OS／omicron V4 上の複数の浮動小数点演算方式をサポートするエミュレータの実現

4M－7

山本 康弘、早川 栄一、並木 美太郎、高橋 延匡
東京農工大学 工学部 電子情報工学科

1. はじめに

コンピュータを使用した数値演算では、その用途の多様性から幅広い数値を表現できることが要求され、さまざまな浮動小数点での数値の表現方法が生まれてきた。例を挙げれば、IEEE、URR、松井・伊理方式などである。

これらの仕様、特性は異なった視点から作られているため、大きく異なってくる。ここで、我々がこれらの形式が異なった浮動小数点を評価する場合は、同一プログラムにおいて、誤差の計測、桁落ち、アンダフローなどの考察を行うことが考えられる。

また、数値演算の分野の教育においては、実行中のメカニズムを見せることで、演算の方法を容易に理解させるようなことも考えられる。

このため、我々はOS／omicron V4 [1] 上に、ユーザが容易に手を加えられ、複数の浮動小数点表現方式の実行環境を提供する演算エミュレータを実現した。また、言語処理系においては、Intel x86 系のコードを生成するCAT386 [2] に、ダイナミックリンクを使用して、複数の浮動小数点表現方式に対応した実行コードを生成する機構を追加した。

2. システムへの要求

複数の浮動小数点表現方式に対応した演算エミュレートを行うときには、その選択において多くの実行環境が予想される。このとき、実行システムや言語処理系は、全てに対応できる環境を整えておくか、環境が変わる時には再生成する必要がある。

また、浮動小数点演算には多くの専用ハードが存在する。だが、教育に使用する時などでは、この専用ハードで実行するか、エミュレータで実行するかは、実行段階になるまでは、確定しない。

そのため、本システムでは、システム等の再生成の手間を省き、表現形式、実行環境に対応したシステムを動的に構築できるようにする。

3. システムの設計方針

本システムの設計方針は次のとおりである。

(1) 表現方式に応じて、システムを動的に構築する
使用する浮動小数点の表現形式を変更するたびに、OS や言語処理系を再構成する手間を省く。

(2) 専用ハードの有無で実行コードを変更しない
専用ハードの使用は、上で述べたように実行段階にならなければわからない。このため、専用ハードをモデルとしてエミュレートを行う場合には、別コードを生成することなしに実行できるようにする。

4. 演算エミュレータの設計

演算エミュレータは、筆者の所属する研究室で開発されたOS／omicron V4 上に実装する。システムを動的に構築するために、次のようにモジュール分けを行った。

(1) プロセッサ情報操作部
(2) 命令解析部
(3) 演算実行部

これらの詳細を次に述べる。また、全対構成を図1に示す。

(1) プロセッサ情報操作部

現在の実行タスクのコンテキストを操作する。この部分では、カーネル外に置かれる演算のメインモジュールに、カーネル内部の情報を送り、コンテキスト情報の変更を行う。このルーチンは使用環境に応じて、変更されない部分になる。

(2) 命令解析部

(2)、(3) はマルチタスク環境下での使用を想定し、この二つで、一つのタスクとしてプロセッサ情報操作部から起動される。ここでは、FPU 不在トラップを発生した実行コードを解析して演算の実行に必要な数値をメモリ中から取得したり、結果の格納先のアドレスを算出する。ここの解析ルーチンを変更することで、Intel x87 以外のプロセッサモデル(例えばレジスタ形式)のエミュレートも可能になる。

The Implemation of Floating Point Processing Emulator Supported Plural Model for OS／omicron V4
Yasuhiro YAMAMOTO, Eiichi HAYAKAWA, Mitarou NAMIKI and Nobumasa TAKAHASHI
Tokyo University of Agriculture and Technology

(3) 演算実行部

命令解析部から実行する演算を受け取って、表現方法に応じて演算を行う。(2)、(3) はダイナミックリンカにより、各表現方法、プロセッサモデルに応じたモジュールがリンクされて、システムを構築する。

また、コンテキストスイッチ部を、エミュレータ、ハード使用時の両方に対応したものにした。これは、プロセッサモデルの変更により格納するデータが変わるので、ダイナミックリンクにより、カーネル構成を変更する。

5. CAT386 の変更

複数の浮動小数点演算をサポートするために、言語処理系の CAT386 の変更を行った。CAT386 本体もダイナミックリンクを使用することで、再構築の手間を省く。このため、表現方式が変わったときに、リンクする必要がある部分を CAT386 本体と分離した。分離した所は次の二つに分けられる。また、全対構成を図 2 に示す。

(1) 定数生成部

各浮動小数点表現方式では、同じ数値を表すにしても、内部のビットパタンが異なってくる。このため、各表現方式に応じたビットパタンを生成する。

(2) 実行コード生成部

プロセッサモデルに応じたコードを生成する。このモデルを変更し、エミュレータの命令解析部を変更することで、仮想的なプロセッサのテストなどを行える。

6. 浮動小数点エミュレータ、CAT386 の実現

現在、浮動小数点エミュレータを OS／omicron V4 上に実現し、ダイナミックリンクされて動作している。また、CAT386 を Intel x86 系上で実現した。CAT386 が生成する浮動小数点表現形式は IEEE 方式で、演算エミュレータは Intel 80387 以降に準拠したモデルになっている。

表 1 にエミュレート時の四則演算、実数のロード、ストアの実行速度を示す。計測は Pentium 90MHz 上で行った。

実ハードウェアと比較して、エミュレートの速度が非常に遅いのは、フォールト割込みを使用していること、コンテキストスイッチが OS エミュレートの開始、終了に必ず発生していることに原因がある。

だが、目的から考えると、速度は重要視していないので、これは大きな問題ではないと考える。

表 1　演算のエミュレート速度

	エミュレート	実ハードウェア
数値のロード (32Bit実数)	132μs	33.3ns
数値のストア (32Bit実数)	144μs	77.8ns
加算・減算	161μs	111.1ns
乗算	178μs	177.8ns
除算	312μs	811.1ns

ロードはアキュムレータスタックに、ストアはアキュムレータスタックから。四則演算はスタックトップとその下のものの計算。

7. おわりに

本稿では、OS／omicron V4 上に実現した浮動小数点演算エミュレータのシステムについて述べた。ダイナミックリンクを使うことで、柔軟にシステムを構成できた。またフォールトを利用したエミュレータの実装は、専用ハードのエミュレーションにおいても有効であると考える。

今後は、URR や松井・伊理方式など、他の表現方式に対応したエミュレータ、コンパイラの構築を行う。

参考文献

[1] Hayakawa, et.al : "Basic Design of SHOSHI Operating system that Supports Handwriting Interface", 情報処理学会論文誌, Vol.35, No.12

[2] 中村浩之, 他 : "80386 用 OS／omicron 開発のための言語 C 処理系の実行環境の設計", 情報処理学会第 44 回全国大会, 2F−10, 1992.

[3] 森岳志, 他 : "OS／omicron における複数の浮動小数点方式のサポート", 情報処理学会第 33 回全国大会, pp.325−326, 1986.

[4] 浜田穂積, 他:"万能実数値表現法 URR と OS", 情報処理学会コンピュータ・システムシンポジウム論文集, Vol.87,No.2 (1987), pp.121−130

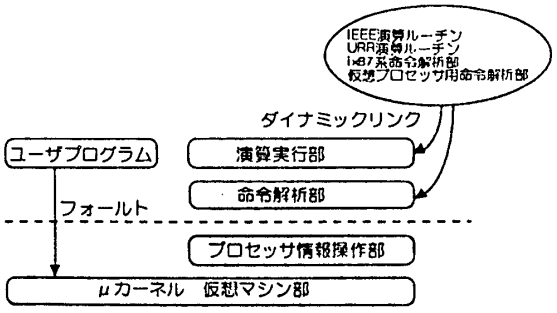

図 1　エミュレータの全体構成

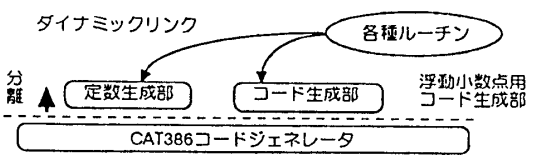

図 2　CAT386の全体構成