組み込みマイコン用Ｃコンパイラにおける命令スケジューリングの一方式

2Ｎ－9

小谷 謙介, 佐山 旬子, 田中 旭, 湯川 博司
松下電子工業(株)

１．はじめに

組み込み制御用ソフトウェア規模は年々増大し、高級言語による開発が要望されている。これに対し我々は、高級言語による開発とオブジェクトコードサイズの低減という相反する条件を満たす為に、マイコンとＣコンパイラの一体設計を行ない、少数レジスタ･基本命令語長 1byte の組み込みマイコンを開発した。しかし近年では、これに加えて組み込みマイコンに対する性能面での要求が高まっている。

今回、コンパイラからの高速化技法として、命令スケジューリング方式を開発したのでこれを紹介する。本方式はレジスタ数が少なく、レジスタの使い回しが多い場合でも有効に働く。

組み込み用途のコンパイラでは、生成オブジェクトサイズがコストに直結するため、生成コードサイズの削減が重要である。このため性能最優先のＣＰＵ等に用いられるような、コードサイズを増加させるおそれのある方式の採用は難しい。この点を考慮して、今回は命令のコピーを伴わない基本ブロック内でのスケジューリングを採用した。

２．パイプライン処理と命令スケジューリング

パイプラインはプロセッサの処理速度向上のために今日では一般的に用いられる技法である。しかし、パイプライン処理は常に適切に実行される訳ではない。ある命令の実行が先行命令の実行結果に依存している場合には、後の命令の実行を先行命令の結果が確定するまで待つ必要がある。(ハザード)

命令スケジューリングはこのハザード状態をオブジェクトコードの順番を入れ換えることにより解決する技法である。連続して実行するとハザードが生じる命令を並べ替え、実行順序を入れ換えることによりハザードが生じない命令列に変換する。

An Instruction Scheduling Method of C Compiler for Embedded Micro Computer
Kensuke Odani, Junko Sayama, Akira Tanaka, Hiroshi Yukawa
Matsushita Electronics Corporation

３．従来方式とその問題点

基本ブロック内で命令スケジューリングを行なう基本的手順は、以下のようになる。[1]

1　命令列を解析し、依存関係による依存グラフを形成する。
2　全ての命令が出力されるまで以下を繰返す。
　2.1　依存グラフの根ノードを候補集合とする。
　2.2　候補集合からハザード解消の観点から最も適した命令を選びだし出力する。
　2.3　出力した命令を依存グラフから取り除く。

命令の実行順序を規定する依存関係としては、Write-Read(データ依存)、R-W(逆依存)、W-W(出力依存)が存在する。しかし、R-W, W-W は命令スケジューリング時の依存関係としては条件が厳し過ぎ、実際には可能であるはずの命令の並べ替えによるハザード解消が不可能になることがあった。

例）以下のプログラムを処理する場合を考える。

```
1...Mov 10,D0
2...Mov D0,A0
3...Mov A0,mem
4...Mov 20,A0
5...Mov (A0),D1
6...Add D1,D1
```

この例では 5-6 間にハザードが存在する。このプログラムの依存関係グラフは図１の様になり、命令実行順序は１通り(ソースプログラム通り)に固定され、ハザードを解消することはできない。

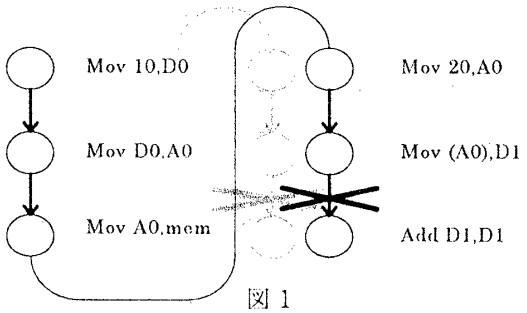


図 1

レジスタ数が少ないほど、同じレジスタの使い回しが増えるため、このような事態が生じやすくなる。

4．今回の方式

上記問題点を解消するため、本方式では W-R 依存関係とリソース使用状態の動的把握により命令スケジューリングを実現した。本方式の手順を以下に示す。

1　命令列を解析し、W-R 依存関係のみによる依存グラフを形成する。

2　全ての命令が出力されるまで以下を繰返す。

2.1　依存グラフの根ノードを候補集合とする。

2.2　候補集合から 1 命令が出力されるまで以下を繰返す。

2.2.1候補集合からハザード解消の観点から最も適した命令を選びだし出力候補とする。

2.2.2出力候補が設定するリソースが現在使用中であるかどうかをリソース使用状態フラグにより判定する。

未使用の場合　出力候補を出力してもプログラムの意味が変化してしまうことはないので、出力候補を出力し、依存グラフから取り除く。また、リソース使用状態フラグを更新する。

使用中の場合　当該候補命令を出力すると有効な値を破壊してしまうことになるので、この出力候補を不適として候補集合から取り除き、ループ先頭(2.2)へ戻る。

次に前出のプログラムを処理した際の結果を示す。W-R 依存関係のみによる依存関係グラフは図 2 のようになる。

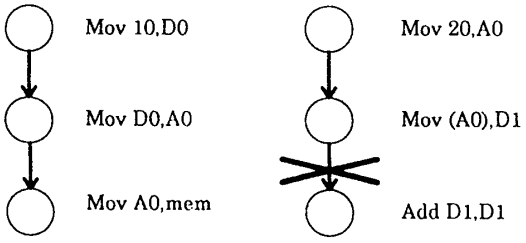
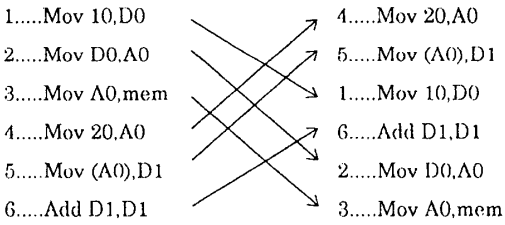


図 2

この場合には、命令列が 2 つの Tree に分離されているので、命令の並べ替えが可能となる。このため、ハザードが生じている命令 5-6 間に他命令を挿入することができ、図 1 では解消できなかったハザードが解消可能となる。

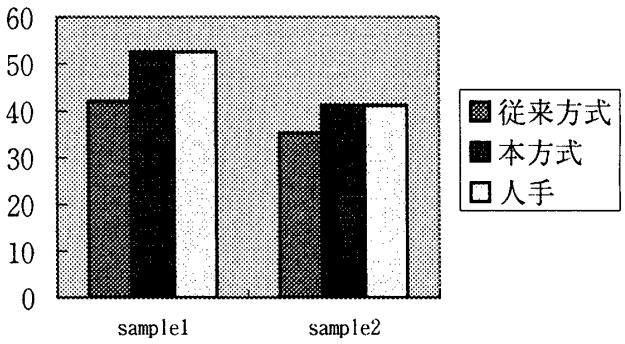


また本方式では、候補集合にある命令全てがリソース使用状態フラグ判定の結果出力不可能となり、デッドロック状態に陥ってしまう場合があり得る。これに対しては、スケジューリング部にバックトラック機構をそなえ、1 つ前の命令選択をやり直すことでデッドロック状態から脱出できるようにした。

5．評価

一般的なベンチマークプログラムに対して命令スケジューリングを行いハザードの解消率を調べた。プログラム中の全ハザード数を 100 としたときに解消されたハザードの割合を示す。本方式では人手でスケジューリングした場合と遜色ない結果が得られ、従来方式に比べ約 10%の改善が見られた。

図 3　ハザード解消率



6．まとめ

組み込み用パイプライン処理マイコンに適合した命令スケジューリング方式を示した。本方式は W-R 依存関係のみを用い、ハザード解消率を向上している。今後は、本方式により分離された命令依存グラフにレジスタ再割当を施し、更にハザード解消率を向上させていく予定である。

参考文献

[1] 松岡 恭正、河内 浩明、茂木 強 「コンパイラにおける最適化の技法」 インターフェース Mar. 1989, p196 - 210