高水準言語指向マシンにおける命令実行部の作成

5E-8

吉野 真澄*, 天満 隆夫*, 坪谷 英昭*, 田中 稔**, 市川 忠男**
*広島大学大学院, **広島大学工学部

1. はじめに

最近のプログラミング言語(例えば、Ada, Clu, Modula-2 など)には、抽象データ型、例外処理、データのきびしい型づけなど、信頼性の高いプログラムを書ける機能が導入されている。このような言語で書かれたプログラムは種々の実行時チェックを必要とする。通常、実行時チェックはプログラムのコンパイリング段階にコンパイラが生成し付加するコードによって行なわれるため、コンパイラの負荷が増大している。また、これらのコードは実行速度向上のために運用段階では取り除かれる場合があり、これでは実行時信頼性が保証されているとはいえない。本研究では、Ada、Pascalといった言語を対象とし、(1) プログラムの実行時信頼性の保証、(2) コンパイラの負荷の軽減を目標として、実行時チェックをマシンのレベルで行なう高水準言語指向マシンSOA(A Software Oriented Architecture)を設計し、その命令実行部をソフトウェアで作成した。

2. 高水準言語指向マシンSOA

SOAの特長を以下に述べる。

(1) データ写像方式

SOAではプログラム中のオブジェクトは Object Descriptor(OD)、Type Descriptor(TD)によってメモリ内に写像される。ここでオブジェクトとは Ada における定数や変数などのデータ、型、手続きや関数などの副プログラム、ラベル、タスク及びパッケージのことをいう。

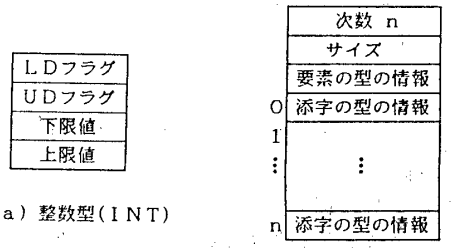
ODはオブジェクトに関する情報を保持し、すべてのオブジェクトに対してODはただ一つ存在する。図1にODの書式を示す。図において、OTI (Object Type Identifier)はSOAが処理するプリミティブな型を示す。Pフラグならびに Mフラグは対象とするオブジェクトの型宣言がどこにあるかという情報を保持する。Cフラグならびに Lフラグはそれぞれ対象とするオブジェクト(データ)が制約を持っているか、あるいは定数であるかといった情報を保持する。モジュール名、プロセス名の属性には、このオブジェクトの型宣言が自プロセス以外でなされているときに、型宣言しているプロセスの情報が保持される。ここで、プロセスとはAda における手続きや関数などの副プログラムのことをさし、モジュールとは主プログラム、パッケージ、及びタスクのことをさす。TDのオフセットはこのオブジェクトのTDがどこに記憶されているかを示す。データ値のオフセットはオブジェクトの値がどこにあるかを示している。

TDは型に関する情報を保持し、型によって異なる書式をとる。図2に例として整数型と配列型のTDの書式を示す。

図3に写像方式の概要を示す。

OTI	Lフラグ	Cフラグ	Mフラグ	Pフラグ	モジュール名	プロセス名	TDのオフセット	データ値のオフセット

図1 オブジェクト・ディスクリプタ (OD)



(a) 整数型(INT)

(b) 配列型(ARRAY)

図2 TDの書式

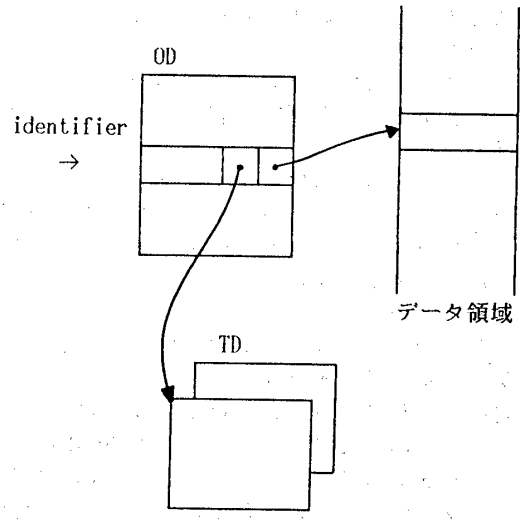


図3 写像方式の概要

Implementation of an Interpreter of a High Level Language Machine
Masumi YOSHINO, Takao TENMA, Hideaki TSUBOTANI, Minoru TANAKA, and Tadao ICHIKAWA
HIROSHIMA UNIVERSITY

(2) 演算方式

ＳＯＡではスタックを用いて処理を行なう。スタックには図４に示すようなオブジェクトに関する情報が積まれる。ＯＴＩは演算命令を解釈するために用いられる。すなわち、ＳＯＡでは加算に対する命令コードは一つしかなく、実際に行う加算はＯＴＩに従って決定される。ＩフラグとＣフラグはそれぞれそのオブジェクトが演算の中間値であるか、及び制約を課されているかを表す。ＴＤへのポインタはオブジェクトの型に関する情報を得るために用いられる。データ値へのポインタはデータ領域に写像されたオブジェクトにアクセスするために用いられる。データ・サイズは演算の中間結果のデータ領域を解放するために用いられる。スタックに積まれる情報はＯＤ，ＴＤを参照することにより得られる。

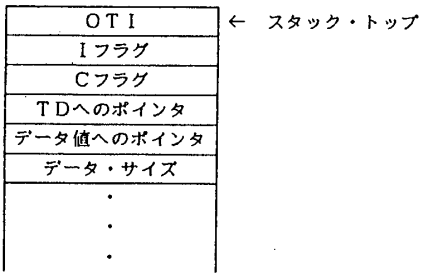


図４　スタックに積まれる情報

３．実行時チェック

(1) チェックの種類

ＳＯＡは命令実行時に以下のようなチェックを行ない、エラーの検出を行なう。

・代入可能性のチェック

あるオブジェクトに値を代入する際、そのオブジェクトに制約が課されていれば、代入される値がその制約を満たすかどうかのチェックを行なう。

・配列の添字チェック

配列のある要素にアクセスする場合、アクセスする要素の添字がその範囲制約を満たすかどうかのチェックを行なう。

・可変レコード型におけるチェック

可変レコード型のオブジェクトにおいて、アクセスが許されていないフィールドにアクセスをしていないかどうかのチェックを行なう。

・アクセス型のチェック

アクセス型のオブジェクトがオブジェクトを指しているかどうかをチェックする。

上記のようなチェックにおいてエラーが検出されると、マシンは制御を例外処理ルーチン(例外ハンドラ)に移す。例外ハンドラが定義されていなければ、エラー・メッセージを出力した後、プログラムの実行を停止する。

(2) チェックのメカニズム

具体例を用いて実行時チェックがどのように行なわれるかについて説明する。代入文 a := b + c は次のような命令列に変換される。

PUSH b; PUSH c; ADD; PUSH a; STORE;

a に範囲制約が課されていれば、代入時にb + c の値が a の範囲制約を満たしているかどうかをチェックしなければならない。このチェックはスタックに積まれた情報を用いて行なわれる。図５にSTORE 命令が行われる前のスタックの状態を示す。STORE 命令の実行時にオブジェクト a に範囲制約が課されているかどうかをＣフラグによって判定する。次にＴＤへのポインタをたどることにより制約の上下限値を得た後、代入する値が制約を満たすかどうかを検査する。b + c の計算結果の値が制約を満たしていれば a のデータ領域へその値を書き込む。制約を満たしていなければ例外CONSTRAINT_ERRORを発生させる。

a に制約が課されていなければチェックは行わずにすぐ値が書き込まれる。

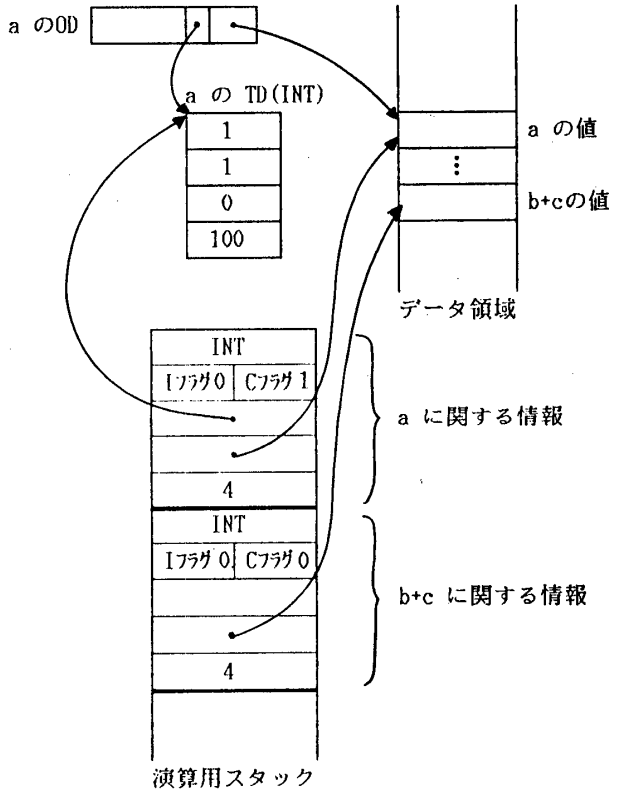


図５　実行時チェックにおけるスタックの状態

４．おわりに

以上に述べたＳＯＡをVAX11/750 上にＣ言語で作成した。プログラム行数は約4000行である。現在、ＳＯＡ上でのプログラムの実行を支援するツール(コンパイラ、デバッガなど)を作成中である。

参考文献

(1)G.J.Myers;"Advances in Computer Architecture," John Wiley & Sons, 1978.

(2)箱崎勝也,山本昌弘;"高級言語マシンの実際," 産報出版, 1981.

(3)H.Tsubotani, N.Monden, M.Tanaka, and T.Ichikawa;"A High Level Language-Based Computing Environment to Support Production and Execution of Reliable Programs," IEEE Transactions on Software Engineering, Vol. SE-12, No.1, pp.134-146, Jan. 1986.