1X-3

ベンチマークプログラムによるＣＰＵアーキテクチャの性能評価の考察

野尻 徹、海永 正博
（株）日立製作所 システム開発研究所

１．緒言

Dhrystone[1]は、コンパイラ、メモリシステム、ＣＰＵ等を含めたシステムの性能評価を行うために作成されたベンチマークプログラムである。近年このベンチマークプログラムをＣＰＵ単体の性能評価の指標として用いられるようになってきた。

本発表では、最近のコンパイラの最適化技術を考慮して、Dhrystoneを指標として用いたＣＰＵ単体の性能評価に対して考察した結果を報告する。

２．本論

最近のコンパイラ[2]ではその最適化処理を、中間語レベルでのプログラミング言語よりのものと、マシン語レベルでのＣＰＵアーキテクチャよりのものとに分けている。プログラミング言語よりの最適化処理としては、ループ不変式のコードの移動、共通部分式の削除、値の伝播、デッド・コードの削除等が挙げられる。ところがこれらの手法を用いると、Dhrystoneが持つ本来の主旨とは異なる、つまり整数演算主体のシステムプログラムにおける各種文、式、変数、型等の様々な統計データを基に算出された出現頻度とは別の分布に従うオブジェクトコードが生成される。

そこで、最適化処理を行なった場合と行なわない場合とに関して、Ｃ言語で書かれたDhrystone 1.1から三種類のＣＰＵ（ＣＩＳＣ２種、ＲＩＳＣ１種）のオブジェクトコードを生成した。その結果得られた、内部のループ１回の実行に対するオブジェクトコードの命令のステップ数と実行ステップ数を以下の表に示すここでプログラミング言語よりの最適化処理を行なった際に、Dhrystone本来の各種出現頻度分布を変移させるソースプログラム上の要因のうち主なものを以下に列挙する。

（１）デッド・コードの削除
- Proc0：IntLoc２への二つの代入文
- Proc4：Bool1Locへの二つの代入文
- Func2：CharLoc（０もしくは'Ａ'）によるｉｆ文
- Func2：IntLocへの代入文

（２）共通部分式の削除
- Proc1：PtrParIn->IntCompの連続したアクセス

（３）値の伝播
- Func1：ｉｆ文中のCharLoc2（＝CharPar1）
- Func3：ｉｆ文中のEnumLoc（＝EnumParIn）

３．考察

総じて、ＲＩＳＣはＣＩＳＣに比べ静的命令ステップ数、および実行命令ステップ数共に最適化処理による減少の割合が高い。これは、最適化処理を行なわない場合ナイーブなコード生成を行なうために、ディレイドブランチに関する物も含めて、ピープホール最適化処理の効果がより大きく現われたためである。

A Study of Performance Evaluation of CPU Architecture with Benchmark Programs,
Tohru Nojiri, Masahiro Kainaga
Systems Development Laboratory, Hitachi, Ltd.

またＣＩＳＣ２はＣＩＳＣ１に比べて、命令ステップ数は小さい値を示している。その原因として、２オペランドに対して３オペランドの命令形式であるほかに、サブルーチンコール、構造体の代入等の高機能な命令を持つことが挙げられる。しかし各命令の実行サイクル数を考慮すると、必ずしも実行時間の短縮は達成されない。

一般にプログラミング言語よりの最適化処理は、中間語レベルでの処理において効果を発揮する。したがて、ソースコードでこの効果が現われることは、Dhrystoneが本来持つ意図に基ずいたＣＰＵ単体の性能評価に適しているとはいいがたい。コンパイラを含めたシステムの性能評価に用いることは意味があるという見方もあろう。しかし最適化コンパイラの評価を行なうには別の指標が必要となろう。

４．結言

以上述べてきたように、最適化コンパイラにより Dhrystone 1.1からコードを生成すると、プログラム言語機能の実際の出現頻度の分布からの変移が生じる。この対策として、Dhrystone 2.0[3]が作成された。

本稿では命令のステップ数のみに言及をとどめたが、性能評価の観点からは、各命令の実行サイクル数、パイプラインの効果、およびメモリアクセスの各々が実性能にどのように関与しているかが関心事となろう発表ではこれらを含めて、Dhrystone 2.0にもとづいた評価についても報告する予定である。

５．参考文献

(1) Reinhold P. Weicker, Dhrystone: A Synthetic Systems Programming Benchmark, Communication of the ACM, ACM, Vol.27, No.10(Oct. 1984), 1013-1030

(2) Steven S. Muchnick, et al., Optimizing Compilers for the SPARC Architecture: An Overview, Proc. of COMPCON Spring 88, IEEE, Mar. 1988, 284-293

(3) Reinhold P. Weicker, Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules, SIGPLAN NOTICES, ACM, Vol.23, No.8(Aug. 1988),49-62

ＤＨＲＹＳＴＯＮＥのオブジェクトにおける命令ステップ数
（実行ステップ数/静的ステップ数）

関数名	処理内容	ＣＩＳＣ１		ＣＩＳＣ２		ＲＩＳＣ	
		no-opt	opt	no-opt	opt	no-opt	opt
ｐｒｏｃ０	主関数	97/86	76/66	74/63	61/53	115/101	85/76
ｐｒｏｃ１	構造体	85/51	83/48	27/28	24/25	94/59	90/54
ｐｒｏｃ２	数え上げ	19/20	15/15	11/12	11/11	29/31	19/19
ｐｒｏｃ３	関数参照	15/17	15/17	11/12	10/11	28/31	19/21
ｐｒｏｃ４	フラグ	11/11	2/2	9/9	2/2	22/23	6/6
ｐｒｏｃ５	外部変数	5/5	3/3	4/4	3/3	12/12	8/8
ｐｒｏｃ６	スイッチ文	22/37	21/31	13/24	12/21	28/51	20/38
ｐｒｏｃ７	加算	33/11	24/8	9/3	9/3	51/17	18/6
ｐｒｏｃ８	配列	81/64	46/38	45/32	33/26	106/84	58/53
ｆｕｎｃ１	文字	33/15	24/9	24/10	12/6	51/21	36/12
ｆｕｎｃ２	文字列	42/50	30/32	27/35	21/23	45/53	33/30
ｆｕｎｃ３	数え上げ	12/14	8/9	6/8	4/4	13/17	6/8
ｓｔｒｃｐｙ	文字列複写	78/13	78/13	61/9	61/9	83/7	83/7
ｓｔｒｃｍｐ	文字列比較	88/19	88/19	75/16	75/16	123/7	123/7
合　計		621/413	513/310	396/265	338/213	800/514	604/345