PascalによるPascalのための実行時支援システム

東京工業大学総合情報処理センター

白濱律雄　前野年紀
Ritsuo Shirahama　Toshinori Maeno

要旨

アセンブラで書かれていたＰａｓｃａｌ用実行時支援システムを作り直してＰａｓｃａｌで記述し、保守改良を容易にした。

従来のコンパイラに手をくわえないで、そのコンパイラでそのコンパイラ用の実行時支援システムを作成し、次に両者を若干変更して、実行時支援システムの構成を簡潔にした。コンパイラとの機能分担が明確になり、Ｐａｓｃａｌだけで閉じたセルフ・サービス方式のシステムになった。

作成の目的、経過、改良例を説明し問題点、意義について述べる。

目次

１．　目的とその背景

アセンブラで書かれていたＰａｓｃａｌ用実行時支援システムを作り直して、Ｐａｓｃａｌで記述し、保守改良が容易な形にした。

ソフトウェア工具の作成にＰａｓｃａｌを使用している。現用の処理系（ＰＡＳＣＡＬ８０００〔2〕。以後ＡＡＥＣ版と呼ぶ）は多くの人の手を経て、いいものに育ってきている〔１，２，３〕。最近はメーカ製の処理系もあるが、使う気になれない。しかし、道具を作るための道具の常で、保守改良をおこたれば陳腐化する。コンパイラはＰａｓｃａｌで書かれており、我々も保守改良してきている〔3〕。

ところが実行時支援システムは４２００行もあるアセンブラ・プログラムで、しかも内部説明書がないため、重大な虫が発見された時、それを取り除くだけで精一杯であった。そのため、手続きの個数の制限（２５６個まで）を除く、漢字を扱えるようにする、というような、実行時支援システムの改造を伴なう改良案件がたまってしまった。改造するよりも、今後の発展の土台になるものを作ることを選んだ。

作成言語としてＰａｓｃａｌを選択したのは、使い慣れている、他に適当な言語が手元にない、

という消極的理由ばかりでなく、Ｐａｓｃａｌで書けば全てがそれだけで閉じるという積極的理由もあった。

アセンブラで書くのは、最小にとどめる方針でのぞんだ。ＩＢＭ系のオペレーティング・システム（ＯＳ）は、高級言語との相性が悪い。しかも、使う側でも複雑な処理を避けられないのであるから、アセンブラには頼るべきでない。こんな例があった。ＯＳは日付けを１月１日からの通算日数で管理しているので、実行時支援システムで月と日に変換しなければならない。アセンブラで書かれたコードに不良があり、うるう年には狂うようになっていた。桁違いに複雑な入出力処理がアセンブラでまともにできるであろうか？

２章と３章では、ＡＡＥＣ版の紹介をかねて、実行時支援システムの役割と構成を説明する。

４章が新版の作成経過と構成である。５章、６章でその改良成果と残された問題点を述べる。

なお、この研究には東京工業大学総合情報処理センターのＨＩＴＡＣ　Ｍ－２００Ｈシステム（ＯＳはＶＯＳ３）を利用した。

２．　実行時支援システムの役割と構成

土台にしたＡＡＥＣ版を例にとり、実行時支援システムの役割と構成を説明する。

２．１　役割（コンパイラとの関係）

処理系はコンパイラと実行時支援システムとからなる。コンパイラは、入出力のような複雑な処理はサブルーチン呼び出しの形に変換する。オブジェクトは、対応するサブルーチンと結合されて、目的の処理を行なう。このサブルーチンの集りが実行時支援システムである。

実行時支援システムは、言語のための環境を提供する仮想計算機を形成しているといえる。

２．２　構成

ＡＡＥＣ版の構成と呼び出し関係を図１に示す。作業は大きく、オブジェクトの実行の前（ＩＮＩＴ）、最中（ＲＵＮ）および後（ＴＥＲＭ）にわけられる。ＭＣ１が全体を制御する。オブジェクトは、実行時支援システムにより環境が整えられてから呼び出されるようになっている。ＳＢはＲＵＮの出店のようなもので、３章で説明する。

（１）　前準備（ＩＮＩＴ）

コンパイラが前提としている実行環境を設定する

（ａ）メモリを取得する。取れるだけ取って、スタックおよびヒープ領域として使う。

（ｂ）テキスト型ファイルＯＵＴＰＵＴを開く。実行時支援システムがエラー・メッセージを出す口を確保することでもある。

（ｃ）手続き表を一本化する。この処理系では、１つの手続きを翻訳単位にできる（外部手続きと呼ぶ）。各翻訳単位に付属する手続き表（手続きの入口点番地列）を１つにまとめる。再帰呼び出しで行なう。

（2） 実行中の支援（ＳＢとＲＵＮ）

これが実行時支援システムの主要部分である。その大部分は入出力であり、標準手続きと標準関数に対応している。また、手続きから飛び出すｇｏｔｏや手続き呼び出しも実行時支援システム経由で実行される。

（3） 後仕末（ＴＥＲＭ）

エラーを検出した場合には、メッセージを表示し、状況把握のための情報としてトレース・バック（手続き呼び出しの連鎖と各々の局所変数の値）を出力する。

開かれたままのファイルがあれば閉じ、メモリを返す。

その他、共通データ領域、浮動小数点数出力ルーチン、算術関数群等がある。各々のサイズを表１に示す。

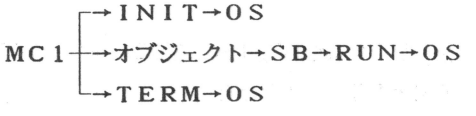

図１ ＡＡＥＣ版の構成と呼び出し関係

ＭＣ１（全体の制御）	２００行
ＩＮＩＴ（環境設定）	４５０行
ＲＵＮ（実行中の支援）	１６００行
ＳＢ（ＲＵＮの出店）	１３０行
ＴＥＲＭ（トレース・バック等）	８７０行
共通データ領域	５０行
浮動小数点数出力ルーチン	２９０行
算術関数群合計	５８０行
合計	４２００行

表１ ＡＡＥＣ版のサイズ

３．　ＡＡＥＣ版に近付いて見ると

ＡＡＥＣ版の興味深い所（ＳＢ）を紹介する（図２）。メモリの小さい計算機用に１バイトでもオブジェクトを小さくしようとして設計されたのではないかと思われる。

３．１　方式

スタックとヒープは１つの領域を向い合わせで使う。衝突するとスタック・オーバフローである。

ディスプレイ方式を採用している。汎用レジスタ（１６個）の内６個をディスプレイ・ベクタとして使う。静的な入れ子の深さは６段までである。

最外殻のディスプレイ・レジスタの内容は不変であり、ＳＢのベース・レジスタとしても使う。つまりＳＢは、取得したメモリの先頭（スタックの底）にコピーして使われる。

３．２　手続きの呼び出しと戻り

（１）内部手続き

ｃａｌｌというのは、手続き呼び出しの命令列を短かくするためにコードをここに埋たものである。手続きを呼ぶ時はここに飛び込み（１命令と手続き番号）、７命令で、

（ａ）ディスプレイ・ベクタ等環境をおさえているレジスタを退避し、

（ｂ）手続き本体用ベース・レジスタ（入口点の番地）を手続き表からもとめ、

（ｃ）呼び出される手続き用にディスプレイ（その時点のスタック・ポインタ）を準備し、

（ｄ）スタック・ポインタを必要なだけ延し、

（ｅ）ヒープと衝突していないことを確認し、

目標に飛び込む。

コンパイラの場合、手続きが２５２，呼び出しが１０００もあるので、オブジェクト短縮効果は大きい。

（２）外部手続き

ｃａｌｌを通ってつなぎのコードに飛び込み、そこで手続き表のポインタ（実は、ｃａｌｌの中の命令）を退避更新し、その上でまたｃａｌｌを通って目標に入る。

（３）パラメータ手続き

オブジェクト中で数命令実行した後ｃａｌｌの途中に飛び込む。

（４）手続きからの戻り

戻る時もスタック中を経由する（ｒｅｔｕｒｎ）。どの深さから抜けるかにおうじて６種類（１つにつき２命令）ある。時間切れになった時、これらの命令を書き換えることによって終了処理に入るためである（６．（２）参照）。

３．３　データ領域

（１）共通領域

実行時支援システムとオブジェクトの共通領域であり、ＯＵＴＰＵＴのファイル変数へのポインタ、ヒープ・ポインタ等がある。

（２）ジャンプ・テーブル

ＲＵＮの各種ルーチンに飛び込むための中継点であり、サブルーチン呼び出し命令がならんでいる。巧妙な作りで、オブジェクトからは１命令で飛び込める。

（３）手続き表

ｃａｌｌ中の命令列を短かくするために、手続き表はすぐに手が届く範囲においておく。この表の場所が２５６項目分しかない。つまり、この制限は翻訳単位に関するのではなく、実行単位での上限である。コンパイラの場合既に手続き数が２５２になっているので、機能追加がしにくくなっている。

（４）主プログラム用変数領域

ユーザの宣言した変数に割り当てられる。その開始番地（偏位）はコンパイラもつかっているので、実行時にはかえられない。手続き表を延ばせば、直接アドレスできる変数の範囲が狭まる。既に４０９６バイトの内、１５８４バイトが上記の用途にしめられている。

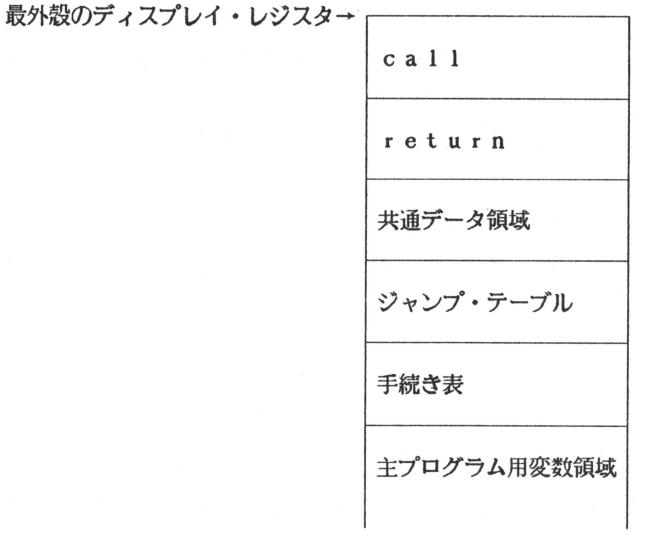

図２　　スタックの底のプログラムとデータ（ＳＢ）

４．　ＴＩＴ版の作成経過と構成

新しく実行時支援システムを設計し直し、同時にコンパイラを大改造すると、デバッグが大変である。まずは、オブジェクトプログラムと実行時支援システムとのインタフェースはかえないで、実行時支援システムをＰａｓｃａｌで作成して（ＴＩＴ版と呼ぶ）アセンブラ版と置き換え、その後で改良していく方針でのぞんだ。

４．１　ＴＩＴ１版

まず、ＡＡＥＣ版をそのままおきかえる形で作成した。構成を図３に示す。

実行時支援システム本体（ＲＳＰ１）とオブジェクトの両方がＰａｓｃａｌの主プログラムになっている。

スタックは実行時支援システム用とオブジェクト用に２つ使い（前者は少量を固定長でとる）、ＲＳＡ１が、両者のインタフェースを合わせ、それらがコルーチンとして動くようにする。高速化のために、ここだけで終える処理もある（例えば、手続き呼び出しのサポート）。アセンブラで１４０行ある。ＯＳを呼び出すのに、アセンブラの小さなルーチンを介する（ＯＳＣ１）。バラバラのものが１３ある（合計２００行）。これらはオブジェクトの構造が関わるような操作はしない。単にＯＳを呼ぶ（いわゆるマクロ命令の実行）だけである。ＲＳＡ１やＯＳＣ１は、ＰＡＳＣＡＬ８０００が持つＦＯＲＴＲＡＮプログラムを呼び出す機能を利用して呼び出す。

この構成では、Ｐａｓｃａｌで書かれた実行時支援システムのそのまた後立てが必要になる。とりあえず、ＡＡＥＣ版をそのまま実行時支援システムの後立てとして用いるが、できるかぎり、そのお世話にならないようにした。テストがすんだ段階で、ＡＡＥＣ版のかわりに、ＲＳＰ１が動くために必要な最低の機能しかもたない縮退版ＭＲＳを作成した。アセンブラで２００行（図４）。ＲＳＡ２とＭＲＳは非常によく似ている。

これでも一応使い物になるが、ｗｒｉｔｅのような手間のかかる処理はＭＲＳが支援しないので、ＲＳＰ１では使えない。エラー・メッセージやトレース・バックを出すのが面倒で、テストのためにスナップ・ショットを打つことも気楽にはできない。ＭＲＳがＡＡＥＣ版ならば可能であるが、いつまでもＡＡＥＣ版を捨てられないのも困る。

４．２　ＴＩＴ２版

Ｐａｓｃａｌで書いてある部分を主プログラムではなく、外部手続きにした（ＲＳＰ２，図５）。ＲＳＰ２もＰａｓｃａｌの全ての機能を利用できるようにするためである。形は１つの手続きであるが、概念的にはサービスに対応する手続きの集りである。ＲＳＰ２内で実行時支援システムのサポートを必要とする操作（例えばｗｒｉｔｅ）をすると、ＲＳＡ２経由で再帰呼び出しになりＲＳＰ２自身が処理する（セルフ・サービス）。ＲＳＡ２はオブジェクトから呼ばれたかＲＳＰ２から呼ばれたかを区別しない（できない）。コンパイラも、実行時支援システム呼び出し、外部手続き呼び出し関係等を変更した。スタックは一本化され、環境管理用のデータは、主プログラムの変数用ディスプレイでさされる領域におく。

この構成にするためには、環境作成の内どこから後をＰａｓｃａｌでできるか（どれだけはアセ

ンブラでするしかないか）が問題であった。結局、全体を制御するＭＣ２でメモリの取得とレジスタ設定をおこなえば、後はＰａｓｃａｌでできることがわかった。手続き表は一本化をやめ、分散したまま使い、外部手続き呼び出し時にポインタを退避、更新する。

エラー検出時にユーザに必要な情報を適切な形式で表示する所では、Ｐａｓｃａｌで書いている有り難さが発揮された。気楽にいろいろ試し、改良していける。アセンブラでは到底不可能なことである。

ＯＳＣ２は、ＯＳＣ１と機能は同じだが、外部Ｐａｓｃａｌのインタフェースで呼ばれる。リエントラントにするためである。

４．３　アセンブラで残した部分

アセンブラで残してある部分は、コンパイラのコード生成部とオブジェクトを大きくしてまで展開する価値がなく、Ｐａｓｃａｌで書くとそこへの出入に要する手間で遅過ぎる処理（ＮＥＷ，ＭＡＲＫ，ＲＥＬＥＡＳＥ等）である。

Ｐａｓｃａｌプログラムから直接ＯＳを呼べるようにもできるが、どうせＯＳを呼べばかなりの時間がかかる。呼び出しの手間は無視できる。全体のサイズ、スピード、効果、保守の容易さ等のバランスできまる。

ＴＩＴ２版の構成要素ごとのサイズを表２に示す。

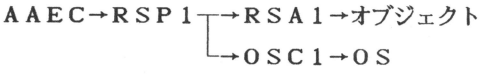

図３　ＴＩＴ１版の構成と呼び出し関係（テスト中）

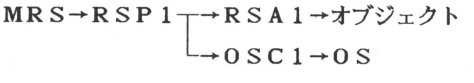

図４　ＴＩＴ１版の構成と呼び出し関係（本番）

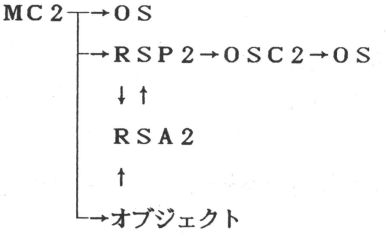

図５　ＴＩＴ２版の構成と呼び出し関係

ＭＣ２（メモリの取得と全体の制御）	１００行	
ＲＳＰ２（本体。おもに入出力）	１４００行	（ｐａｓｃａｌ）
ＲＳＡ２（ＲＳＰ２の出店。インタフェース整合）	２００行	
ＯＳＣ２（ＯＳを呼ぶ小さなルーチンの集り）	２５０行	
合計	１９５０行	

表２　ＴＩＴ２版のサイズ

５．　改良成果

現時点までに以下のような改良がなされている。

（１）　機能分担の明確化。分散していた実行環境の管理（手続き呼び出し、スタック、ヒープが関係する部分）を全面的にコンパイラから実行時支援システムに移した。集中管理されるので、理解しやすくなった上、コンパイラは小さくなり、実行時支援システム側だけで改良できる範囲がひろがった。

（２）　レジスタ割り付けを変更して手続き呼び出しを高速化した（もともとＦＯＲＴＲＡＮのサブルーチン呼び出しの６割）。レジスタ割り付けが関係する処理が局所化されたので、変更は容易であった。

（３）　リエントラント化（共用可能化）。Ｐａｓｃａｌのオブジェクトはもともとリエントラントであったので、アセンブラ部分をリエントラントに作ることにより、実行時支援システム全体を主記憶中に常駐させて複数プログラムで共用できるようになった。ＰａｓｃａｌプログラムをＴＳＳコマンドにした時の応答がよい。実行時支援システムが大きくなっても気にしなくてよく、オブジェクト保存用ファイル領域は少なくてすむ。

（４）　文の実行回数を計る機能を加えた。プリプロセッサで行なうより短かいコードで高速にカウントできる。エラー検出による終了でも結果が取れるようになった。

（５）　従来、個々のプログラムで宣言していたライブラリ・プログラムの一部をシステムに組み込んだ。例えば、漢字コードのファイルを扱うルーチン群である。

（６）　ＡＡＥＣ版はエラーを検出すると、その場所を番地（１６進）で指摘していたが、ＴＩＴ版では、手続きの先頭からの行数で表示する。

６．　問題点と将来構想

（１）　標準Ｐａｓｃａｌの機能だけでは無理で、規格外機能を使っている。

（ａ）外部手続き　実行時支援システムは外部手続きである。また、ＯＳを呼ぶ等も外部手続きのような顔をしてアセンブラで書いてある。

（ｂ）ポインタ　オブジェクト中のデータの参照にポインタ型変数を使う。ヒープの外を指すので、実行時検査をうけないように指定している。

（ｃ）レコード型のｃａｓｅの悪用　ｃａｓｅでＦＯＲＴＲＡＮのＥＱＵＩＶＡＬＥＮＣＥのように重ねて定義しておき、一方から代入して他方から取り出す。

（ｄ）タイプ変換関数　変数の値を他の任意の型として取り出す機能。

（２）　非同期に発生する事象（例えば演算結果のあふれ、時間切れ）への対応が現在不十分であるが、ＯＳの設計が悪いためである。例外処理ルーチンをＯＳに登録することはできるが、時間切れ、プログラム割り込み、タスク異常終了の３つが別々で、インタフェースも異なる。例外処理ルーチンに渡すパラメタをＯＳに預けることもできない（リエントラントにしにくい）。時間切れになっても、そこでオブジェクトの実行を打ち切って終了処理に入ることはできない。ＡＡＥＣ版が不十分を承知でｒｅｔｕｒｎの書き換えをしていたのはそのためである。ＴＩＴ版では、時間切れになったらあるフラグをたて、文の実行回数のカウント・アップ時にそのフラグを検査する。したがって、確実ではあるが、実行回数計測モードでしか機能しない。

（３）　標準手続きのコンパイラ組み込みを簡単にできるようにしないと、せっかく改造が楽になった実行時支援システムがいきてこない。コンパイラも実行時支援システムも骨だけにして、後はバラバラでライブラリとしてつけていく方向をめざしている。何が骨、にかわ、肉になるかが問題である。各々のモジュラリティを上げ、全体としてのサイズを下げ、保守を容易にする。

７．まとめ

いい言語の処理系全体を完全に保守改良できるようになった。これは強い武器である。

処理系の使い勝手は、実行時支援システムの出来具合に左右される。処理系外の変化を素早く取り入れ、先取りし、木目の細かいサービスをして行くことは、低級言語では難しい。

Ｐａｓｃａｌで書いたので、Ｐａｓｃａｌ処理系全体がＰａｓｃａｌ（と少量のアセンブラ）だけで完結した。セルフ・サービス方式だから、実行時支援システム用の実行時支援システムは不要である。その分サイズが小さいという利点もある。中味がわからない他言語の実行時支援システムが入らないので気持がよい。オブジェクトが若干大きくなり、実行時間も多少のびているが、軽微であり、問題にしていない。

全てが１つの言語ですめば、必要な知識、道具が少なくてすむ。保守対象の言語でかいていれば、いざという時にもコンパイラの改造で切り抜けられる。痒い所に手が届かない思いをしなくてすむ。

このシステム中には、現オペレーティングシステム上で、高級言語プログラムの実行を支援するための部品がそろっているので、他の言語用にも使えると思われる。処理系開発がコンパイラ作成とこのシステムの改造だけですむ。実行時支援システムの作成は泥臭く手間のかかる仕事であるから、それが楽になれば、言語の試作実験を気楽にできる。

最後に、小さくなったとはいえ、２０００行もの実行時支援システムが必要なのは、ＯＳが悪いからであることを強調したい。Ｐａｓｃａｌが必要とする程度のサポートがＯＳの標準的なサービスになっていないのがおかしい。よくできたハードウェア、ソフトウェアの上であれば実行時支援システムなどいらないはずである。

謝辞

算術関数群をＰａｓｃａｌで書いてくださった日立製作所システム開発研究所の浜田穂積氏に感謝します。また、討論していただいた東京工業大学情報科学科の、角田博保、佐渡一広、久野靖の３氏およびＰＴＴ（ｐｒｏｇｒａｍｍｉｎｇ　ｔｏｏｌｓ　ａｎｄ　ｔｅｃｈｎｉｑｕｅｓ）のメンバにも感謝します。

参考文献

〔1〕Ｈｉｋｉｔａ，Ｔ．　ａｎｄ　Ｉｓｈｉｈａｔａ，Ｋ．
ＰＡＳＣＡＬ　８０００　Ｒｅｆｅｒｅｎｃｅ　Ｍａｎｕａｌ　ｖｅｒｓｉｏｎ　１．０
Ｕｎｉｖｅｒｓｉｔｙ　ｏｆ　Ｔｏｋｙｏ，１９７６

〔2〕Ｃｏｘ，Ｇ．　ａｎｄ　Ｔｏｂｉａｓ，Ｊ．
ＰＡＳＣＡＬ　８０００　Ｒｅｆｅｒｅｎｃｅ　Ｍａｎｕａｌ　ｖｅｒｓｉｏｎ　１．２
Ａｕｓｔｒａｌｉａｎ　Ａｔｏｍｉｃ　Ｅｎｅｒｇｙ　Ｃｏｍｍｉｓｓｉｏｎ，１９７８

〔3〕白浜律雄、前野年紀　字句解析部の高速化について（ソフトウェアの調整技法）
第２０回プログラミングシンポジウム　情報処理学会　１９７９