RISCプロセッサ向け最適化機能とその評価 *

5D－7

中平直司　松山学　堀田耕一郎 †

富士通株式会社‡

1　はじめに

近年、１クロックサイクル１命令以上を実行するＲＩＳＣプロセッサの台頭が著しく，このアーキテクチャを持ったＲＩＳＣプロセッサがワークステーションの主流を占めている。これらのＲＩＳＣプロセッサは，コンパイラの最適化を前提に設計されているため，プロセッサが持つ資源を有効に活用する最適化技術が性能向上に非常に大きな役割を占める。前回，これらのＲＩＳＣプロセッサに適した最適化コンパイラの構成について論じ、高レベルな最適化と命令レベルの最適化の重要性[1]について述べた。本稿では、ＲＩＳＣプロセッサ向けの最適化機能について備えるべき要件及びその実装について述べる。また、実際にこれらの最適化を実装したＣ／ＦＯＲＴＲＡＮコンパイラの実行性能についても述べる。

2　ＲＩＳＣプロセッサ向けの最適化の要件

ＲＩＳＣプロセッサ向けの最適化機能としては、特に以下の要件が重要である。

(1) 最適化の段階的な適用

一度だけの最適化の適用では、ＲＩＳＣプロセッサで要求される高い実行性能を得ることは困難である。この要求を満たすためには、最適化を段階的に適用していくことが望ましい。特に、ＲＩＳＣプロセッサに要求されるきめ細かな最適化を十分に行なうことが重要である。

(2) 最適化の繰り返し

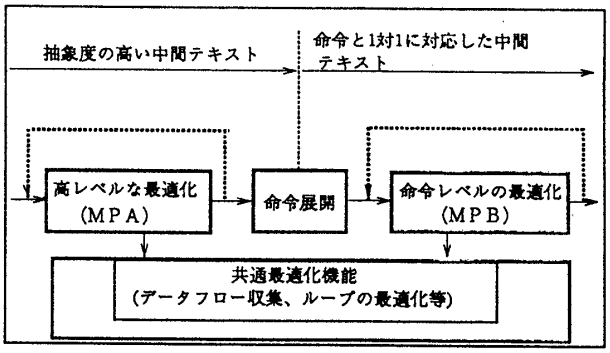
同じ最適化を繰り返し実施することにより、最適化が促進される場合がある。このため、最適化を必要に応じて繰り返し実施できることが必要である。特に、ＲＩＳＣプロセッサの性能劣化の原因となる分岐命令を削減する最適化を繰り返し実施できることは必須の機能である。

(3) 命令レベルの並列性の引き出し

ＶＬＩＷ／スーパスカラの並列ＲＩＳＣプロセッサが持つ複数の演算器を命令スケジューリングで有効に活用できるように、スケジューリング可能な命令数を増やす最適化や、データの依存関係を解消する最適化が重要である。

*Optimization functions for RISC Processors and Evaluation
†Tadashi NAKAHIRA,Manabu MATSUYAMA, Kohichiro HOTTA
‡FUJITSU LIMITED

図 1: 最適化の構成と中間テキストの関係



3　最適化機能の実装

ここでは、２節で述べた最適化の要件に対して、どのような最適化をコンパイラに実装したかを方法も含めて述べる。

3.1　最適化の段階的適用

最適化を段階的に適用するため、図１に示すような、広域最適化を含む高レベルな最適化（ＭＰＡ）と命令レベルの最適化（ＭＰＢ）が別々に動作する構成をとった。この最適化の構成に伴って、中間テキストについても、ソース情報を多く含む抽象度が高い中間テキストとハードウェア命令に１対１に対応した中間テキストに分離した。命令と１対１に対応する中間テキストへの変換は、命令展開部が行なう。

3.2　命令レベルの最適化

命令レベルの最適化を実施する目的は、中間テキストが命令に１対１に対応した段階で、もう一度最適化を実施することにより、ＲＩＳＣプロセッサの最適化に要求されるきめ細かな最適化を十分に行なうことにある。命令レベルの最適化では、基本的な最適化（共通式の除去、ループの最適化等）をＭＰＡと共有化する。その他に、広域的なメモリアクセスの削減（ストア／ロード命令の除去）、パラメータの最適化、命令展開によって生じる冗長な命令の除去、より高速な演算への置換といった、ハードウェアと密接に関連した最適化を実施する。

図 2: リネーミングとループリダクションを適用した例

```
Q=0
DO I=1,1000
    Q = Q + A(I)*B(I)
CONTINUE
    ⇓
Q1=Q2=Q3=Q4=0
DO I=1,1000,4
    Q1 = Q1 + A(I)*B(I)
    Q2 = Q2 + A(I+1)*B(I+1)
    Q3 = Q3 + A(I+2)*B(I+2)
    Q4 = Q4 + A(I+3)*B(I+3)
CONTINUE
Q=Q1+Q2+Q3+Q4
```

3.3 最適化の繰り返し

最適化を繰り返す機能を、ＭＰＡとＭＰＢのそれぞれの最適化処理部に持たせた。繰り返す回数は、翻訳速度への影響を考慮し、最適化のレベルに応じて決定した。

繰り返す最適化は、共通式の除去や複写伝搬等の基本的な最適化を中心にし、分岐の最適化等の制御構造の変更を伴う最適化も組み入れた。また、これらの最適化が使用するデータフロー情報やコントロールフロー情報を必要な時点でとり直すことができるように配慮した。

3.4 命令レベルの並列性を引き出す最適化

並列性を引き出す最適化を実装するに当り、ループ内の命令に対して並列性を引き出す最適化に重点を置いた。基本ブロックを拡張し、スケジューリング可能な命令数を増やす最適化では、ループアンローリングを、データの依存関係の解消では、変数のリネーミングやループリダクションを実装した。また、命令スケジューリングによる命令の移動の可否は、メモリアリアッシングの能力に依存するので、その解析能力も強化した。

図２にアンローリングを実施した後に，リネーミングおよびリダクションを適用した例を示す。この例では、本最適化を適用することにより、各演算の依存関係がなくなり、命令スケジューリングにより命令レベルの並列性が生かされるように並び換えることが可能となる。

4 最適化の効果

上記の最適化をＳＰＡＲＣアーキテクチャ向けＣ／ＦＯＲＴＲＡＮコンパイラに実装し、性能を評価した。性能評価プログラムには、ＳＰＥＣ９２とリバモアループを使用した。測定マシンは、並列ＲＩＳＣプロセッサを搭載するＳＳ１０と並列ＲＩＳＣプロセッサでないＳＳ２を用いた。表１、２にそれぞれの最適化を実施した時の効果を示す。

(1) 命令レベルの最適化の効果

CINT92、CFP92 共に大幅な性能向上が得られる。共にパラメータの最適化や命令展開によって生成されたメモリアクセス等の冗長な命令を削減できたことが、性能向上の主な要因である。この結果から、ハードウェア命令と１対１に対応した段階でもう一度最適化を実施することが非常に有効であることが分かる。

表 1: 各最適化の効果

マシン SS2 (クロック:40 MHz メモリ: 48Mbyte)

最適化機能	CINT92	CFP92
命令レベルの最適化	1.03-1.21	1.01-1.18
最適化の繰り返し	1.01-1.05	1.00-1.02

表 2: ループリダクションの効果

マシン SS10 (クロック: 36 MHz メモリ: 48Mbyte)

最適化機能	リバモア 14	リバモア 24
ループリダクション	1.11	1.06

(2) 最適化の繰り返しの効果

整数演算が主体の CINT92 では最高５％の性能向上が見られる。特に実行のコストが特定のループに集中することがないシステムコードでは、本最適化が有効であると言える。繰り返しの回数は、基本ブロックの命令数に変化がなくなるまで繰り返す条件で行なったが、ほとんどのプログラムは、３回までに収束しており、翻訳性能に与える影響は少ない。

(3) 並列性を引き出す最適化の効果

ループリダクションと変数のリネーミングより、リバモアループの性能が６％と１１％向上する。これらの最適化が効果を発揮するループに限定すると２５％から４０％程度の性能の向上が得られる。ループアンローリングだけを単純に行なうのではなく、これらの最適化を組合わせることが並列ＲＩＳＣプロセッサで高い性能を得るために有効であることが分かる。

5 おわりに

今回、ＲＩＳＣプロセッサ向け最適化の要件について述べ、その要件を満たした最適化機能の実装に触れた。さらに、実装した最適化機能についてその効果を測定し報告した。得られる効果から見て、命令レベルの最適化はＲＩＳＣプロセッサの性能を引き出すためには、必須の機能であるとが分かる。

並列ＲＩＳＣプロセッサの性能を向上させるためには、命令レベルの並列性を引き出す最適化技術が重要となる。本稿で述べた最適化をベースとして、並列ＲＩＳＣ向けの最適化の充実を図ることが、今後の重要な課題である。

参考文献

[1] 堀田 他, 「*RISC* プロセッサ向け最適化コンパイラの構成要件」, 情報処理学会第 46 回全国大会, 1993