1X-4

ウィンドー・レジスタの評価

堀川　隆
日本電気(株)　C&Cシステム研究所

1　はじめに

大容量レジスタ・ファイルを搭載したRISCマイクロ・プロセッサが注目されている。レジスタ・ファイルを介してプロシージャ・コールの際のパラメータ受け渡しを行なうことにより、パラメータに関するチップ外部へのアクセスを削減できることが大きな特徴である。この方式では、プロシージャ・コールが高速になる反面、プロシージャ・コールの深いネストに起因するレジスタ・ファイルのオーバー・フロー、アンダー・フロー、および、プロセス切り換え時におけるレジスタ・セーブのオーバーヘッドが問題となる。そこで、2種類の実システムを測定して得られたデータに基づいて、上記項目の評価を行なった。

2　ウィンドー・レジスタ

2.1　概要

パラメータ受け渡しに使用する大容量レジスタ・ファイルの概念的な構造は、図1に示すように、カレント・ポインタとリミット・ポインタを持つリング構造である。カレント・ポインタは、スタック・ポインタに相当しており、コール時には前方、リターン時には後方へ移動する。このポインタが、リミット・ポインタを越えて移動しようとすると、オーバー・フローやアンダー・フローが発生する。

ここでは、プロシージャ・コール時における、カレント・ポインタの移動量が固定されている方式（以下、**レジスタ・ウィンドー方式**）と、パラメータの個数に応じて移動量を変えられる方式（以下、**パラメータ・キャッシュ方式**）について評価を行なった。

2.2　パラメータのレジスタ渡し

プロシージャ・コール時には、プログラム中に陽に指定したパラメータに加えて、プログラム・カウンタの値（戻り番地）とスタック・ポインタの値をレジスタに格納する。すなわち、プロシージャ・コール時には、パラメータ数+2個のレジスタを使用するものとした。

また、プロセス切り換え時には、有効な値が格納されているレジスタ（カレント・ポインタとリミット・ポインタの間）をメモリに退避し、レジスタ・ファイルを空の状態に戻した後、プロセスを切り換えるものとした。

3　評価

3.1　評価方法

ハードウェア・トレーサ[1]を用い、8Mステップのメモリ・アクセス系列を下記の2システムから各々2サンプルずつ計4サンプル採取した。測定期間は、1サンプル当り3～5秒間である。ハードウェア・トレーサによると、測定操作に起因する測定対象システム動作の乱れが全くないため、実時間制御システムについても実働状況を調べることが可能である。

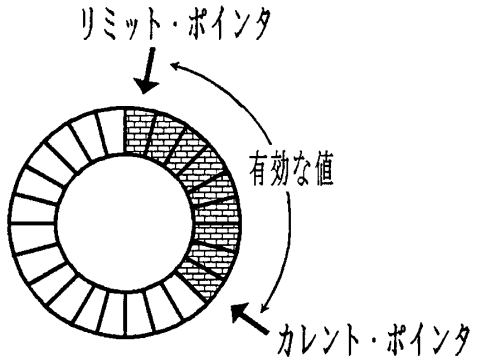

図1: パラメータ受け渡し用レジスタの概念

unix
　cコンパイラとオブジェクトの実行

リアルタイム制御システム
　高負荷をかけたときのイベント処理

測定により得られたメモリ・アクセス系列データを命令系列データに変換した後、パラメータの受け渡し、プロシージャ・コール、リターン、プロセス切り換えのための命令を抽出し、後のシミュレーションに使用した。

3.2　評価内容

ここでは、以下に示す評価を行なった。

出現頻度
　命令系列データをもとに、下記項目について統計をとった。

- コール、リターン、プロセス切り換えの回数
- 陽に指定されるパラメータ数の平均

パラメータに関するメモリ・アクセス数
　下記要因によるメモリ・アクセス数とレジスタ数の関係をシミュレーションにより求めた。ここでは、レジスタ・ウィンドー方式とパラメータ・キャッシュ方式の両者を対象とした。

- プロシージャ・コール時のオーバー・フローによるストア
- リターン時のアンダーフロー
- プロセス切り換え時のレジスタ・セーブ（ストア）

4　結果および考察

4.1　出現頻度

結果を表1に示す。unixとリアルタイム制御システムの大きな違いは**プロセス切り換え回数**に現われてい

Window Register Evaluation using Realtime Tracedata
Takashi Horikawa, NEC Corporation

	unix	リアルタイム システム
コール、リターン回数	41000～47000	35000～40000
プロセス切り換え回数	42～46	750～1300
平均パラメータ数	1.7～1.9	2.2～2.3
プロセス切り換え間のコール回数	890～1100	26～53

表 1: 出現頻度

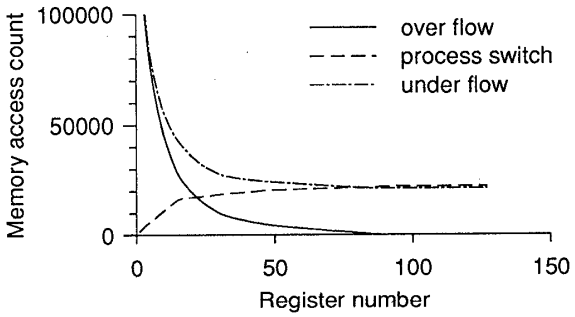

図 2: パラメータ・キャッシュの効果 (realtime)

る。すなわち、リアルタイム制御システムでは、unix より 15～30 倍のプロセス切り換えを行なっている。その他の項目には、これほど大きな差はない。その結果、プロセス切り換えの間に実行されるプロシージャ・コール回数は、unix の方が 20～40 倍多くなっている。

また、コール時に渡されるパラメータの個数は大部分が 6 個以内であり、これを越えるパラメータ個数は全体の 1 %以下であることもわかった。

4.2 ワークロードによる違い

unix とリアルタイム・システムのデータをもとに、パラメータ・キャッシュ方式について行なったシミュレーション結果を図 2、図 3に示す。unix の方が、レジスタ数増加によるメモリ・アクセス減少の効果が大きい。これは、プロセス切り換え間のコール回数が多いためであると考えている。ただし、1 回のプロセス切り換え時にセーブすべきレジスタ数は unix の方が多い。

注目すべき点は、**オーバー・フローとプロセス切り換えの関係**である。リアルタイム・システムでは、レジスタ数が 20～30 を越えると、前者より後者の影響が大きくなる。すなわち、リアルタイム制御システムでは、ある程度以上にレジスタ数を多くすることは効果がなく、各々のプロセスにレジスタを割り当てる方が望ましい。これは、一般にいわれていたことではあるが、ここで行なった評価により、定量的な形で明らかにすることができた。

なお、プロセス切り換えによりレジスタが空になった後のリターンはアンダー・フロー扱いとしたため、アンダー・フロー回数は、オーバー・フロー回数ほどは減少しない。

4.3 シフト幅を固定する影響

プロシージャ・コール時のシフト幅を 8 に固定したレジスタ・ウィンドー方式のシミュレーション結果を図 4に示す。図 3との比較より、レジスタ・ウィンドー方式は同じレジスタ数のパラメータ・キャッシュ方式に比べ、パラメータ・アクセスに関しては 3～6 倍のメモリ・アクセスを行なっている。

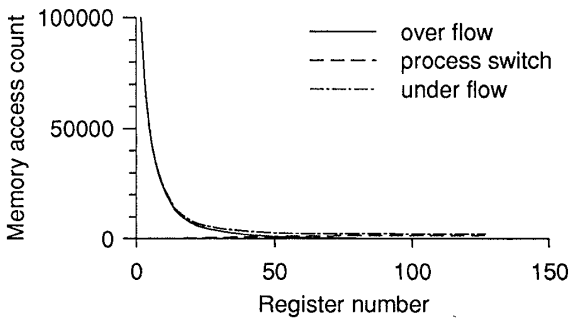

図 3: パラメータ・キャッシュの効果 (unix)

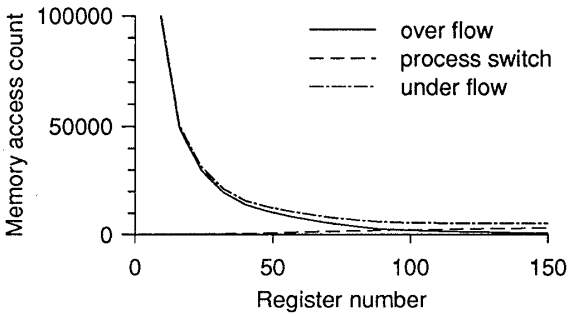

図 4: レジスタ・ウィンドーの効果 (unix)

この差の要因は、**オーバーフローによるメモリ・アクセスの違い**である。すなわち、1 コール当り陽に指定されるパラメータ数の平均は約 2 であることから、シフト幅が 8 のレジスタ・ウィンドー方式では、平均 4 個分のレジスタを余分にシフトすることになるため、オーバーフローが発生し易くなっているのである。プロセス切り換え時にセーブするレジスタ数については、大きな差はない。

なお、レジスタ・ウィンドー方式においてパラメータの受け渡しに使わなかったレジスタは、ローカル変数として使用することによりメモリ・アクセス削減に役立つため、ここで示した結果のみから直ちに優劣を比較することはできない。

5 おわりに

大容量レジスタ・ファイルを利用した『パラメータのレジスタ渡し』について評価を行なった結果、その効果は**ワークロードの種類に大きく依存**することが明らかになった。従って、大容量レジスタ・ファイルを有効に活用するためには、ワークロードの性質を考慮した検討が必要である。

参考文献

[1] 堀川、大鷹、大野、加藤、“ハードウェア・トレーサを用いた計算機アーキテクチャ評価システム”, 情報処理学会 OS 研究会,87-OS-34-3, 昭和 62 年 2 月.