

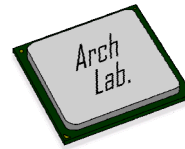
計算機アーキテクチャ 第二 (O)

10. コンピュータシステム

1



SimMips: 教育・研究に有用な
Linuxが動く5000行のMIPSシステムシミュレータ



○藤枝 直輝(東工大)
渡邊 伸平(東工大)
吉瀬 謙二(東工大)

ComSys2008 2008/11/13

Agenda

3

- ◆ 開発の背景
- ◆ SimMipsの概要
- ◆ デモンストレーション
- ◆ SimMipsの実装
- ◆ 評価 - シミュレーション時間 -
- ◆ 応用例 - メニーコアシミュレータへの組み込み -
- ◆ まとめ

ComSys2008 2008/11/13

開発の背景

4

- ◆ コンピュータシステムの複雑化
 - ▶ システムシミュレータも同様
 - ▶ シンプルで扱いやすいシステムシミュレータへの要求
 - ◆ プロセッサの高速化
 - ▶ コードをシンプルに保ちつつ現実的な速度でシミュレーションが可能
- ↓
- ◆ シンプルさと可読性を重視した
MIPSシステムシミュレータSimMips

ComSys2008 2008/11/13

SimMipsの概要

5

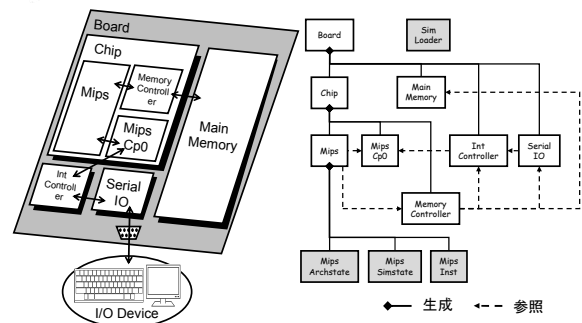
- ◆ 命令セット: MIPS32 R2のサブセット
 - ▶ 実装命令数は約100種
(浮動小数点命令を除くほぼ全て)
- ◆ コード量: 4,422行(v0.5.0)
 - ▶ コメント, 空行を含む
- ◆ 主要部分(~v0.4.0)の開発期間: 約2ヶ月半
 - ▶ 開発は学生2名
 - ▶ 休日などを含む

```
744 define.h
21 main.cc
622 board.cc
297 memory.cc
227 simloader.cc
899 mips.cc
767 mipsinst.cc
309 cp0.cc
536 device.cc
4422 total
```

ComSys2008 2008/11/13

SimMipsのモデル(v0.5.0)

6



ComSys2008 2008/11/13

SimMipsの2つのモード

7

- ◆ Appモード (Applicationモード)
 - ▶ 静的リンクされたユーザプログラムを実行
 - ▶ プロセッサシミュレータとしての利用
- ◆ OSモード
 - ▶ OSカーネルを動作させるモード
 - ▶ あらかじめカーネルやRAMディスクがロードされた状態を想定
 - ▶ システムシミュレータとしての利用

ComSys2008 2008/11/13

Demonstration

8

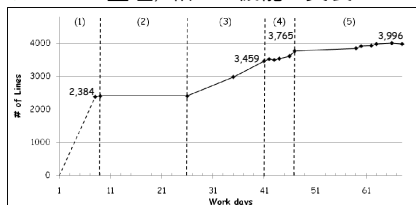
- ◆ Appモードのデモ(2種類)
- ◆ OSモードのデモ

ComSys2008 2008/11/13

開発の段階

9

1. 計算コア部分(Appモード)の実装
2. 制御コア(CPO)の機能について調査
3. CPO, 割り込み, シリアル(OSモード)の実装
4. デバッグと検証
5. コードの整理, 細かい機能の実装

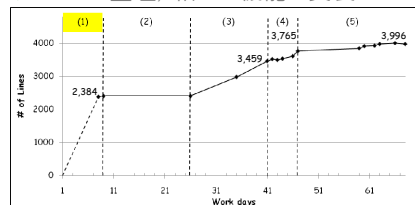


ComSys2008 2008/11/13

開発の段階

10

1. 計算コア部分(Appモード)の実装
2. 制御コア(CPO)の機能について調査
3. CPO, 割り込み, シリアル(OSモード)の実装
4. デバッグと検証
5. コードの整理, 細かい機能の実装

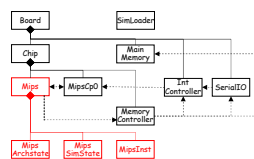


ComSys2008 2008/11/13

Mipsクラス

11

- ◆ MIPSの計算コア部分
- ◆ 可読性のため, ハードウェアにはないクラスを追加
 - ▶ レジスタファイルなどはMipsArchstateクラス
 - ▶ 統計情報(命令ミックスなど)はMipsSimstateクラス
 - ▶ 命令情報(デコード結果など)はMipsInstクラス



ComSys2008 2008/11/13

プロセッサのステージを意識した構成

12

```
int Mips::step_func()
{
    ...
    fetch();
    decode();
    regfetch();
    execute();
    if (inst->attr &
        LOADSTORE) {
        memsend();
        memreceive();
    }
    writeback();
    setnpc();
    return ...;
}
```



ComSys2008 2008/11/13

フェッチ、デコード

13

```
inline void Mips::fetch()
{
    ...
    mcid = mc->enqueue(addr, 4, NULL);
    ...
}

inline void Mips::decode()
{
    ...
    inst->ir = mc->inst[mcid].data032;
    inst->decode();
    ...
}

addu rd,rs,rt
opcode      funct
0  rs  rt  rd  0  33
31 26 25 21 20 16 15 11 10 6 5 0
```

```
void MipsInst::decode()
{
    opcode = (ir >> 26) & 0x3f;
    funct = ir & 0x3f;
    ...
    switch (opcode) {
    case 0:
        switch (funct) {
        case 33:
            op = ADDU;
            attr = READ_RS | READ_RT | WRITE_RD;
            break;
        ...
        }
    }
}
```

ComSys2008 2008/11/13

レジスタフェッチ、実行、ライトバック

14

```
inline void Mips::regfetch()
{
    ...
    if (inst->attr & READ_RS)
        rrs = as->r[inst->rs];
    if (inst->attr & READ_RT)
        rrt = as->r[inst->rt];
    ...
}

inline void Mips::execute()
{
    switch (inst->op) {
    ...
    case ADDU:
        rrd = rrs + rrt;
        break;
    ...
    }
}

inline void Mips::writeback()
{
    ...
    if (inst->attr & WRITE_RD)
        as->r[inst->rd] = rrd;
    ...
}

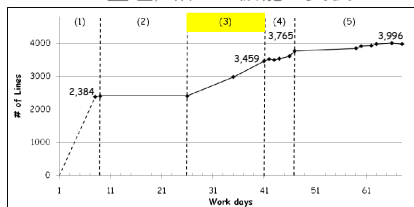
addu rd,rs,rt
⇔ r[rd] = r[rs] + r[rt];
0  rs  rt  rd  0  33
31 26 25 21 20 16 15 11 10 6 5 0
```

ComSys2008 2008/11/13

開発の段階

15

1. 計算コア部分(Appモード)の実装
2. 制御コア(CPO)の機能について調査
3. CPO, 割り込み, シリアル(OSモード)の実装
4. デバッグと検証
5. コードの整理, 細かい機能の実装



ComSys2008 2008/11/13

Linuxの動くシミュレータにするために

16

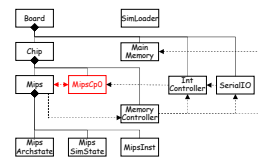
- ◆ 制御コアCPO(Coprocessor Zero)の実装
 - ▶ 例外
 - ▶ TLB(Translation Lookaside Buffer)
 - ◇ MIPSではソフトウェアによる制御
 - ▶ アドレス変換
- ◆ 割り込みコントローラの実装
- ◆ シリアル入出力の実装

ComSys2008 2008/11/13

MipsCp0クラス

17

- ◆ 例外の取り扱い
 - ▶ 制御レジスタに例外に関する情報を記録
 - ▶ 例外ハンドラ(通常0x80000180)から実行再開
- ◆ TLBとアドレス変換
 - ▶ TLB<->制御レジスタ間データ授受命令の実装
 - ▶ 仮想ページ番号(VPN)でTLBエントリを検索
 - ◇ エントリなし→TLBミス例外
 - ◇ 有効ビット無→ページフォルト

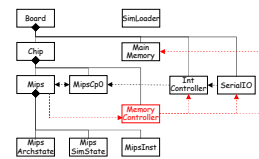


ComSys2008 2008/11/13

MemoryControllerクラス

18

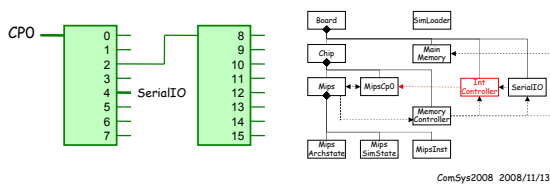
- ◆ ロード・ストアは必ずこのクラス経由
- ◆ アドレスを見て適切なクラスへのリード・ライトを行う



ComSys2008 2008/11/13

IntControllerクラス

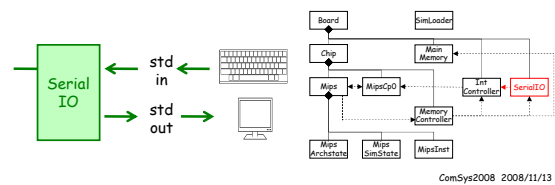
- ◆ 割り込みコントローラ
 - ▶ Intel 8259相当(x2)
 - ▶ 接続デバイス(現在はSerialIOのみ)から割り込みを受け取り、CPOへ送る



ComSys2008 2008/11/13

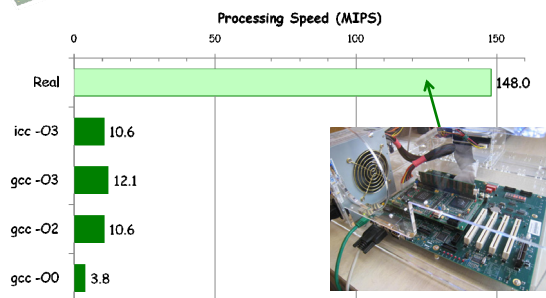
SerialIOクラス

- ◆シリアル接続のI/Oコントローラ
 - ▶ ns 16550相当
 - ▶ 入力はキーボード(標準入力)から受け取り, 出力はコンソール(標準出力)へ



ComSys2008 2008/11/13

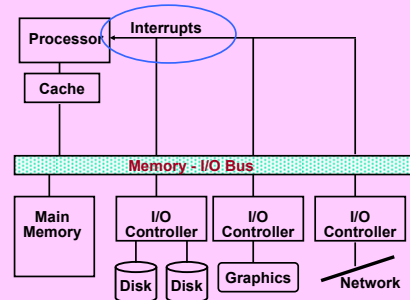
SimMipsのシミュレーション時間



※Xeon X5365(3.0GHz), gcc 4.1.2, icc 10.1
ベンチマークはクイックソートを使用

ComSys2008 2008/11/13

I/O Systemの利用方法と割り込み



22

Communication of I/O Devices and Processor

- How the processor directs the I/O devices
 - **Memory-mapped I/O**
 - Portions of the high-order memory address space are assigned to each I/O device
 - Read and writes to those memory addresses are interpreted as commands to the I/O devices
 - Load/stores to the I/O address space can only be done by the OS
 - **Special I/O instructions**

Communication of I/O Devices and Processor

- How the I/O device communicates with the processor
 - **Polling** – the processor periodically checks the status of an I/O device to determine its need for service
 - Processor is totally in control – but does **all** the work
 - Can waste a lot of processor time due to speed differences
 - **Interrupt-driven I/O** – the I/O device issues an interrupts to the processor to indicate that it needs attention

SimMipsを用いたメニーコア研究

25

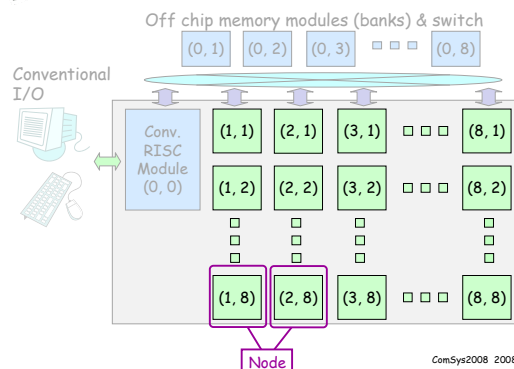
- ◆ マルチコア・メニーコア
→ 近年の高性能汎用プロセッサのトレンド
- ◆ シミュレータを短期間で構築することが重要
- ◆ メニーコアシミュレータSimMc※のコア部分にSimMips(のAppモード)を使用
 - ▶ SimMips側への変更は行わず
 - ▶ 短期に、かつネットワーク部分に注力した開発が可能に

※植原ほか: シンプルで効率的なメニーコアアーキテクチャの開発, 情報処理学会研究報告 2008-ARC-180 (Oct. 2008)

ComSys2008 2008/11/13

アーキテクチャモデル

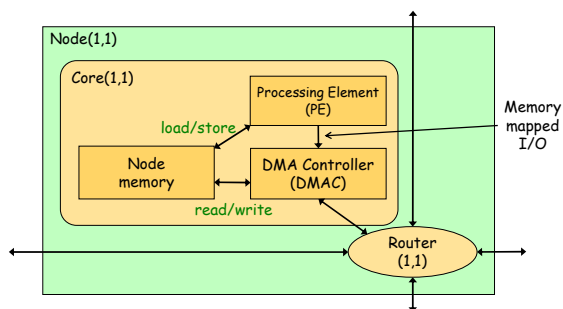
26



ComSys2008 2008/11/13

ノードの構成

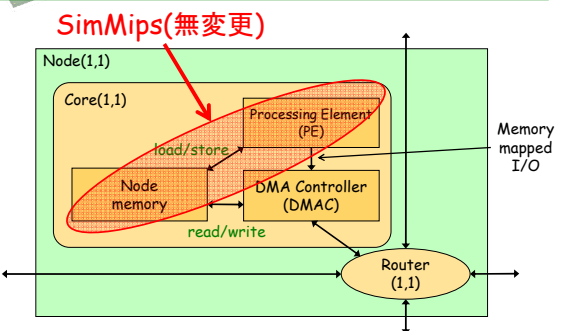
27



ComSys2008 2008/11/13

ノードの構成

28



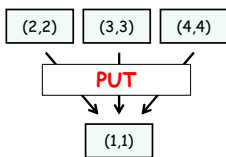
ComSys2008 2008/11/13

SimMcの可視化ツール

29

- ◆ チップ内ネットワークを流れるフリットの視覚化

例)



ComSys2008 2008/11/13 29

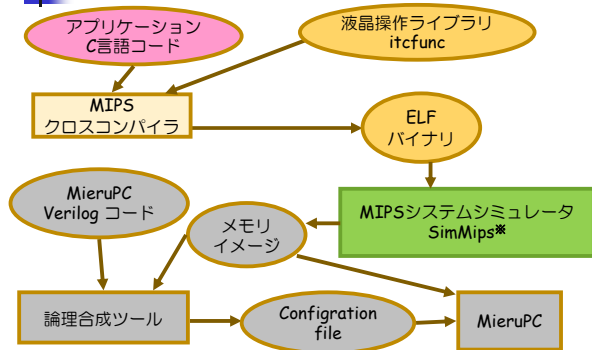
まとめと課題

30

- ◆ シンプルで可読性の高いMIPSシステムシミュレータSimMipsを開発
 - ▶ 教育・研究への有用性
- ◆ 本日使用したバージョンは <http://www.arch.cs.titech.ac.jp/SimMips/>にて公開中
- ◆ 今後の課題
 - ▶ 機能追加: ネットワークなど... (→v1.0)
 - ▶ OS等を含めた包括的な教育・研究プラットフォームの構築

ComSys2008 2008/11/13

アプリケーションの開発フロー



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

マルチレベル・ストライド値予測機構による
命令レベル並列性の向上
(JSPP 1999)

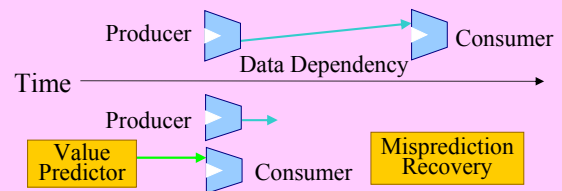
研究の背景

- 真のデータ依存関係が命令レベル並列性を制限
- 生産者から消費者へのデータの流れを解消する技術として値予測

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

研究の背景

- 真のデータ依存関係が命令レベル並列性を制限
- 生産者から消費者へのデータの流れを解消する技術として値予測



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

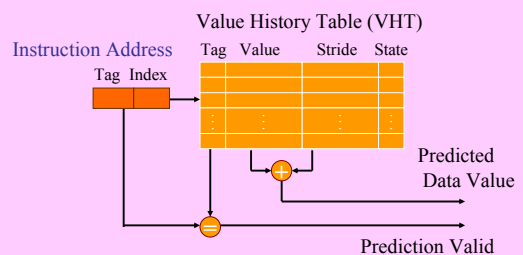
関連研究: 値生成のアルゴリズム

- Last-value予測
 - 最も近い過去に得られた値を予測値
- ストライド値予測
 - 最も近い過去に得られた2回の値の差分 Stride と、Last-value の和を予測値
- 2レベル値予測
 - 過去のn個の履歴の中からひとつを選択
- ハイブリッド値予測
 - 複数のアルゴリズムから選択

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

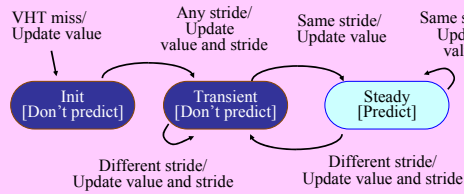
ストライド値予測機構

Predicted Value = Last-value + Stride



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

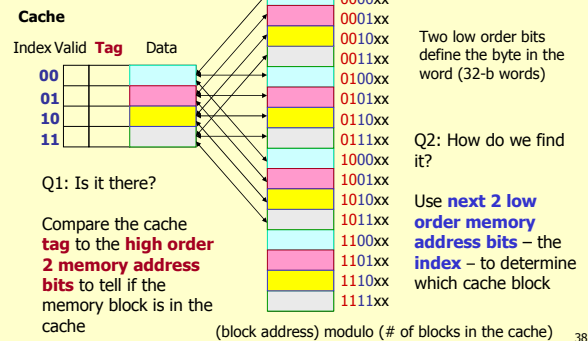
ストライド値予測機構 (cont.)



Stateフィールドの推移と予測アルゴリズム

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

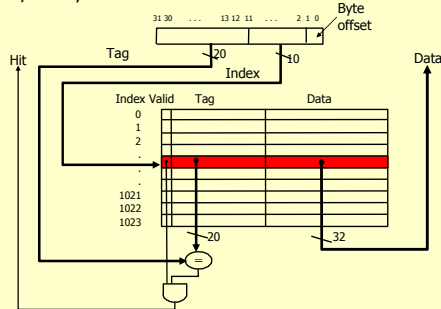
Caching: A Simple First Example



38

MIPS Direct Mapped Cache Example

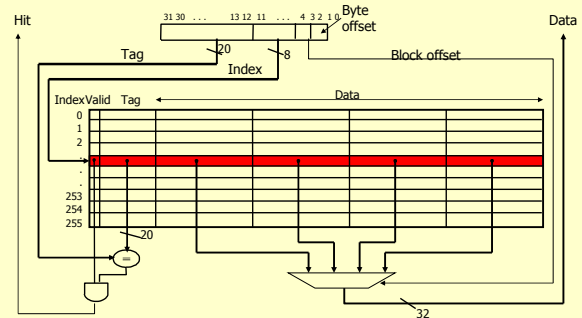
- One word/block, cache size = 1K words



39

Multiword Block Direct Mapped Cache

- Four words/block, cache size = 1K words



40

課題: SimMipsプロセッサシミュレータ

- 004_rand, 050_tokei を動作させ、正しく動作していることを確認せよ。
- データ値予測 (last-value predictor) の予測精度を測定する仕組みを追加し、予測精度を測定せよ。レジスタに値を書く命令が対象。
 - e40m 004_rand.mex
 - e40m 050_tokei.mex
- データキャッシュのヒット率を測定する仕組みを追加し、ヒット率を測定せよ。
 - ダイレクトマップ方式、ラインサイズは4ワードとする。
 - セット数を8, 16, 32, 64, 128, 256に変更した場合のヒット率を示せ。
 - このキャッシュのヒット率を改善する任意の方式を実装し、その効果を示せ。
- MieruPC用の面白いアプリケーションを作成せよ。その魅力を示せ。
 - ソースコードは各自のディレクトリに格納すること。
 - MieruPC用のアプリケーションとして利用させてください。

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

課題: SimMipsプロセッサシミュレータ

- 1月25日の講義開始時にレポートを提出 (Intel系のプロセッサとLinuxの利用を推奨)
 - 追加したコードの主要部分の記述と説明
 - 実行結果はただしいか? その根拠を示せ。
 - それぞれの課題にどれくらいの時間を必要としたか?
 - 感想, 苦労した点など。
 - SimMipsに対する要望など。
- 質問などは
 - kise at cs.titech.ac.jp まで

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

SimMips-0.5.5 の使い方など

```

window-1
[archo@nc440 ~/linux]$ tar xzf ../nc2008_sample_app_01.tgz
[archo@nc440 ~/linux]$ tar xzf ../SimMips-0.5.5.tgz
[archo@nc440 ~/linux]$ cd nc2008_sample_app/
[archo@nc440 nc2008_sample_app]$ ls
001_hello 004_rand 005_keycode 030_hall 051_tokai  README.txt
002_color 010_timer 021_keycode 031_hall 052_tokai  lib
003_color 011_timer 022_keycode 050_tokai 060_dairagon
[archo@nc440 nc2008_sample_app]$ cd 001_hello/
[archo@nc440 001_hello]$ make
*linux-libm=uclibc-gcc -Wall -static -Os -nostdlib -I../lib main.c ../lib/
*linuxlib.o -T ../lib/Makefile.script -o 001_hello.aex
/home/share/SimMips/bin/mangen -b 001_hello.aex 52 > 001_hello.bin
[archo@nc440 001_hello]$ make run

```

```

window-1
Hello, world!

```

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

43

アナウンス

- 講義スライド, 講義スケジュール
 - www.arch.cs.titech.ac.jp
- 講義用の計算機
 - 131.112.16.56 (情報工学科の演習室からは入れません)
 - ssh arco@131.112.16.56
 - mkdir myname
 - cd myname

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

44