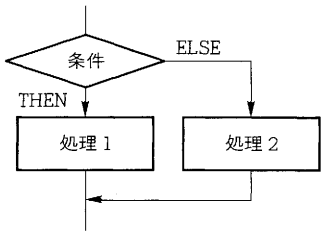
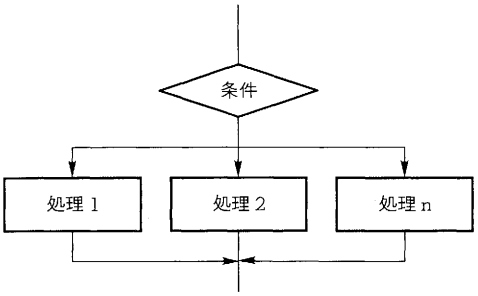
SPDとは，Structured Programming Diagrams の頭文字をとったもので，文字どおり訳せば「構造化プログラム図」と言えます。

構造化というのは，プログラムの論理処理の構造化のことで，論理の基本構造を用いて論理の設計，組み立てを行うことです。

すべてのプログラムは，論理の基本構造（順次，選択，繰り返し）の組み合わせのみで作成することができ（これを構造化定理といいます），構造化することでプログラムの流れが明確になり，信頼性が向上します。プログラムの構造化を表現するには，いろいろな方法がありますが，当社では，SPDという図式化技法を用いています。

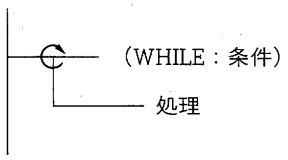
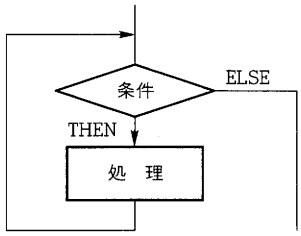
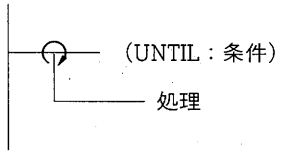
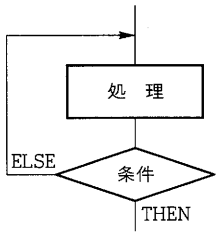
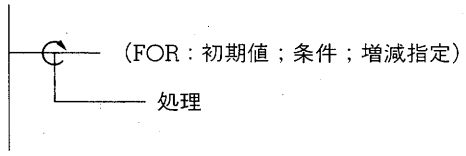
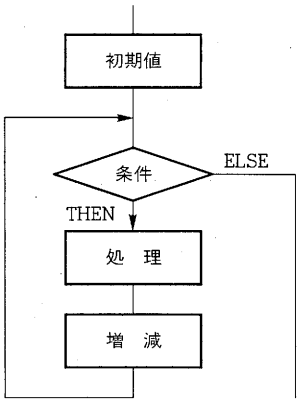
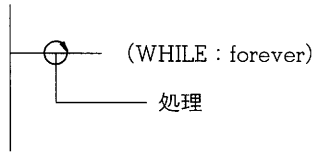
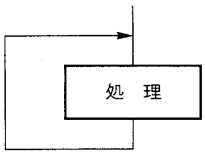
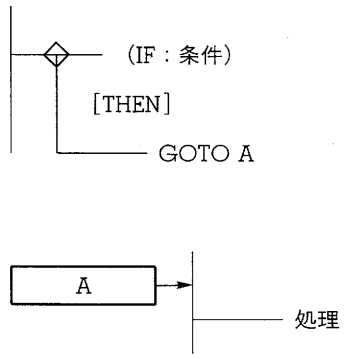
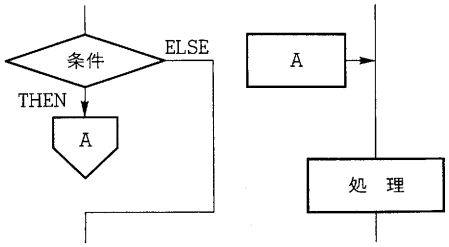
以下に，SPD技法で用いるSPD記号の説明，およびフロー・チャート記号との対比を示します。

表付－1　SPD記号とフロー・チャートの対比（1/2）

処理名称	SPD記号	フロー・チャート記号
順次処理	処理 1 処理 2	
条件分岐 (IF)	(IF：条件) [THEN] 処理 1 [ELSE] 処理 2	
条件分岐 (SWITCH)	(SWITCH：条件) [CASE：1] 処理 1 [CASE：2] 処理 2 ⋮ [CASE：n] 処理 n	

付

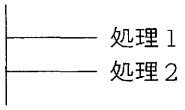
表付－1　SPD記号とフロー・チャートの対比（2/2）

処理名称	SPD記号	フロー・チャート記号
条件ループ (WHILE)		
条件ループ (UNTIL)		
条件ループ (FOR)		
無限ループ		
結合子		

1. 順次処理

順次処理は，処理を上から下へ出現順に実行します。

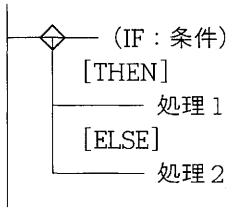
●SPDチャート



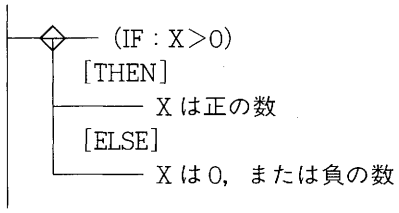
2. 条件分岐：２分岐（IF）

IFに示した条件の真偽（THEN/ELSE）により処理内容を選択します。

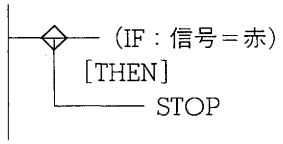
●SPDチャート



例１．Xの正負判別



２．信号が赤ならSTOPする



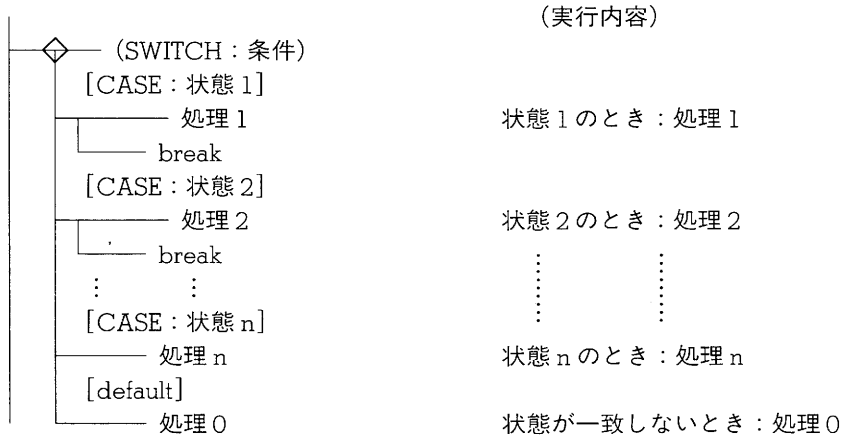
付

3. 条件分岐：多分岐（SWITCH）

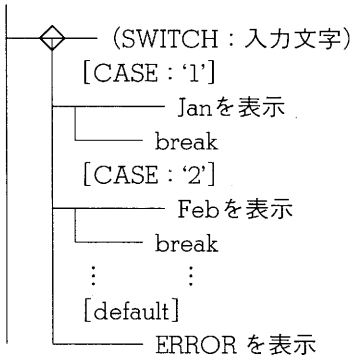
SWITCH に示した条件を，CASEで示された状態と比較し，処理を選択します。SWITCH文の処理は，一致した状態のみの処理を実行する場合と，一致した状態から下へ処理を続ける場合の二通りがあります（処理が下へ続かない場合は，'break' を記述）。また，一致した状態がない場合は，'default' の処理を実行します（'default' の記述は任意）。

（1）一致した状態のみの場合

●SPDチャート

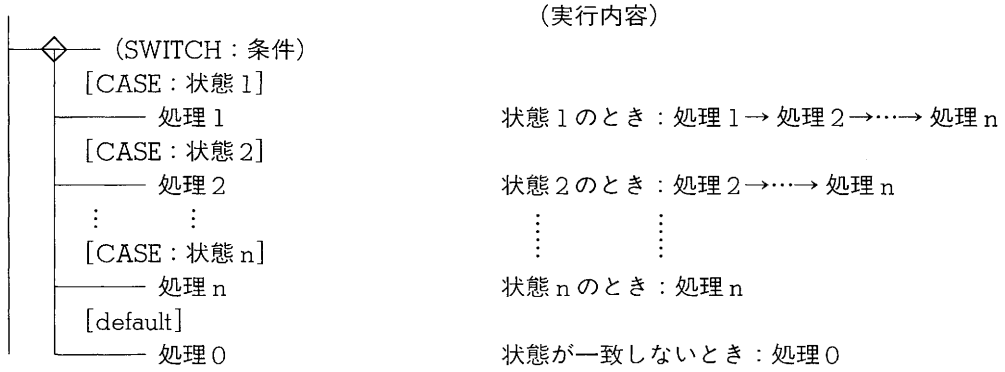


例　入力文字により，月名を表示する

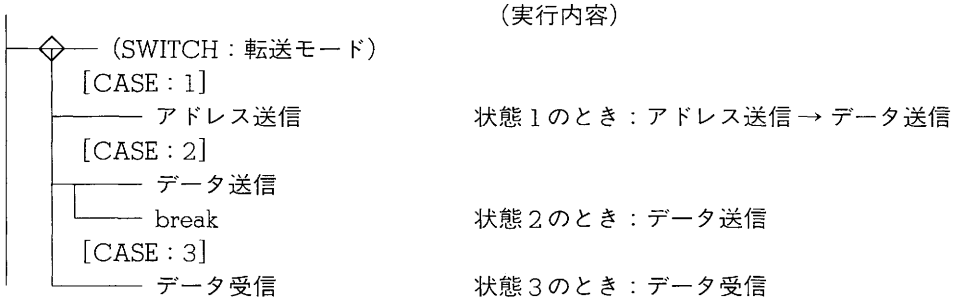


(2) 一致した状態から処理が続く場合

●SPDチャート



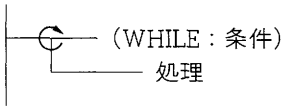
例　シリアル・インタフェースの送受信



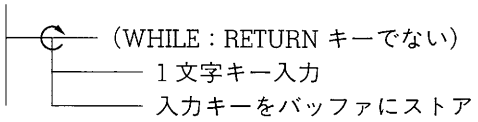
4. 条件ループ（WHILE）

WHILEに示した条件を判定し，条件が成立している間，処理を繰り返し実行します（初めから条件が不成立の場合は，処理を実行しません）。

●SPDチャート



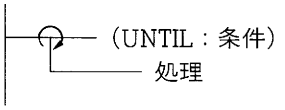
例　RETURNキー入力があるまで，キーをバッファリングする



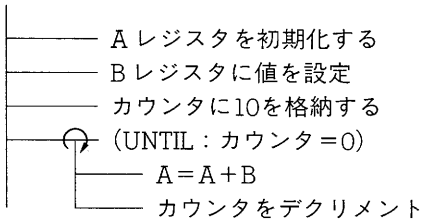
5. 条件ループ（UNTIL）

処理を行ったあとに UNTIL に示した条件を判定し，条件が成立するまで処理を繰り返し実行します（初めから条件が不成立の場合でも，処理を一時実行します）。

●SPDチャート



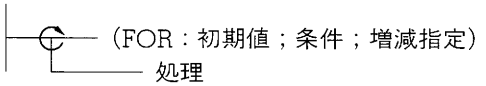
例　Bレジスタの値を10倍してAレジスタに格納する



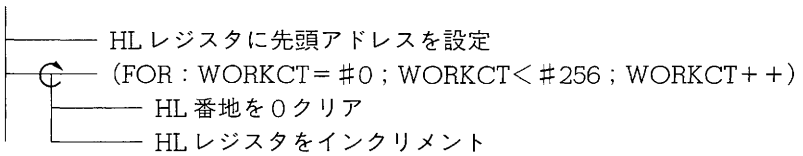
6. 条件ループ（FOR）

FORに示されたパラメータの条件が成立している間，処理を繰り返し実行します。

●SPDチャート



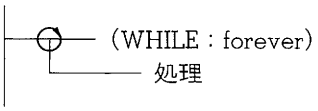
例　HL番地から256バイトを０クリアする



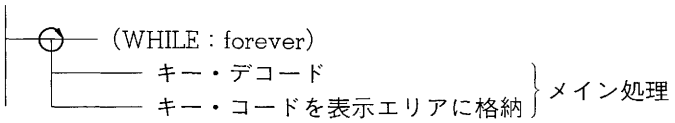
7. 無限ループ

WHILE の条件として 'forever' を設定すると，処理を無限に繰り返し実行します。

●SPDチャート



例　メイン処理を繰り返し実行する

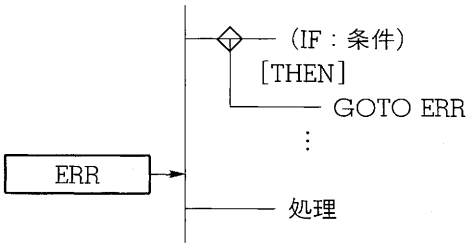


8. 結合子（GOTO）

無条件に指定されたアドレスに分岐します。

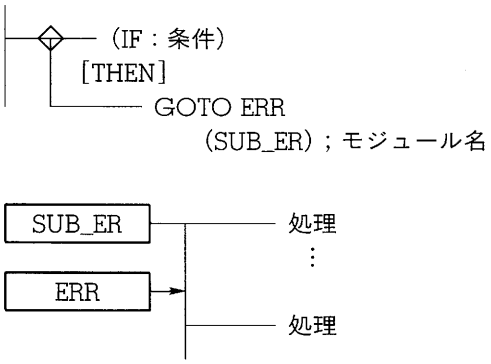
●SPDチャート

(1) 同じモジュールに分岐

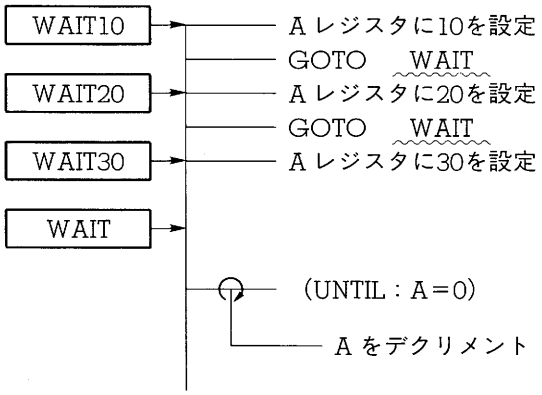


付

(2) 異なるモジュールに分岐



例　サブルーチンの開始アドレスで，パラメータを選択しウエイトを設定する



9. 結合子（継続）

1モジュールのSPDが，複数のページにおよぶ場合に使用し，処理の流れを示します。

●SPDチャート

〔メ モ〕

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン（インフォメーションセンター）

電　話　：044-548-8899
FAX　：044-548-7900
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部		〒108-8001	東京都港区芝５－７－１　（日本電気本社ビル）	(03)3454-1111
中部支社	半導体第一販売部 半導体第二販売部	〒460-8525	愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(052)222-2170 (052)222-2190
関西支社	半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540-8551	大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社	札　幌	(011)231-0161	宇都宮支店	宇都宮	(028)621-2281	北陸支社	金　沢	(076)232-7303
東北支社	仙　台	(022)267-8740	小山支店	小　山	(0285)24-5011	富山支店	富　山	(0764)31-8461
岩手支店	盛　岡	(019)651-4344	甲府支店	甲　府	(0552)24-4141	福井支店	福　井	(0776)22-1866
郡山支店	郡　山	(0249)23-5511	長野支社	松　本	(0263)35-1662	京都支社	京　都	(075)344-7824
いわき支店	いわき	(0246)21-5511	静岡支社	静　岡	(054)254-4794	神戸支社	神　戸	(078)333-3854
長岡支店	長　岡	(0258)36-2155	立川支社	立　川	(042)526-5981,6167	中国支社	広　島	(082)242-5504
水戸支店	水　戸	(029)226-1717	埼玉支社	大　宮	(048)649-1415	鳥取支店	鳥　取	(0857)27-5311
土浦支店	土　浦	(0298)23-6161	千葉支社	千　葉	(043)238-8116	岡山支店	岡　山	(086)225-4455
群馬支店	高　崎	(027)326-1255	神奈川支社	横　浜	(045)682-4524	松山支店	松　山	(089)945-4149
太田支店	太　田	(0276)46-4011	三重支店	津	(059)225-7341	九州支社	福　岡	(092)261-2806

C98.2

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］　78K/0シリーズ　アプリケーション・ノート　浮動小数点演算プログラム編
（U13482JJ2V0AN00（第２版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
そ　の　他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは
NEC 販売員，特約店販売員，NEC 半導体ソリューション技術本部員，
その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC 半導体テクニカルホットライン
FAX：(044)548-7900

キリトリ