RISCプロセッサ向け広域命令スケジューリング*

5D－8

松山 学　中平 直司　五十嵐 寛　林 正和　堀田 耕一郎†

富士通株式会社‡

1 はじめに

RISC, とりわけ並列RISCのようなアーキテクチャにとって, 命令スケジューリングはハードウェアの性能を引き出すための必須の技術であり, 我々は前回の大会で[1] これについて報告した.

今回は, 並列RISCプロセッサの持つ並列性を, 前回報告した局所命令スケジューリングよりもさらに高く引き出す方法として, トレース・スケジューリングという技術に注目し, これを実装したコンパイラを試作した. 本稿ではその効果及び問題点を報告する.

2 トレース・スケジューリング

2.1 局所命令スケジューリングの問題点とトレース・スケジューリング

局所命令スケジューリングでは基本ブロックがスケジューリングの単位であるため, 対象となる命令数が少なく, 並列性の向上の可能性が限られている.

そのためスケジューリングの可能性を拡げる手法として, より大きな範囲でスケジューリングを行なうスケジューリング手法が提案されている. その中で今回我々は実行確率の高いルート(トレース)を選んで, 重点的に最適化するという考えの'トレース・スケジューリング'という手法を選択した.

トレース・スケジューリングを構成する処理の内, 重要なものについて以下に挙げる.

2.2 トレースピック

分岐命令を越えてトレースを伸ばす時に, 1つのトレースをどの基本ブロックまで伸ばすかを判定するのがトレースピックである.

トレースピックでは, 分岐命令のどちらかの後続節を優先的に取ることで, 分岐命令を越えた複数の基本ブロックを, トレースとして取り出す.

トレースの取りだしを効果的に行うためにはトレースの途中からの分岐, 途中への合流が少なくなるよう選択することが望ましい.

今回の試作では, 分岐確率をソースプログラムに埋め込んで, コンパイラに与えることで効果的なトレースの選択ができるようにした.

2.3 命令スケジューリング

トレース・スケジューリングがトレースの範囲を決定すると, それに対する命令スケジューリングとレジスタ割り付けが, 次に重要度の高いトレースの決定に先駆けて行なわれる. この結果, 重要度が高いトレース内の変数は重要度の低いトレース内の変数より効果的にレジスタに割り付けることができる.

今回の試作では, 命令スケジューリングとレジスタ割り付けは並行して処理を行なっていく方法を取っており, 互いの効果を高め合う結果となっている.

2.4 ブックキーピング

ブックキーピングはスケジューリングの結果, 命令がトレースの分岐点や合流点を越えて移動した場合にプログラムの意味が変わらないように, その命令のコピーをトレースの分岐点や合流点につながる基本ブロックにコピーする処理である.

複数の分岐命令の順序がスケジューリングによって入れ替わった時など, コピーの量が爆発的に増えることがあるため, スケジューリングの段階でトレースを取り巻く制御の流れを良く意識する必要がある.

3 他の最適化との関連

コンパイラの持ついくつかの最適化機能の内, 特にプログラムの並列性を高め, トレース・スケジューリングと関連の深いものについて述べる.

---

*Global Scheduling for RISC Processors

†Manabu MATSUYAMA, Tadashi NAKAHIRA, Yutaka IGARASHI, Masakazu HAYASHI, KohIchiro HOTTA

‡FUJITSU LIMITED

スケジューリング方法 (TS: トレース・スケジューリング)	実行速度比	平均命令実行数 (命令数 / サイクル)	Proc_8 の実行 (サイクル数)	プログラムサイズ (bytes)
局所命令スケジューリングのみ	1.00	0.99	31	162808
TS (分岐確率指定無し)	1.04	1.00	29	162980
TS (分岐確率指定あり)	1.06	1.03	29	162912

(実行速度比は局所命令スケジューリングのみの場合を 1 とする.)

表 1: トレース・スケジューリングの効果

3.1 アンローリングとインライン展開

ループのアンローリングや, 関数のインライン展開などの最適化を行なうと, ループや関数の内側の基本ブロックを大きくできる可能性があるが, 内部に条件分岐を持つループや関数では, 基本ブロックの広がりはそれらの分岐によって限定されてしまう. 局所的な命令スケジューリングでは, そのため, これらの最適化によって効率が高められることが少なかった.

トレース・スケジューリングは, 分岐を越えたスケジューリングが可能なので, これらの最適化によって, 局所的な命令スケジューリングより処理範囲を拡げることができ, 一段と効率を高められると考えられる.

3.2 変数のリネーミング

アンローリングなどの最適化を実行すると, 一部の変数は生存範囲が著しく広がり, それらの変数を操作する命令の間に強い依存性が生まれる可能性がある.

変数のリネーミングを行なうと, 変数の生存範囲を狭くでき, プログラムの並列度の向上に貢献できる.

4 性能測定

以上の事柄を考慮に入れたコンパイラを試作し, その効果を見た.

使用したマシンは, SS-10 (Super SPARC 搭載, クロック: 36MHz 主メモリ: 48Mbyte) である.

プログラムは Dhrystone Ver 2.1 を使用した. ただし, プログラムの実行結果を調査し, それに基づいて, ループの回転数, 分岐命令の分岐確率をプログラムの各所 (計 13 箇所) に書き込んだものも使用した.

4.1 効果

命令スケジューリングより前の段階で種々の最適化 [3] を試みた後に,

- 局所命令スケジューリングのみを実行したもの.
- トレース・スケジューリングを実行したもの.

の 2 通りの処理を行なって生成したオブジェクトの速度を測定した. 結果を表 1に示す.

全く同じソースプログラムを使用したものについては約 4%, 分岐確率を与えたソースを使用すると 6% の速度の向上が見られた.

得られた翻訳結果を見ると, トレース・スケジューリングを行なうことにより並列度が増し, その結果, もっともコストの高い関数 (Proc_8) の実行時間が縮小していることが分かる.

プログラムが単純なためか, トレース・スケジューリングによるサイズの増加はあまりない.

4.2 評価

使用したハードウェアは, 最大並列度があまり大きくなく, またスーパースカラであるため, 並列実行できるところはハードウェアが自動的に並列実行するという, スケジューリングの効果が出にくい環境でのものであり, 6% の性能向上は高く評価できると考える.

5 まとめ, 今後の課題

トレース・スケジューリングを用いた, コンパイラの並列性の向上について述べ, さらに, それに基づくコンパイラを試作し, その効果を見た. 今後はさらに, トレース・スケジューリングの技術の改善, 他の最適化の併用による効果の違い, あるいは分野によるプログラムの傾向の違いによるトレース・スケジューリングの効果について検討を重ねて行きたい.

参考文献

[1] 林 他
「RISC プロセッサ向け命令スケジューリング」
情報処理学会第 46 回全国大会, 1993

[2] John R. Ellis *Bulldog: A Compiler for VLIW Architectures*
The MIT Press 1986

[3] 中平 他 「RISC プロセッサ向け最適化機能とその評価」
情報処理学会第 47 回全国大会, 1993