## ■ JRuby で親クラスのメソッドを呼ぶ

JRuby でオーバライドして定義ししたメソッドで,
親クラスのメソッドを呼ぶときは，単に super( 引数 ) でいいみたい.
Java でよく書く,

```
paintComponent(Graphics g){
  super.paintComponent(g);
}
```

みたいなのは,

```
def paintComponent(g)
  super(g)
end
```

になる.

## ■ Improving Program Efficiency by Packing Instructions into Registers

ISCA '05
Harnessing Horizontal Parallelism and Vertical Instruction Packing of Programs to Improve System Overall Efficiency の IRF 的な親論文.
良く使う命令をレジスタにおいておくことで効率的にアクセスするというのが目的.
IRF(instruction register file) と IMM(immidate table) を定義している.

## ■結果

- IF の後部分か ID の前部分にいれると，クリティカルパスには影響しない
- 平均 19% のコードサイズの削減
- 32 エントリの IRF を使うことで，平均 37% の I-Fetch エネルギーを削減
  - 全体では 15%

## ■ ISA の拡張

IRF の命令を使うために ISA を拡張する

Loosely Packed Instruction

- 通常の MIPS コードの下位 5bit を使って IRF を埋めこむ
- IRF を実行した後に，MIPS コードを実行する
- IRF が呼べない時には，IRF 部分を nop にしておけば通常の MIPS コードになる
  - shamt は，rs と共用
  - lhi では，16bit 即値が読めなくなる
  - lui で，前の命令の rs と 16bit 即値で 21bit を表現．(lui には IRF は埋め込めない )

Thigtly Packed Instrcutoins

- opcode に続けて最大 5 個の IRF を呼びだす.
- 命令コードのうち即値だけが違うものは，IMM へのポインタと組み合わせる
  - IMM へのポインタ (param) は最大 2 個．この時 IRF は最大 3 個

Positional Register

・組み合わせることでより効率があがる

## ■ Compiler の変更

・ the compiler is a part of VPO for the MIPS architecture
・ performance statistics を取得するために SimpleScalar の PISA にも使えるように
・ GAS の pseudoinstructions で簡単に

IRF の選択

・ profiling the application
・ selecting the most frequently ocurring
・ the top 32 immediate values を求めて IMM に登録.