KL1 の RISC 風 PIM/i 命令への展開について

佐藤正俊*、武田浩一、加藤研児
masatosi@okilab.oki.co.jp
沖電気工業株式会社

1 はじめに

我々は、第五世代コンピュータ・プロジェクトの一環として並列推論マシン PIM/i[1] の研究開発を行っている。PIM/i の対象言語は、並行論理型言語 KL1 であり、一般には、KL1 のような記号処理言語の処理方式は、3 つの方式（1] インタプリタ方式：言語を内部コードとして直接実行する方式、2] 仮想マシン方式：想定した仮想マシンの機械語を中間コードとし実行する方式、3] ネイティブコード方式：対象とするマシンの機械語まで展開し実行する方式）に分類される [2]。

我々は、PIM/i プロセッサの RISC 型や LIW 型の特徴 [3] を活かした最適化を考慮し、KL1 の処理方式としてネイティブコード方式（KL1 を PIM/i の機械語まで展開する方式）を選択している。ネイティブコード方式は、対象とするマシンの機械語に合わせた最適化が最も効果的に行える方式であると言える。しかし、ネイティブコード方式での課題は、単純な機械語の生成では静的なコードサイズが爆発的に多くなる点であり、これは、実行時においてキャッシュのヒット率の低下や共有バスの負荷の増大を引き起こす。また、これらを避けるために処理をサブルーチン化する場合、パラメータ渡しのオーバヘッド等の課題が予想される。

本稿では、PIM/i でのネイティブコード方式による KL1 の展開について、静的なコードと実行時の様子を考察したので報告する。

2 PIM/i 機械語への展開

2.1 PIM/i のハードウェア

PIM/i の構成は、複数クラスタから成り、1 クラスタは 8 台のプロセッシングエレメントが共有バスを介して共有メモリに接続される。プロセッシングエレメントは共有メモリの他にローカルメモリ（コード及びデータ）を持ち、頻繁にアクセスするコードやデータはローカルメモリに配置できる。それぞれの大きさは、ローカルコードメモリが 32 KW、ローカルデータメモリが 16 KW である。

PIM/i の機械語命令セットは、RISC 型命令セットを基本とし、タグの支援を行うとともに、パイプライン処理と複数操作の同時実行 (LIW) によって、コンパイラによる最適化の可能性を広げている。

*Native PIM/i Code Profiles using KL1 Programs. Masatoshi SATO, Koichi TAKEDA, Kenji KATO
Oki Electric Industry Co., Ltd.

表 1: 1 KL1b 当たりの命令数とサブルーチン呼び出し数

	KL1b 数	KL1b 当たりの命令数	KL1b 当たりのコール数
hanoi	117	6.2	0.5
prim	129	5.5	0.6
qk	255	5.5	0.5
qu	411	5.7	0.5
平均	228	5.7	0.5

2.2 ＫＬ１処理方式

ＫＬ１処理方式はネイティブコード方式であり、PIM/i の機械語への変換の手順 [4] は、KL1 プログラムを抽象マシン命令である KL1b にコンパイルすることから始める (KL1 コンパイラ)。次に、各 KL1b に対応する PIM/i の機械語へ変換する (KL1b ポストコンパイラ)。ここでの変換は、KL1b 毎に定義された PIM/i 機械語へマクロ定義を基に、マクロ展開することで実現している (マクロエキスパンダ)。一方、処理単位の大きなものは、ランタイムライブラリとしてローカルメモリ (LM) に置き、KL1b 毎に定義された PIM/i 機械語 (テンプレートと呼ぶ) は、このランタイムライブラリを呼び出すことでコードサイズを小さく抑える。

3 考察

3.1 ベンチマーク

KL1 ベンチマークは、ハノイの塔 (hanoi)、素数生成問題 (prim)、およびクイーン問題 (qk、qu) である。ベンチマークの実行にあたっては、このコードの他に、ランタイムサブルーチンを必要とする。

3.2 静的展開特性

図 1は、1 KL1b 命令がいくつの機械命令に展開されるかを示したものである。横軸は KL1b 命令が展開された命令数を表し、縦軸はベンチマーク中の KL1b 命令の出現数を示している。これによれば、2 命令に展開された KL1b 命令が多く、さらに KL1b 命令の約 56% が 4 命令（コードキャッシュのラインサイズ）以下に展開されている。

表 1は、各ベンチマークに対する静的なコードに対する KL1b 数、展開後の 1 KL1b 当たりの命令数、1 KL1b 当たりの実行時サブルーチンコール数を示している。1

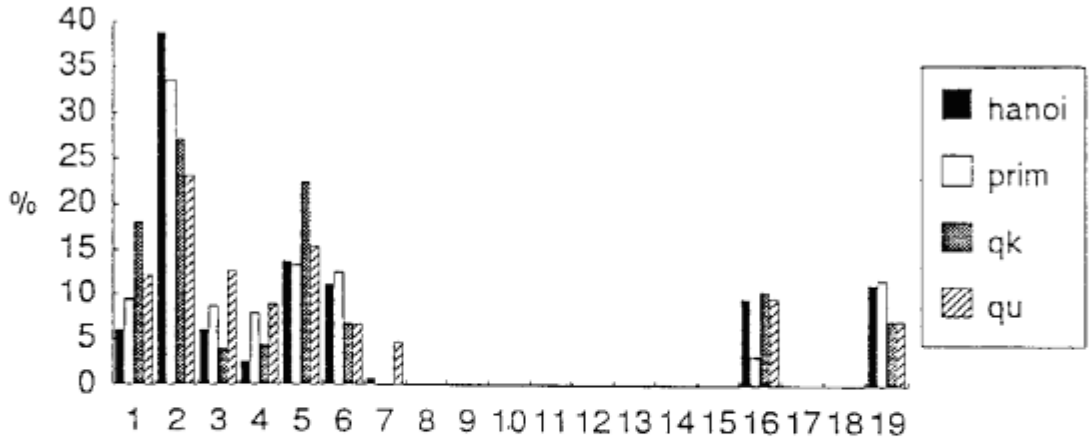

図 1: 1KL1b 当たりの展開命令数の分布

表 2: 1 KL1b 当たりの実行命令語数

	実行 KL1b 数	KL1b 当たりの実行命令数	KL1b 当たりの呼び出し回数
hanoi	2.5M	11.8(9.2/2.6)	0.29(0.20/0.10)
prim	2.7M	9.4(7.0/2.4)	0.28(0.23/0.06)
qk	12.7M	4.7(2.5/2.2)	0.24(0.11/0.13)
qu	26.0M	5.1(2.2/3.0)	0.25(0.12/0.13)
平均	11.0M	7.8(5.2/2.6)	0.27(0.17/0.11)

(l/g) は、ランタイムライブラリ (l) と展開命令 (g) の内訳

KL1b 命令当たりの展開数の平均は約 6 命令であり、図 1より約 80% の KL1b 命令が 6 命令以下に展開された。また、1 KL1b 命令当たりに含まれているサブルーチン呼び出し数は約 0.5 であり、半数近くがサブルーチン呼び出しを行なわず直接機械命令に変換されている。

KL1b 命令が 6 倍の機械命令に変換されるのは、WAM コードの場合 (10 ～ 20 倍 [5]) に比べると少ない。これは、KL1b が WAM コードに比べると機能的に低いことと、ランタイムサブルーチンの呼び出しを行っていることによると考えられる。このことから、KL1 コンパイラは、KL1b 命令のうち単純なものを多く生成すると言える。

3.3 動的実行特性

表 2は、ベンチマーク実行時の実行 KL1b 数と 1 KL1b 当たりの実行命令数及びサブルーチンの呼び出し回数を示している。ここで、実行 KL1b 数はメガ回で示し、1 KL1b 当たりの実行命令数及びサブルーチンの呼び出し回数においては、() 内に、ランタイムライブラリと展開命令との内訳を示している。表において 1 KL1b 当たりの実行命令数の平均は、8 命令程度であるが、これは hanoi や prim が GC の影響を受け実行命令数が多くなっているためで、GC の実行命令数を除けば 5 命令程度であった。また、1 KL1b 当たりのサブルーチンの呼び出し回数は、平均で 0.27 であり、静的なコードの分析の結果 (0.5) より少なかった。これは、KL1b に展開されたサブルーチン呼び出し命令の 1 ／ 3 が実際には呼び出されなかったことを意味する。

ランタイムライブラリと展開命令との内訳は、実行命令数及びサブルーチンの呼び出し回数においても、ほぼ 1 : 1 であった。(hanoi、prim のランタイムライブラリが多いのは GC のためである。) これは、ランタイムライブラリをローカルメモリ (LM) に、展開命令をグローバルメモリ (GM) に配置しているので、コードアクセス比を示している。

4 おわりに

ネイティブコード方式による静的コードサイズと実行命令語数について、PIM/i での展開の様子を考察した。これによると、1 KL1b 当たり 6 命令に展開され、爆発的なコードサイズの増大は避けることができた。また、サブルーチン呼び出しは、1 KL1b 当たり 1/3 程度で、それほどオーバヘッドになっていないと言える。

日頃、助言をいただく (財) 新世代コンピュータ技術開発機構 (ICOT) 第 1 研究室、および沖電気の PIM 担当諸氏に感謝する。

参考文献

[1] 大原他: 並列推論マシン PIM/i の概要、情報処理学会第 40 回全国大会予稿集、1990

[2] 幅田他: 記号処理言語プロセサ O l i v e のアーキテクチャ評価、情報処理学会研究報告 90-SYM-56-6

[3] 武田他: 並列推論マシン PIM/i の拡張プロセッサのアーキテクチャ、情報処理学会第 40 回全国大会予稿集、1990

[4] 佐藤他: 並列推論マシン PIM/i の開発支援環境 - 翻訳系 -、情報処理学会第 41 回全国大会予稿集、1990

[5] G.Borriello et al. Special or General-Purpose Hardware for Prolog: A Comparison. TR UCB/CSD 87/314, Computer Science Division, UCB, 1986.