ICOT Technical Report: TR-493

TR-493

並列推論マシンPIMにおける抽象機械語KL1-Bの実装
—高級機械語を実装するための道具立—

山本　礼己、今井　明、中川　貴之、
川合　英夫、伸瀬　明彦(東芝)、中越　靖行、
宮崎　芳枝、堂前　慶之(富士通SSL)

July, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列推論マシンPIMにおける抽象機械語KL1-Bの実装
－高級機械語を実装するための道具立－

Implementation of processing system for KL1-B abstract machine language of PIM

–Tools and environments for the implementation of high level machine language–

山本礼己* 今井明 中川貴之 川合英夫

Reki YAMAMOTO Akira IMAI Takayuki NAKAGAWA Hideo KAWAI

(財)新世代コンピュータ技術開発機構†

Institute for New Generation Computer Technology (ICOT)

仲瀬明彦

Akihiko NAKASE

(株)東芝

TOSHIBA Corporation

中越靖行 宮崎芳枝 堂前慶之

Yasuyuki NAKAGOSHI Yoshie MIYAZAKI Yoshiyuki DOUMAE

(株)富士通ソーシアルサイエンスラボラトリ

FUJITSU Social Science Laboratory Ltd.

概要

現在開発中の並列推論マシンPIMは抽象機械語KL1-Bコードを並列実行する推論マシンである。KL1-Bは並列推論機能に加え、OS機能をサポートし、WAMなどと比べると、高機能な抽象機械語である。

このKL1-Bの並列実行系をPIMハードウェアに実装するため、KL1-B実行系の仕様を記述する言語PSLを設計し、PSLで記述したKL1-B実行系の仕様VPIM(Virtual PIM)を開発している。

VPIMは仕様書であると同時に、これをコンパイルするとPIM実機のファームウェアプログラムとなることを狙っている。また、PSLをC言語に変換することにより、VPIMは汎用計算機上でシミュレータとしても動作する。

既に、PSLの言語仕様、PSLによるプログラムの開発環境、シミュレータ用のPSL言語処理系が出来上がり、PSLによるKL1-B並列実行系の仕様の記述も進んでいる。現在、このKL1-B実行系の仕様の一部はPSL言語処理系によりコンパイルされ、シミュレータとして動作しており、KL1-B並列実行方式の検証に入っている。

今回は、PIM開発の道具立てとして、PSLの言語仕様及びPSLプログラミング環境について報告する。

1 はじめに

ICOTでは並列推論マシンPIM[2]の開発にあたり、そのプロトタイプマシンとしてMulti-PSI/V2[4]を開発している。Multi-PSI/V2が共有メモリを持たない疎結合系であったのに対し、PIMでは共有メモリを持った密結合系が更にネットワークを介して結合している点で異なっているが、どちらも各要素プロセッサが抽象機械語KL1-Bを実行する、並列推論マシンである。

KL1-Bは並列論理型言語KL1[3]実行のための抽象機械語で、並列推論機能に加え、OS機能の一部をサポートし、WAM[1]などと比べると、高機能な抽象機械語である。このため、KL1-B実行系の設計にあたって、その仕

*JUNET:yamamoto@icot.junet

†108 東京都 港区 三田1丁目 4-28 三田国際ビルヂング21F, Phone: 03(456)3193

様の記述を高級言語で書くことが開発の効率を上げると考えた。

Multi-PSI/V2 ではKL1-B実行方式をはじめC言語で設計し、それをMulti-PSI/V2のファームウェアに書き直していが、C言語の仕様とMulti-PSI/V2のファームウェア命令の仕様の違いから、直接変換することは難しかった。

そこでPIMの開発にあたっては、C言語程度の記述性を保ちながら、PIM実機のファームウェアレベルのアーキテクチャに親和性のある、KL1-B実行系仕様記述言語PSL(PIM Specification descriptive Language)を開発することにした。また、PSLで記述したKL1-B実行系の仕様をVPIM(Virtual PIM)と呼ぶ。

この両者を合わせて、PSL/VPIM システムと称し

- KL1-B実行系仕様記述言語 PSL(PIM Specification descriptive Language)
- PSL プログラム開発環境
- PSL で記述された KL1-B実行系仕様 VPIM(Virtual PIM)
- 汎用計算機上のシミュレータ PIM/s

なる項目を開発している。

PSLによるVPIMの記述には、次の様に大別して三つの目的が有り、PSLの仕様はそれを満たすべく設計された。(図1参照)

(1)PIM 仕様書

PSLによる記述がそのまま、あるいは、適当な(機械的な)編集の下に、PIMの機能仕様、実現方式などの仕様書として読めることを狙っている。このため、PSLの言語仕様は、書きやすさ、読みやすさを目指し、ファームウェアレベルの機能を記述する計算機言語としては高級な仕様になっている。また、PSLで記述するPIMは、その実現方式を種々試みる事が目的であり、『書き直しやすさ』も要求される。更に、実験内容に即して、並列キャッシュ制御、クラスタ内PE間通信、クラスタ間通信などの機能を基本機能として持つ必要が有る。

(2) 実機ファームウェアレベルのソースプログラム

(1)の仕様書をコンパイルする事により、それがそのままPIM実機の上で走行することを目指し、PSLの言語仕様は実機のアーキテクチャを考慮している。具体的には、スタックを用いたサブルーチンコールの実現の難しさを考慮し、値を返さないマクロを定義することによってプログラミングを行う、マクロ展開型の言語とした。さらに、実機への効率的コード変換のために、PSLの基本マクロは、実機アーキテクチャ、すなわちタグアーキテクチャに対して、親和性を高くしている。

(3) シミュレータのソースプログラム

(1)の仕様書を既存の計算機言語に変換する事により、その言語をサポートしているシステム上で、PSLで記述したVPIMがシミュレータとして走行することを目指し、PSLの言語仕様は既存の言語(具体的にはC言語)への変換の容易さ大きく考慮している。また、シミュレータでは各種評価を行うので、そのための情報抽出機能、情報編集機能などを効率的に実現する必要が有る。このため、既存の言語(つまりC言語)による記述を、シミュレータ用にPSLプログラム中に記述できる機能を追加した。

VPIMを変換してできたシミュレータをPIM/sと呼ぶ。PIM/sに関しては『4. PIM/s』で述べる。

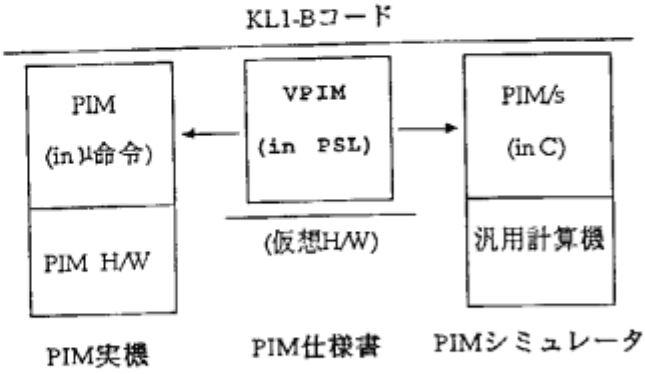


図1 PSL/VPIMの目的

2 PSLの構成

PSLはタグアーキテクチャ向きのマクロ言語である。PSLの主たる構成要素は、マクロ定義の記法と基本マクロのライブラリである。基本マクロの主要操作対象はタグ付きワードの集合からなる二つのデータクラス、レジスタ及びメモリである。

基本マクロとその操作対象を合わせて、下位レベルの仮想ハードウェアと考えることが出来る。この下位レベル仮想ハードウェアは、メモリを共有する複数のPEから成るクラスタが、更に複数個集まってネットワークを構成し、階層構造を成す。詳細は『4. PIM/s』で述べる。

PSLで書かれたプログラムはマクロ展開により、基本マクロ、レジスタ名および定数に展開される。基本マクロには汎用のマイクロプロセッサなどにみられる算術演算の他に、PSLの特徴として、タグによる条件分岐や、或いは並列キェッシュ制御付きメモリ操作などのマクロが有る。

本章では、マクロの操作対象およびマクロの記法を説明する。また、付録にコーディング例を示した。

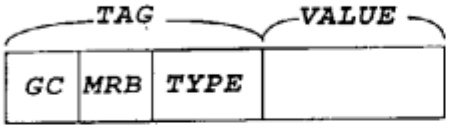
2.1 PSLマクロの操作対象

レジスタは各PE(Processing Element)毎にローカルに存在し、名前により参照される。基本マクロでサポートする算術、比較などの演算の引数は全てレジスタか即値である。

一方、メモリはクラスタ内の全PEにより共有されていて、アドレスにより参照される。PIMは、共有メモリのためのキャッシング機構[6]を持っており、これを制御する細かい記述を可能にするため、基本マクロとし

て、キャッシュ制御付きメモリアクセス機能をサポートする。

タグは特に並列推論マシンPIMの仕様を記述するために特殊化してあり、typeフィールド、MRB(Multiple Reference Bit)[7]フィールド及びGCフィールドから成る。



PSLでは更に、PIMがKL1-B命令の実行系であるという観点から、KL1-B命令の引数を参照する仕組みとして、オペランドフィールドなるデータクラスが有る。これをアクセスするための専用マクロをサポートする。

また更に、特殊レジスタとして、演算結果のテストのためのCCR(Condition Code Register)、それぞれのPEの状態を示すPESR(Processing Element Status Register)、プロセッサ要素間のシグナル機能をサポートするSCR(Slit Check Register)[5]などをサポートする。これらのレジスタは、基本マクロの暗黙の引数としてのみ現れる。

2.2 PSLマクロの記法

2.2.1 識別子

ユーザが定義する名前である。どのようなものに名前を付け得るのかについては『2.2.4 識別子の種類』で述べる。

識別子は英字（a-z、A-Z）数字（0-9）及び下線（_）から構成される（但し、一文字目は英字または下線である）。名前の長さは限定しない。

2.2.2 注釈

注釈は /* で始まり、 */ で終わる。入れ子の注釈はサポートしない。

2.2.3 定数

10進表記の符号付き整数、および16進表記の符号無し整数をサポートする。

2.2.4 識別子の種類

PSLでサポートする識別子にはシステムの予約語を除いて、以下の種類が有る。識別子の定義方法は『2.2.7 マクロの定義』で述べる。

(1) 定数マクロ
　定数に付ける名前
(2) データマクロ
　レジスタに付ける名前
(3) オペランドフィールドマクロ
　KL1-B命令のオペランドフィールドに付ける名前
(4) 操作マクロ
　一般操作に付ける名前
(5) 条件分岐マクロ
　条件分岐機能に付ける名前
(6) ケース群マクロ
　switch文のケース群に付ける名前
(7) goto文ラベル
　goto文の飛び先ラベルに付ける名前

2.2.5 式

PSLではいわゆる式（expression）というものはサポートしていない。全ての演算（代入も含めて）は、操作マクロの形で提供される。

2.2.6 プログラム

マクロ定義の集合がPSLプログラムとなる。プログラム中における定義の出現順は問わない。複数ファイルの分割コンパイルも可能とする。

2.2.7 マクロ定義

以下に構文の概略を述べる。

(1) 定数マクロ定義 :
"#CONST_define"　マクロ名　定数

(2) データマクロ定義 :
"#DATA_define"　レジスタ名

(3) オペランドフィールドマクロ定義 :
"#OPF_define"　KL1-B命令オペランドフィールド名

(4) 操作マクロ定義 :
```
"#PSL_define" 操作マクロ名"("
                        [ マクロ引数列 ]")"
  "{"
    [ 文... ]
  "}"
```

(5) 条件分岐マクロ定義 :
```
"#CTRL_define" 条件分岐マクロ名"("
                        [ マクロ引数列 ]")"
  "{"
    [ 文... ]
    条件分岐マクロ参照
  "}"
```

(6) ケース群マクロ定義 :
```
"#CASE_define"　ケース群マクロ名"()"
  "{"
            ケースマクロ参照 [ ":"
    "case"  ケースマクロ参照 ] ...
  "}"
```

2.2.8 文

(1) 操作マクロ参照 :
　マクロ名"(" [マクロ引数列] ");"

　#PSL_defineされたマクロの参照。

(2) 複文 :
　"{" 文... "}"

文の集まりを一つの文と認めるための構文。

(3) ラベル付き文 :
ラベル識別子 ":" 文

goto 文による分岐先。

(4) 制御文 :
条件分岐マクロ参照 "goto" ラベル識別子 ";"

条件分岐。

| 条件分岐マクロ参照 複文

条件成立時 『複文』 を実行。

| 条件分岐マクロ参照 複文 "else" 複文

条件成立時『else』の前の『複文』を、
不成立時『else』の後の 『複文』を実行する。

| "TypeSwitch(" レジスタ名 "){"
{ "case" ケースマクロ参照 ":" 文... }...
["default:" 文...]
"}"

指定レジスタの Type フィールドを
『ケースマクロ』で指定された定数
または定数群と比較し、分岐。

| "ValueSwitch(" レジスタ名 "){"
{ "case" ケースマクロ参照 ":" 文... }...
["default:" 文...]
"}"

指定レジスタの Value フィールドを
『ケースマクロ』で指定された定数
または定数群と比較し、分岐。

| "LOOP" 複文

無条件無限ループ、脱出は『"break;"』による。

| "goto" ラベル識別子 ";"

無条件分岐。

| "break;"

TypeSwitch 文、 ValueSwitch 文、
LOOP 文からの脱出。

(5) 関連記号
以下には (1)-(4) で用いられた関連記号の定義を記す。

条件分岐マクロ参照 :
マクロ名 "(" [マクロ引数列] ")"

#CTRL_define された条件分岐マクロの参照。

ケースマクロ参照 :
定数マクロ参照 | ケース群マクロ参照

定数マクロ参照 :
マクロ名

ケース群マクロ参照 :
マクロ名 "()"

3 PSL コンパイラ

PSL による VPIM の記述の目的は、(1) 仕様書、(2) 実機ファームソース、(3) シミュレータソースの 3 点であるが、この内 (2) および (3) を実現するために PSL コンパイラを開発している。

この二つの目的を達成するため、その構成は共通のマクロ展開部と、マシン依存のコード生成部とから成る。

現在、共通のマクロ展開部と、シミュレータ向けコード (C 言語ソース) 生成部が完成しており、更に実機向けコード生成部を開発中である。(図 2 参照)

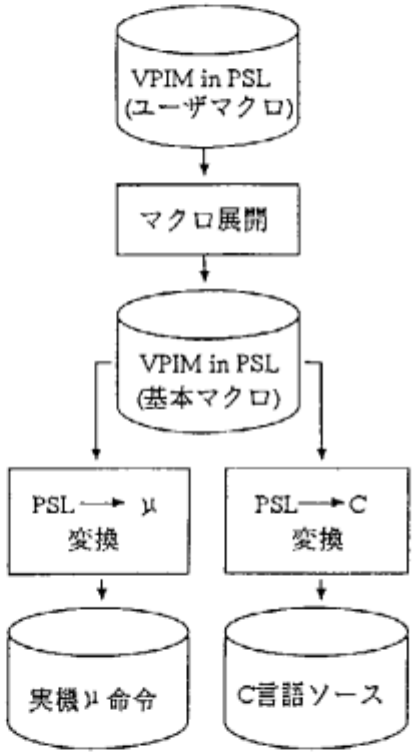


図2 PSL言語処理系

4 PIM/s

4.1 目的

PIM の研究開発では、論理型言語の並列実行方式を各種実験評価していく。この評価は実機で行う前に、先ずシミュレータで行うべきである。このため PSL で記述した PIM の仕様 VPIM(Virtual PIM) を、汎用計算機用にコンパイルして、シミュレータとして動作させることにした。また、VPIM 自身のテスト、デバッグも必要である。これらの目的のため、汎用計算機上のシミュレー

ティングシステムとして、PIM/s(PIMシミュレータ)を開発している。

4.2 構成

並列推論マシンPIMは、複数の推論エンジンを並列に稼働させるものである。一方、世の中には既に、汎用マイクロプロセッサを複数台並列に稼働させる汎用並列処理システムが有る。そこで、この汎用並列処理システムこの上で、並列推論マシンのシミュレートを行う事にした。

PIM/sは、PSLで記述されたVPIMを、汎用並列処理システムがサポートする言語に変換したものを、コンパイルして実行動作させるものである。このVPIMは個々のKL1-B命令の機能をを、仮想的な下位ハードウェアの基本操作である、PSLの基本マクロを組み合わせて記述したものである。しかし、PIMの実ハードウェアに相当する部分の記述はVPIMの中には無いので、汎用並列処理システム専用に開発する必要が有る。すなわちPIM/s実現のためには、PIM/sの構成要素として以下のものが必要になる。

(0) VPIM
(1) 下位レベル仮想ハードウェア
(2) KL1-B命令コードディスパッチャ
(3) KL1-Bプログラムローダ
(4) KL1クロスコンパイラ、アセンブラ、リンカ

4.3 下位レベル仮想ハードウェアとその実現

ここでは、下位レベル仮想ハードウェアを汎用並列処理システム上に実現する方式を説明する。

下位レベル仮想ハードウェアは、メモリを共有する複数のPEから成るクラスタが、更に複数個集まってネットワークを構成し、階層構造を成す。下位レベル仮想ハードウェアは以下の構成要素から成る。

4.3.1 タグ付き演算装置

各PEに有ってタグ付き演算を行う。

PIMはタグアーキテクチャの推論マシンである。一方、汎用並列処理システムはタグをサポートしていない。よって、タグ付きの演算処理は、tagフィールドとvalueフィールドに分解してシミュレートする。コンディションコードレジスタなど、演算の暗黙の引数となる特殊レジスタへの操作をもシミュレートする。

4.3.2 レジスタ

各PEに固有の記憶領域である。汎用レジスタと特殊レジスタが有る。

汎用レジスタはタグ付きwordの配列と考える。汎用並列処理システムではタグ付きのデータ型をサポートしていないので、構造体データとして定義し、タグ付きwordをシミュレートする。汎用レジスタはPE毎のローカルな記憶領域なので、汎用並列処理システムでは、privateな変数として定義する。

特殊レジスタには、演算結果に関する情報を保持するコンディションコードレジスタ(CCR)、PEの状態を保持するプロセッシングエレメントステータスレジスタ(PESR)、例外要因を保持するスリットチェックレジスタ(SCR)などが有る。CCR、PESRはPE毎のローカルなレジスタであり、privateな変数とするが、SCRはPE間の信号の遣り取りを媒介するものであり、各PE毎に存在するが、クラスタ間に渡って操作できる必要があり、sharedな変数とする。

4.3.3 共有メモリ

クラスタ内のPEに共有される記憶領域である。タグ付きword単位のロック機構を持つ。

共有メモリもタグ付きwordの配列と考える。共有メモリはクラスタ内のPEの間で共有される記憶領域なので、汎用並列処理システムでは、sharedな変数として定義する。共有メモリの排他制御は、汎用並列処理システムの持つメモリロック機能を用いる。メモリの並列キャッシュ機能の評価を行うため、キャッシュコントローラのシミュレートも行う。

4.3.4 ネットワーク

ネットワークはクラスタ間のメッセージを交換するルートである。汎用並列処理システムでは、同システムの持つプロセス間通信機能によりこれをシミュレートする。

5 PSLプログラミング環境

現在PSIによるプログラム開発環境はUNIX上に作成している。以下には、各種目的のために開発したツール群の概要を解説する。

5.1 設計、コーディング支援

- 階層化設計

  PSLによるVPIMのコーディングはそれ自身が仕様書となることを狙っている。このためソースを階層毎のディレクトリに分けて置き、またマクロ名には階層を示すプレフィクスを付け、さらに階層関係に矛盾する呼び出しを自動的に検出するツールを作った。

- コメント入力

  PSLコーディングによる仕様書を補う意味で、形式を統一したコメントを積極的に入れるべくコメントフィールド操作機能をエディタから呼び出せるようにした。

- 構文チェック

  エディタ中から随時PSL構文をチェック出来るようにした。

5.2 テスト、デバッグ、評価支援

- クロスリファレンス

  シンボリックデバッグの便利のため、PSLプログラムのクロスリファレンスをデータベースとして作成するツールを作成した。このツールは定期的、自動的に実行され、データベースを最新のものに保つようにした。

- Cプログラミング

  実行時の各種情報抽出のため、C言語によるライブラリを開発した。シミュレータPIM/sはVPIMを一度C言語に変換した後、実験開発システム用にコンパイルされるので、この時リンクすることが出来る。

6 まとめ

PSLによるVPIMの開発は、'88年の春にその構想が纏まり、言語仕様の設計に入ったのは同年の夏であった。秋にはほぼ言語仕様が固まり、'88年中はテストコーディングを行った。'89年から本格的コーディングに入り、既にコメント込みで50Kステップを開発している。

シミュレータ用のPSL言語処理系も出来上がっており、現在はシミュレータとして動作させてVPIMのデバッグを行っている。

PSLの目的の7割は既に達成されていると考えている。

現在は、PIM実機のハードウェアの開発も進んでおり、これに合わせて実機用のPSL言語処理系の開発、マクロ間共通コードのサブルーチン化の検討などを急いでいる。

またシミュレータ上ではVPIMの実現方式における各種実験評価を行う予定であり、評価ツールの作成、評価ツールとPSL言語処理系間のインタフェース設計などを進めている。

7 謝辞

貴重な御意見を頂いたICOTのMulti-PSI及びPIMOSグループの方々に感謝致します。また、本研究の機会を与えて頂いた、ICOT渕所長、内田第4研究室長に感謝致します。

参考文献

[1] D.H.D.Warren. An Abstract Prolog Instruction Set. Technical Note 309, SRI International, 1983.

[2] A. Goto, M. Sato, K. Nakajima, K. Taki, and A. Matsumoto. Overview of the Parallel Inference Machine Architecture (PIM). In *Proc. of the International Conference On Fifth Generation Computing Systems 1988*, Tokyo, Japan, November 1988.

[3] T. Miyazaki. Parallel Logic Programming Language KL1 – Its Implementation and an Operating System in It –. *Transactions of the Institute of Electronics Information and Communication Engineers*, J71-D(8):1423–1432, August 1988. (In Japanese).

[4] K. Taki. The parallel software research and development tool : Multi-PSI system. In *France-Japan Artificial Intelligence and Computer Science Symposium 86*, pages 365–381, October 1986.

[5] 中川貴之, 後藤厚宏, 近山隆. プロセッサ間ソフトウェア割り込み処理を高速化する スリットチェック機構. 計算機アーキテクチャ研究会, July 1989.

[6] 松本明, 中川貴之, 佐藤正俊, 木村康則, 西田健次, 後藤厚宏. KL1のメモリ参照特性に適した並列キャッシュ機構. 第2回データフローワークショップ予稿集, pages 223–230, Oct. 1987.

[7] 木村康則, 西田健次, 宮内信仁, 近山隆. KL1の多重参照ビットによるGC方式について. データフローワークショップ *1987*, pages 215–222, Oct. 1987.

付録 –PSLコーディング例

```
#PSL_define f_PassiveDeref(reg, work)
{
  LOOP{
    s_IfREF(reg){
      s_DerefReg(reg, work);
      s_IfNotUnbound(reg){
s_IfMrbOff(reg){
  s_ReclaimVariable(work);
};
      } else break;
    } else break;
  }
}

#PSL_define s_DerefReg(data_reg, save_reg)
{
  @DEBUG {
    s_IfArgTypeErr (data_reg) {
      VPIM_ERROR ("s_DerefReg", "Illegal_Goal_Argument_Type");
    }
  }
  p_MoveWord (data_reg, save_reg);
  s_ReadWithMRBor(data_reg, data_reg);
  @DEBUG {
    s_IfKL1TypeErr (data_reg) {
      preg(save_reg); preg(data_reg);
      VPIM_ERROR ("s_DerefReg", "Illegal_KL1_Type");
    }
  }
}
```