

ソフトウェア・シミュレータ
の高速化手法

2H-5

田中 利清

NTT 情報通信処理研究所

1. はじめに

新アーキテクチャ・マシンの設計評価・試験用テストプログラムに対するデバッグ環境の提供、デスクサイドでのプログラム・デバッグ環境の提供および充実したデバッグ手段の提供を目的として、ソフトウェア・シミュレータを開発した。

主要事項は以下のとおりである。

- ① 操作性、レスポンスタイムの観点からワークステーション上に構築した。
- ② テストプログラムのデバッグを開発目的の一つとすることから、シミュレーションのレベルを機械語レベルに設定した。
- ③ 移植性の観点からC言語を使用した。

以下に、開発したソフトウェア・シミュレータのシステム構成概要とアドレス変換の高速化手法についての評価結果を示す。

2. システム構成

開発したソフトウェア・シミュレータのシステム構成を図1に示す。

このうち、命令シミュレーション部、割込みシミュレーション部およびメモリアクセスシミュレーション部が、対象マシン（アーキテクチャ）の動作をシミュレーションする部分である。

3. アドレス変換の高速化
仮想記憶方式を採用するアーキテクチャのシミュレーションにおいては、メモリ・アクセス（命令フェッチを含めて1命令当たり平均1.5～2回）のつど論理アドレスから実アドレスへの変換が必要なことから、アドレス変換のシミュレーションが全シミュレーション時間の大きな割合（70%～80%）を占める。

実マシンにおいてもアドレス変換の高速化は性能向上効果が大きく、汎用コンピュータはもとよりマイクロプロセッサにおいても、過去にアクセスした論理アドレスのアドレス変換のシミュレーションが全シミュレーション時間の大きな割合（70%～80%）を占める。

実マシンにおいてもアドレス変換の高速化は性能向上効果が大きく、汎用コンピュータはもとよりマイクロプロセッサにおいても、過去にアクセスした論理アドレスのアドレス変換のシミュレーションが全シミュレーション時間の大きな割合（70%～80%）を占める。

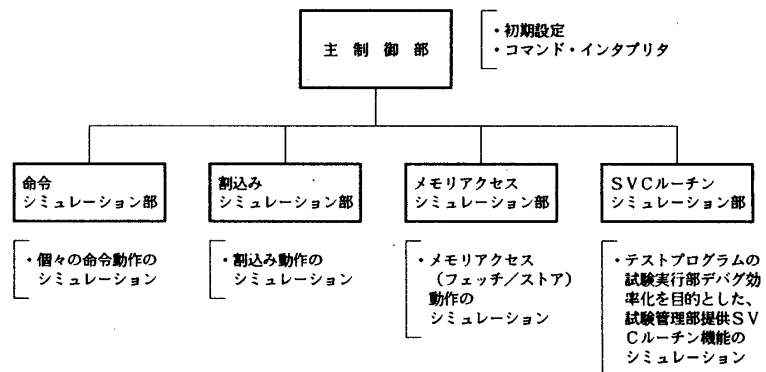


図1 システム構成

表1 アドレス変換高速化手法

方 式	MRU方式	List (FILO) 方式	List (FIFO) 方式	Table方式
方式概要	直前にアクセスした論理ページのアドレス変換情報のみを保持する	過去にアクセスした論理ページのアドレス変換情報を全てリスト構造を用いて保持する	同左	過去にアクセスした論理ページのアドレス変換情報を管理テーブルを用いて保持する
保持エントリ数	2（命令/オペランド各1）	可変	同左	固定（例：256）
検索方法	唯一の保持エントリについて論理ページ番号の比較を行う	リストの先頭から保持されている全てのエントリについて順次論理ページ番号の比較を行う（論理ページ番号が一致するエントリが見つかるまで）	同左	論理ページ番号を用いてハッシュングを行い、その結果のハッシュ値を管理テーブルのエントリ番号として、対応するエントリについて論理ページ番号の比較を行う
検索失敗時の動作	アドレス変換を行い、その結果を保持エントリに登録する（以前のアドレス変換情報は削除される）	アドレス変換を行い、その結果をリストの先頭に追加する（以前のアドレス変換情報は全て保存される）	アドレス変換を行い、その結果をリストの末尾に追加する（以前のアドレス変換情報は全て保存される）	アドレス変換を行い、その結果を管理テーブルに登録する（論理ページ番号のハッシュ値が同一のエントリが既に登録されている場合には前のアドレス変換情報が削除される）
比較	検索時間	◎	△	△
	ヒット率	△	◎	◎

An efficient method for software simulator

Toshikiyo Tanaka

NTT Communications and Information Processing Laboratories

ドレス変換情報をバッファ内に保持してアドレス変換の高速化を図るTLB方式を採用するものが多い。しかし、実マシンでは内部的に並列処理が可能であるが、専用のハードウェアを使用しないソフトウェア・シミュレータでは基本的に逐次処理しかできないため、TLB方式をそのままソフトウェア・シミュレータに採用できない。

(1) 高速化手法

ソフトウェア・シミュレータのアドレス変換高速化手法として、表1に示す4つの方式について、比較検討を行った。検索時間については、処理時間が不要なMRU方式が最も有利であり、目的のアドレス変換情報が見つかるまでリストをたどるList方式が不利である。一方、ヒット率については、過去にアクセスした全ての論理アドレスのアドレス変換情報を保持するList方式が最も有利であり、アドレス変換情報を1エントリしか保持しないMRU方式が不利である。

(2) 高速化手法の評価

具体例として、テストプログラムへの適用結果に基づき、ソフトウェア・シミュレータのアドレス変換高速化手法の定量的評価を行った。評価項目は、高速化手法を用いない場合のシミュレーション時間を T_0 、高速化手法を用いた場合のシミュレーション時間を T_i とした時の、高速化率 $S_i = T_0 / T_i$ を採用した。

高速化率の評価結果を、各方式を単独に適用した場合とMRU方式と他の方式を組み合わせる場合について

て、それぞれ図3、図4に示す。図3および図4から導かれる結論を以下に示す。

- ① 高速化手法を用いない場合に比べて、4つの高速化手法を単独で用いた場合約3倍高速化され、MRU方式と他の3つの方式をそれぞれ組み合わせて用いた場合約3.5倍高速化される。
- ② 高速化手法を単独で用いる場合には、MRU方式が最も効率的である。
- ③ MRU方式と他の高速化手法を組み合わせる場合には、MRU方式とList (FIFO) 方式の組み合わせが最も効率的である。

4. おわりに

ソフトウェア・シミュレータについて、システム構成、アドレス変換高速化手法の比較および評価結果を述べた。上記評価結果は、比較的命令シーケンス長が短い多数のルーチンが順次実行されるという特徴を持つテストプログラムを対象としたものであり、今後他のデバッグ対象プログラムについても評価を行う必要がある。

参考文献

- (1) 高橋茂：電子計算機I—アーキテクチャとハードウェア構成方式—、共立出版
- (2) 井上勝博、三原幸博：ハードウェア動作記述からのシミュレータ自動生成、情報処理学会ソフトウェア工学研究会68—3、1989

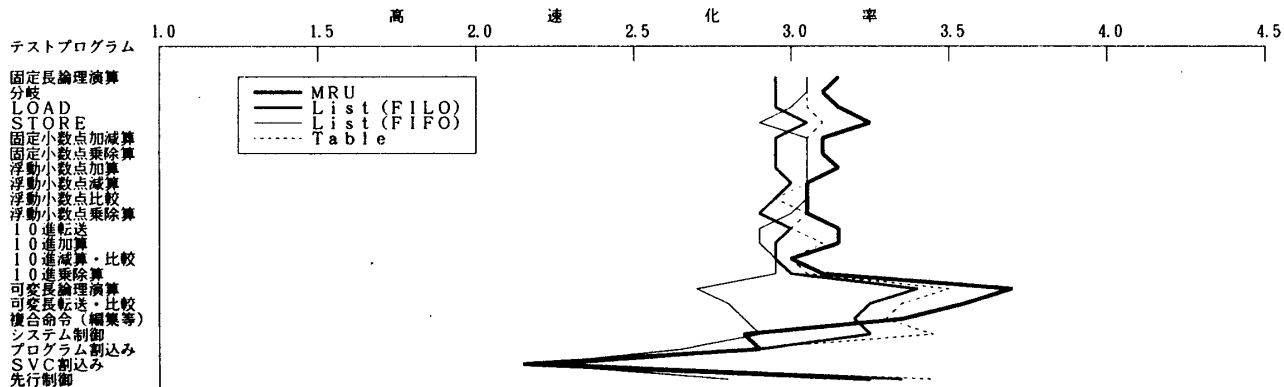


図2 高速化手法の評価結果 (各方式単独)

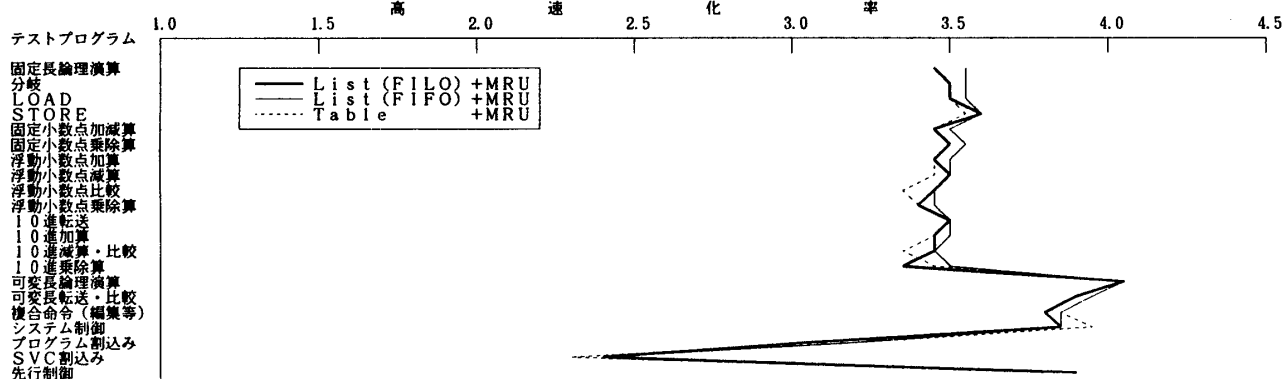


図3 高速化手法の評価結果 (MRU方式と他方式との組み合わせ)