Ephemeral Variables: ALU-NET によるアクセスの局所性の利用

3H－1

辻 秀典,　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

命令レベル並列実行によって性能を得るマイクロプロセッサが一般的となった現在、さらなる性能向上をめざした次世代マイクロプロセッサに関するさまざまな研究が行われている。そのひとつとして、我々は大規模データパス (Very Large Data Path – VLDP)・アーキテクチャ[3] を提案している。これは、マイクロプロセッサにおける命令実行の本質が演算であることに注目し、命令列中のデータフローを ALU-NET [1] と呼ぶ複数の ALU によって構成される機構上で実現することによって、命令実行のスループットを向上させようとするアーキテクチャである。

本稿では、この ALU-NET を利用した局所的なデータアクセスを可能とする Ephemeral 変数を定義し、これがどのようなものか説明したうえで分類する。

2　Ephemeral 変数

マイクロプロセッサにおける処理は、メモリ空間上のデータに何らかの演算処理を施し、得られた結果を再びメモリ空間上に書き戻すことが基本である。データの流れに着目した場合、あるデータはメモリからレジスタに読み込まれ、レジスタを介して処理が行われる。このとき、レジスタへの値の書き戻しから次の参照までの時間は短いものから長いものまでさまざまである。これらのレジスタアクセスのうち、定義から参照までの時間が短いものは、一時的なデータの保存場所としてレジスタを利用するものとみなすことができる。

ALU-NET はこのようなデータアクセスを、ALU 同士の局所的な接続によって実現するもので、これによって、処理を高速化するとともにレジスタの有効利用を可能となる。そして、一時的なデータの保存場所としてレジスタを利用する、局所的なデータアクセスのために使われる変数を Ephemeral 変数と定義する。

3　Ephemeral 変数の分類

命令列中に存在する Ephemeral 変数を利用するためには、これを検出しなければならない。ここでは、検出と利用のために、いくつかの条件により Ephemeral 変数を分類する。

3.1　スーパースカラとの対応

スーパースカラ・アーキテクチャは、一般にはデータ依存を解消するためにリオーダバッファを用いた、レジスタリネーミングが行われている。このようなスーパースカラにおけるデータアクセスのモデルは図 1のようになる。ここでは、Ephemeral 変数がスーパスカラのどのデータアクセスパスによって処理されるかによって次の 3 種類に分類する。

1. forwarding path
2. reoder buffer – instruction window
3. reoder buffer – register file – instruction window

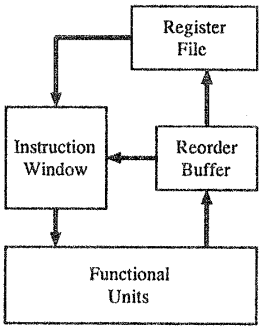

図 1: スーパースカラのデータアクセス・モデル

3.2　Ephemeral 変数の参照数

Ephemeral 変数は、ある命令 (producer) によって生成されるデータを消費する命令 (consumer) に受け渡す変数である。Ephemeral 変数はデータを消費する命令によって参照されるが、ひとつの Ephemeral 変数に対して複数の参照が行われる可能性がある。図 2 では参照数 1 と 3 の例を示している。ここでは、その参照数により Ephemeral 変数を分類する。ALU-NET 上で Ephemeral 変数に対する複数の参照を可能とすれば、処理は効率化できるがそれとともに ALU-NET は複雑化する。

3.3　Ephemeral 変数の検出

Ephemeral 変数の検出を行う際には、図 3 に示すように、最後の参照または再定義を検出することによって、Ephemeral 変数であることを決定する。理想的には最

Ephemeral Variables: Exploiting Access Locality using ALU-NET
Hidenori TSUJI, Shuichi SAKAI, Hidehiko TANAKA
Graduate School of Engineering, The University of Tokyo

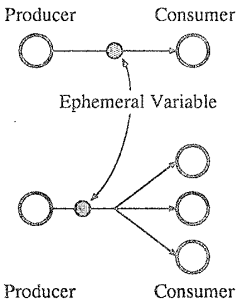

図 2: Ephemeral 変数の参照数

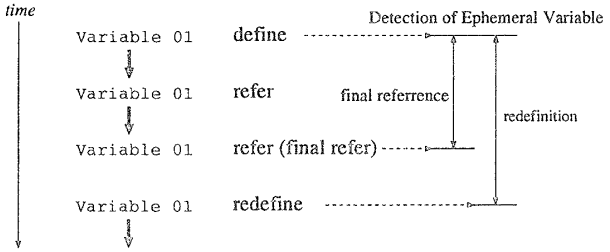

図 3: Ephemeral 変数の検出

後の参照を検出することが望まれるが、最後の参照はプログラムを最後まで実行しなければ保証できない。そのため再定義を検出する方が容易である。

3.4 参照の距離

ここでは、Ephemeral 変数を参照される距離によって分類する。最も単純には、基本ブロック内に限定されたアクセスのみを Ephemeral 変数とする方法である。しかし、基本ブロックの平均サイズは一般的に 5 程度と小さいために、より多くの Ephemeral 変数を検出して利用するためには、ある程度、参照の距離を大きくする必要がある。参照の距離によっては、図 4 に示すように、ある変数を例にとっても、それが Ephemeral 変数になる場合とならない場合がある。

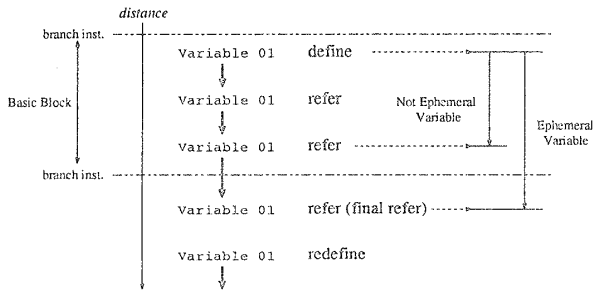

図 4: Distance of Reference

3.5 複数パスにおける保証

Ephemeral 変数の参照が基本ブロックを越える場合には、複数の分岐パス対して Ephemeral 変数であるかどうかの保証を考えなければならない。図 5 は、あるひとつの分岐パスで Ephemeral 変数と検出される場合と、すべての分岐パスで Ephemeral 変数と検出される場合の例である。

あるひとつの分岐パスのみで Ephemeral 変数であると検出された場合に Ephemeral 変数とする場合が最も楽観的であるが、これでは他の分岐パスでは Ephemeral 変数であるかどうかを保証できない。逆に、すべての分岐パス上で Ephemeral 変数であると検出する方法が最も厳密な場合である。これは、どのパスが実行されても Ephemeral 変数として処理することができるが、検出に手間がかかるだけでなく、Ephemeral 変数の割合も減少する。

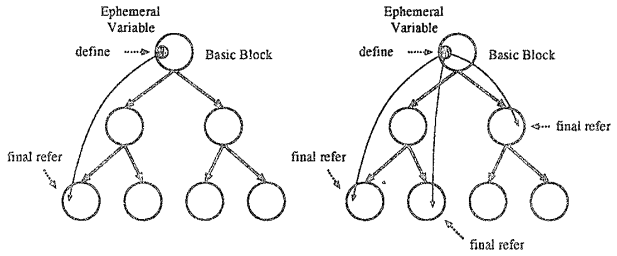

図 5: 複数パスにおける保証

4 最後に

Ephemeral 変数を検出した結果は文献 [2] によって述べられている。その仮定は本稿の分類によると、複数の参照を許可、変数の再定義のによる検出、すべての分岐パスにおいて保証される Ephemeral 変数である。このような基本的で厳しい仮定において、SPECint95 ベンチマークプログラム (SPARC コードで評価) の 21% ～ 34% の命令が Ephemeral 変数を生成していることが分かっている。

今後、本稿で分類した他の仮定を適用すれば、より多くの Ephemeral 変数を検出できると考えられる。また、Ephemeral 変数の分類と ALU-NET の構造の関係について議論する必要もある。

参考文献

[1] 辻秀典, 中村友洋, 吉瀬謙二, 安島雄一郎, 高峰信, 坂井修一, 田中英彦: ALU-NET: VLDP アーキテクチャにおける命令実行機構, 情報処理学会 第 57 回全国大会, Vol. 1, No. 1Q-10, pp. 40–41 (1998).

[2] 高峰信, 辻秀典, 吉瀬謙二, 田中洋介, 坂井修一, 田中英彦: VLDP アーキテクチャにおけるデータアクセスの軽減手法, 情報処理学会研究会 ARCH, Vol. 133, No. 6, pp. 31–36 (1999).

[3] 中村友洋, 吉瀬謙二, 辻秀典, 安島雄一郎, 田中英彦: 大規模データパスプロセッサの構想, 情報処理学会研究会 ARCH, Vol. 124, No. 3, pp. 13–18 (1997).