###ファイル構成 <アセンブリ> `assembler.cpp`に実装 アセンブリを0,1の機械語に変換

<シミュレータ> **実行の仕方** ＜エミュレート入り＞ 1.「エミュレート修正_完動verフォルダ」 or 「最終完動ver2.zip」をダウンロード

2.Make sim

3. ./sim -g

実行の際、最初にファイルを一旦読み込んだ時点でデバッグ用のprint文（ラベルの中身や即値）が流れる それが終わり何も表示されなくなると実際の処理を行っている。

実行が完了すると、統計情報や実行命令数・クロック数・実行予測時間（コアでかかる時間予測）などが出力される。

※ ./sim -fで実行するとfast_mode; 統計情報をとらないため 256*256でも183.75sec.程で実行が完了する

./sim -gでは ストールなども含めた実行命令数を出力 hit,miss率も出す

１ファイルで実行する場合 最初の方のbool debug = true;をfalseに変えると速くなる。trueにすると色々表示される。

デバッグ機能 *がある行で s or n or mと出力される →sかnかmを入力

s:ステップ実行 n: 次の*まで飛ぶ m: レジスタの中身表示（有限の値のレジスタのみを出力します。） もしメモリが出力されて見にくい時は

```
for(int i = 0; i <= MEMORY_SIZE; i++){
                    if(M.at(i).f){
                        cout << "FM[" << i << "] " << M.at(i).f << endl;
                    }
                    if(M.at(i).i){
                        cout << "M[" << i << "] " << M.at(i).i << endl;
                    }
                }
```

を消して下さい

`sim_0206.cpp`; １つにまとめたファイル

画像サイズについて 現在minrt_inline.sは256から変更できませんがminrt_addj.sを入力ファイルとして使用すればl11023の測地を変えて使うことで画像サイズを変更して確認ができます。

実行時間予測 MakefileのSRCS１つ目を sim_for_coreに変えると実機で動いたものと同じアセンブリで実行時間を確認できます。

makeでコンパイル可能

```
–fをつけて実行するとfast_modeに
–gだとキャッシュなどの統計情報が出る
```

```
-dでデバッグモード
（1つのファイルであっても
bool debugやbool fast_modeを変えることによりモードを変えることができます）
```

＜エミュレートなし＞ `g++ sim_c++_改訂版.cpp -o sim_c++_改訂版` の後に `./sim_c++_改訂版` ど 別ファイルでも同様 （contest.sldファイルが必要になります） Ack関数結果

```
2 1024
5 8189
6 8188
hit miss 0 0
clean dirty 0 0
実行命令数 402260398
duration = 3.79298sec.
```

セグフォが出たら おそらくM[rs1+imm]のrs1+immの値が大きくなってMEMORY_SIZEを超えている

MEMORY_SIZEを4294967296などに変更すると実行開始までに３分ほどかかるようになるけどセグフォが解消することがある

ただrs1+immが大きすぎることで実行が停止しないことも：その場合はrs1+immの値が異常に大きくなっていないかチェックする必要がある

注意点

```
void sld_to_ppm(){
    string filename ("contest.dat"); //asm_3
    vector<string> lines;
    string line;
    string s1 = "/Users/maimai/my-3A/cpu-simu/sim_contest.ppm";
```

という部分はstring s1の名前の変更が必要 その他にも /Users/maimai/... という文字列を消して使う必要がある箇所があります。

入力ファイルcontest.dat(ppmファイルをdatファイルに変換したもの）を手元に置いてお使いください (base.datに書き換えられていたらbase.datを使用して下さい)

cppファイルと同じフォルダにdatファイルとアセンブリ(.s)ファイルを置いて実行

レジスタの使い方

```
x0...常に0を格納
x1...戻りアドレスを格納
x2...スタックポインタ
x3...ヒープポインタ
x5-x23...汎用レジスタ
x24...定数を一時的に格納するレジスタ
```

```
浮動小数点数用のレジスタ
f0-29... 汎用レジスタ
f30...ゼロレジスタ
f31...定数用レジスタ
```

x2の初期値はSIZE=1024

それ以外のレジスタは初期値0

'm'でレジスタ出力されたときに値が0のレジスタは出力されず、有限の値のレジスタのみを出力します。

＜FPUのエミュレート＞ TARGET = fpu_sim

(4)コンパイル対象のソースコード:make fpuの時（エミュレートの確認）

SRCS = fsqrt_sim.cpp SRCS += simu_fmul.cpp SRCS += simu_fadd.cpp SRCS += fpu_common.cpp

###RISC-V命令について 疑似命令も一般的なものは処理できる

デバッグ機能 -d: デバッグモード　-dの後に print mem: メモリの中身をprint ＊がついたアセンブリの行で break（一番最初の＊）

実行時の流れ 最初に一度ファイルを全て読み込んだ時点でpcの値などが流れて出力され、 次にファイルを読み込んで実際の処理をするときは、*がある行まではラベルのみが出力され

*の行にくるとその行が出力されるとともに実行が一時停止され、 s or n or m と出力されます。

sかnかmを入力して下さい。

更なる注意 ここにある仕様は一時的なもので、今後デバッグしていくうちに断りなく改変されることがあります。信じすぎないで困ったらコードを少し見るかすぐにslackなどで訊いて下さい。