ショート・リブド・データの動的な予測に関する検討

吉瀬謙二，高峰信，田中洋介，坂井修一，田中英彦
東京大学大学院工学系研究科

1　はじめに

機能ユニットが生成するデータの中で、リネーム・レジスタやレジスタ・ファイルからは一度も供給されることなく消滅するデータのことをショート・リブド・データ (short-lived data) と定義する。別の見方をすれば、ショート・リブド・データとは、生成されたが全く使われないか、フォワーディング・パスのみを用いて供給されるデータのことであり、本来、これらのデータをリネーム・レジスタやレジスタ・ファイルに格納する必要はない。本稿では生成されるデータの多くがショート・リブド・データであることを確認するとともに、その動的予測の可能性を検討する。

2　ショート・リブド・データ

プロセッサのデータの流れを図 1に示す。機能ユニットが生成したデータは、フォワーディング・パスを用いて必要としている機能ユニットに供給されるとともに、リオーダバッファ内のリネーム・レジスタに格納される。リオーダバッファは FIFO になっておりインオーダにレジスタ・ファイルにデータを書き戻す。

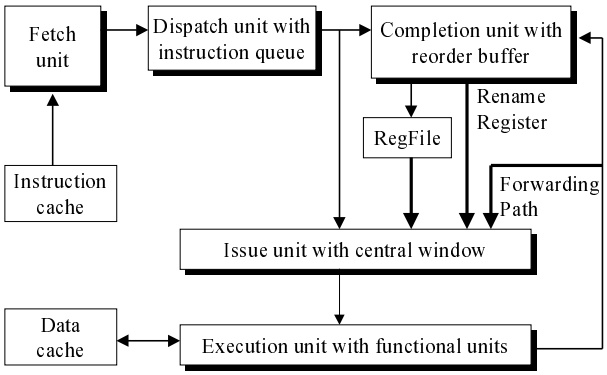

図 1: プロセッサ内のデータの流れ

通常のプロセッサ・モデルでは、生成された全てのデータをレジスタ・ファイルに格納し、同じレジスタ番号のデータが書き戻されるまで、当該データはレジスタ・ファイル中に保存される。

しかしながら、プログラムの実行中に生成されるデータの多くが生成されてから短い間隔でのみ利用されるという性質が指摘されており [1]、リネーム・レジスタやレジスタ・ファイルへの書き込みが必ず必要というわけではない。フォワーディング・パスのみを用いて供給できるデータをリネーム・レジスタに格納する必要はないし、レジスタ・ファイルから一度も供給されないデータをレジスタ・ファイルに書き戻す必要はない。以上の考察から、不必要なデータの移動を削減することが本研究の目的であるが、本稿では、リネーム・レジスタへ格納するデータ量の削減に絞って議論を進める。

リネーム・レジスタやレジスタ・ファイルへの書き戻しは、現在のスーパースカラ・プロセッサではボトルネックとなっていないが、今後、命令レベル並列性の向上や、複数パスの同時実行といった技術によりサイクル当たりに生成されるデータ量は増加する傾向にある。リネーム・レジスタへ格納するデータ量の削減は、今後予測されるボトルネックを解消するための重要な技術である。

文献 [1] ではレジスタ・ファイルへの書き戻しが不要なデータを short lived variable と定義して静的に検出する手法を提案しているが、本稿ではリネーム・レジスタやレジスタ・ファイルから一度も供給されることなく消滅するデータのことをショート・リブド・データ (short-lived data) と定義し、その検出機構と動的予測の可能性を検討する。

3　ショート・リブド・データの検出機構

データが消滅するまでにリネーム・レジスタあるいはレジスタ・ファイルからアクセスされたデータはショート・リブド・データではない。このため、アクセスの有無を示すフラグを用いてショート・リブド・データを検出する。フラグは各リネーム・レジスタと、レジスタ・ファイルと同じエントリ数のアクセス履歴テーブルに保存する (図 2)。

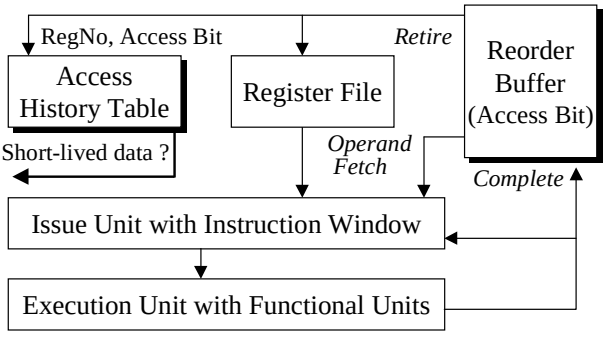

図 2: ショート・リブド・データの検出機構

リオーダバッファに付加するアクセスビットは、データがリオーダバッファ内に存在する時のアクセスの有無を保存し、アクセス履歴テーブルは対応するレジスタ・ファイルのデータに対するアクセスの有無を保存する。これ

An analysis of the dynamic prediction for short-lived data
Kenji KISE, Makoto TAKAMINE, Yosuke Tanaka, Shuichi SAKAI, Hidehiko TANAKA
Graduate school of Engineering, The University of Tokyo

らの情報を用いて、データが消滅する際に、そのデータがショート・リブド・データであるかを検出できる。

4 ショート・リブド・データの動的な予測

ある命令の生成したデータがショート・リブド・データだった場合に、その命令の次回の実行結果もショートリブド・データである確率が高いといった偏りがあれば、ショート・リブド・データの動的な予測の可能性が生まれる。

命令アドレス（プログラム・カウンタ）をインデックスとして前回の実行の結果がショート・リブド・データであったかを保存するテーブルを用意し、前回のデータがショート・リブド・データだった場合に、今回の結果もショート・リブド・データであると予測した場合の予測成功率を測定する。ここで、ショート・リブド・データの検出には3章で定義した機構を用いる。

5 評価環境

トレース・ドリブンのシミュレータを用いて、ショート・リブド・データの割合と予測成功率を評価する。評価に用いるベンチマークを表1にまとめる。プログラムの入力セットは実行命令数が１億程度に収まるように調整し、コンパイルには Gnu C compiler （最適化オプション -O、ターゲット SPARC version 7) を用いた。

Program	Description	Input Set
cc1	From GCC 2.7.1	genrecog.i
compress	File compression	small.in
go	Plays the game of Go	9 9
m88ksim	Moto88K simulator	ctl.raw
perl	Manipulates strings	“admits” in 1/8 input
xlisp	LISP interpreter	6 queens

表 1: ベンチマークプログラム

評価には図1に示す標準的なスーパースカラのモデルを用いる。プロセッサの主なパラメタを表2にまとめる。

Parameter	Value
Bandwidth of decode, dispatch, commit, retire	8 instr. per cycle
Reorder buffer	32 entries
Instruction Window	8 entries
Memory latency (no cache miss)	2 cycle
The number of Functional Unit	No limitation

表 2: 評価に用いたプロセッサ・モデルのパラメタ

6 評価結果

ショート・リブド・データの検出結果を表3に示す。それぞれのプログラムについて、表3の左の列から実行命令数、生成されたデータの量、ショート・リブド・データの量（生成されたデータに対するショート・リブド・データの割合）を示した。表中の M は million を意味している。表3の結果より、生成されたデータの 72%から 82% がショート・リブド・データであることがわかる。もし、これらのデータをリネーム・レジスタやレジスタ・ファイルに書き戻す必要がないとすると、データ転送量の７割以上を削減できることになる。

Program	Executed Inst.	Created Data	Short-lived Data (ratio)
cc1	120 M	60 M	49 M (82.1%)
compress	54 M	27 M	22 M (81.3%)
go	136 M	78 M	62 M (79.2%)
m88ksim	124 M	66 M	54 M (82.8%)
perl	101 M	52 M	40 M (78.3%)
xlisp	42 M	19 M	13 M (72.4%)

表 3: ショート・リブド・データの割合

ショート・リブド・データの予測について、予測成功の回数 (Hit)、失敗の回数 (Miss)、予測成功率 (Hit / (Hit + Miss)) を表 4 にまとめる。ここでいう予測とは、4章で説明した、前回の実行により生成されたデータがショート・リブド・データだったときに今回の実行結果もショート・リブド・データであると予測する単純なものである。

Program	Hit	Miss	Hit Ratio
cc1	46,669,409	2,243,563	95.4%
compress	21,432,337	844,565	96.2%
go	58,699,172	3,052,470	95.1%
m88ksi	53,769,345	667,142	98.8%
perl	38,778,014	1,567,074	96.1%
xlisp	12,690,702	719,340	94.6%

表 4: ショート・リブド・データの予測成功率

評価結果 (表 4) よりショート・リブド・データの予測成功率は 94% 以上に達することがわかった。ショート・リブド・データである確率は 82%以下だが、前回の履歴を用いることで 94%以上の確率で予測できるという結果から、前回のデータがショート・リブド・データだった場合には今回もショート・リブド・データであるという偏りがあることがわかる。

7 まとめと今後の課題

フォワーディング・パスのみにより供給されるデータをショート・リブド・データと定義し、その検出機構と動的な予測の可能性を検討した。標準的なスーパースカラ・プロセッサを想定して評価した結果、生成されるデータの７割以上のデータがショート・リブド・データであることを確認した。また、前回の履歴を用いることにより、94%以上という高い確率でショート・リブド・データを動的に予測できることを示した。ただし、プロセッサ内でショート・リブド・データを予測する場合には、予測に失敗した場合の回復機構が必要となる。回復機構を含む予測機構の検討は今後の課題である。

参考文献

[1] Luis A. Lozano C. and Guang R. Gao. Exploiting Short-Lived Variables in Superscalar Processors. In Proc. of MICRO-28, pages 292-302, 1995.