5K-2

軽量でシンプルなマルチコアシミュレータの開発

佐野 伸太郎† 吉瀬 謙二†
† 東京工業大学 大学院情報理工学研究科

1 はじめに

プロセッサアーキテクチャの研究ツールとして様々なプロセッサシミュレータが利用されている．近年は多数のコアを搭載したマルチコアプロセッサが注目されているため，マルチコア，マルチスレッドに対応するプロセッサシミュレータが必要である．ここでマルチスレッド対応とは，シミュレーション対象のプログラムがマルチスレッドであることを指す．

本研究では軽量なマルチコアプロセッサシミュレータの開発を行う．開発するシミュレータは pthread に対応し，様々なベンチマークを実行することが出来る．本論文では MIPS シミュレータ SimMips[1] をベースにマルチコア対応に必要となった実装について記述し，実際に動作したベンチマークを示す．

2 シミュレータの実装

MIPS システムシミュレータ SimMips[1] をマルチスレッドに対応させるために必要となった機能について記述する．SimMips はシングルスレッドのシミュレータであり，一部の Linux システムコールに対応している．

作成するシミュレータはキャッシュを持たないマルチコアシステムであり，1 コアで 1 スレッドが動作するモデルとする．対応するマルチスレッドライブラリは Native POSIX Thread Library(NPTL) である．

2.1 Load-Link 命令と Store-Conditional 命令

スレッドの同期に必要な Load-Link(LL) 命令と Store-Conditional(SC) 命令について記述する．Load-Link 命令と Store-Conditional 命令の間では，あるメモリ領域が他のスレッドによって書き換えられていないことを保証する．使用方法を下に示す．

```
1: LL   $2, 0x400
2: ADDI $3, $2, 1
3: SC   $3, 0x400
```

この例では 1 行目の LL 命令で 0x400 番アドレスの値を 2 番レジスタにロードする．LL 命令では格納した値を保存する．2 行目で 2 番レジスタの値を 1 つインクリメントし 3 番レジスタに格納する．3 行目の SC 命令で 3 番レジスタの値を 0x400 番アドレスにストアするが，このとき LL 命令で保存した値と 0x400 番アドレスの値が異なっている場合，SC 命令は失敗し，ストア予定のレジスタの値を 0 にする．失敗した場合，他のスレッドが同じアドレスを操作していることを示す．

シミュレータで LL 命令と SC 命令を実装する方法について記述する．LL 命令の実装のためにレジスタを追加する．このレジスタはリンクレジスタと呼ばれ，LL 命令によってロードされた値を格納する．SC 命令はリンクレジスタとストア先の値が同じであれば，新しい値をストアする．値が異なっている場合，ストア予定のレジスタ（上記の例では$3）に 0 を格納する．

2.2 スレッド局所記憶の実装

スレッドごとのグローバル変数を定義するためにスレッド局所記憶（TLS）と呼ばれる機能が存在する．GCC では__thread として宣言する．

```
__thread int tls;
void func() {
    printf(‘‘%d’’, tls);
}
```

__thread で宣言された変数はスレッドごとに異なるアドレスとなる．

TLS 変数にアクセスするために C ライブラリはカーネルから TLS 空間の先頭アドレスを取得し，そこからの相対アドレスでアクセスする．このため，スレッドごとに TLS 空間の先頭アドレスの保持と参照用の命令が必要となる．また TLS 空間は初期化が可能なため，MIPS バイナリから TLS の初期値を取得する必要がある．

TLS 先頭アドレスの設定はシステムコール set_thread_area によって行われる．シミュレータでは，引数で指定された TLS 先頭アドレスを記憶しておく．

TLS 空間の先頭アドレスをプログラムが参照するために，rdhwr 命令を実装する．rdhwr 命令のフォーマットを示す．

```
RDHWR rt, rd
```

rd で指定されたレジスタの値が 29 であった場合 TLS の先頭アドレスを rt レジスタに返す．

TLS 空間は初期化可能である．初期化は C ライブラリが行うが，初期化のためのデータ取得がやや特殊であ

Development of Lightweight and Simple Multi-core Simulator
Shintaro SANO†, and Kenji KISE†
†Tokyo Institute of Technology

```
Program Header:
    .....
     TLS off    0x0001e9f0 vaddr 0x0042e9f0 paddr 0x0042e9f0 align 2**3
         filesz 0x00000004 memsz 0x00000c64 flags r--
    ....
```

図 1: Pthread プログラムのプログラムヘッダ

る．TLS の初期値がどこに存在するかはバイナリのプログラムヘッダに記述してある．図 1 に，objdump で出力した pthread プログラムのプログラムヘッダを示す．C ライブラリはこのプログラムヘッダに書かれた情報から TLS 空間の初期化を行う．このため，C ライブラリは TLS 空間の初期化のために，プログラムヘッダ位置の取得を行う．C ライブラリはプログラムヘッダ位置をプログラム開始時のスタックから取得する．

シミュレータはプログラム開始時にこのプログラムヘッダの位置をスタックに積む必要がある．プログラム開始時のスタックは，プログラムの引数 (argv)，環境変数，ELF 情報の順に積む必要がある．シミュレータはこの ELF 情報の項目に，プログラムヘッダのアドレス，プログラムヘッダのエントリーサイズ，プログラムヘッダのエントリー数の格納を行う．

2.3 システムコール clone

スレッド生成のためのシステムコール clone について説明する．システムコール clone を呼ぶと，そのコアのレジスタの内容を他のコアにコピーする．また，clone の引数としてスタックポインタ情報と TLS アドレスが渡されるため，それらの情報を他のコアに設定する．

2.4 システムコール futex

スレッドを待機させるためのシステムコール，futex について説明する．

futex に FUTEX_WAIT を指定して実行した場合，そのスレッドは待機状態に入る．ただし，引数に指定されているアドレスの値が，引数に指定された値と異なる場合は，システムコールはエラーを返し待機しない．

futex に FUTEX_WAKE を指定して実行した場合，待機状態のスレッドを起こす．起こすスレッドの数は引数で指定する．

3 評価

作成したマルチコアシミュレータの評価を行う．評価はシミュレータのコード行数と動作したベンチマークを用いる．

まず，作成したマルチコアシミュレータのコード行数を示す．コード行数はコメントや空行を含んだ行数で 4,108 行であった．このコード行数は SimMips のすべての機能を含んだ行数である．これはシステムシミュレータとしてはとても少ないコード行数である．作成の基礎となった SimMips のコード行数は約 4,500 行 [1] であるが，これよりも少なくなっている．これはリファクタリングを行ったためである．

次に，動作したベンチマークを示す．動作検証には NAS Parallel Benchmarks3.3 に含まれる OpenMP によって記述されたベンチマークを使用する．スレッド数はすべて 16 スレッドを使用する．ベンチマークサイズは CLASS S である．また，MIPS バイナリの作成に Buildroot を使用する．Buildroot のバージョンは 2011.11 である．今回使用する Buildroot では linux-3.1.4 カーネルを使用している．マルチスレッドライブラリは NPTL である．

作成したシミュレータによって，NAS Parallel Benchmarks の BT，CG, EP，FT，IS，MG，SP ベンチマークが正しく動作した．LU ベンチマークは動作しなかった．動作検証をした 8 種のうち 7 種という多くのベンチマークの動作を確認できた．

4 おわりに

本論文ではマルチコアシミュレータ開発について記述した．開発したシミュレータは pthread に対応し，様々なベンチマークを実行できる．NAS Parallel Benchmarks を実行したところ 8 種中 7 種のベンチマークの実行に成功した．

謝辞

本研究の一部は，科学技術振興機構・戦略的創造研究推進事業 (CREST) 「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による．

参考文献

[1] 藤枝直輝, 渡膳伸平, 吉瀬謙二. 教育・研究に有用な MIPS システムシミュレータ SimMips. 情報処理学会論文誌, Vol. 50, No. 11, pp. 2665–2676, 2009-11-15.