1　マイクロプロセッサ実験簡易チュートリアル

1.1　実験に取りかかる前に…

以下の手順を行なって環境を整えましょう.

1. ログインする. (j から始まる実験室のアカウント)
2. 「~takamasa/pico_setup」と入力
3. ログアウト
4. もう一回ログイン
5. startx

これを行うことで, 必要なツールが使えるようになり実験に必要なファイルがコピーされます.
もう一度ログインし, 自分のホームをみてみると, pico16 というディレクトリができていると思います.
そのディレクトリで実験を行います.

2　ハードウェアの実装

2.1　最低限必要な実装

配布された ppico16.sfl は不完全であり, データハザード及び制御ハザードの実装がきちんとされていません.
そこで今回の実験では, これらの実装を行なってもらいます.
その実装を行なうために, 以下のスクリプトが用意されています.

ptest0.sct	なにも修正せずに動く.
ptest1.sct	データハザード対策をしないと動かない.
ptest1a.sct	ptest1.sct を無理やり動くように NOP をはさみこんだスクリプト
ptest2.sct	制御ハザード対策をしないときちんと動かない.

表 1: スクリプト

つまり, ptest1.sct と ptest2.sct がきちんと動くように ppico16.sfl を修正すれば, ハザード対策はできたことになります.
また, LDHI 命令も追加してください.

2.2　さらにハードウェアの命令を追加する場合

基本的に上述の実装が終われば, ソフトウェアの実装を行うことは可能ですが場合によっては, JAL, JR, JMP などのあると便利な命令を実装する場合があるかもしれません.
その場合は, ptest1.sct などのスクリプトファイルを真似して, 自分でテストファイルを作って実装します.
例えば, JMP 命令を実装する場合

```
% cp test2.sct jmp.sct (JMP 用のスクリプトファイルを作る)
% emacs jmp.sct
  IMEM の命令を書き換える.
  例えば,
  BEQZ r3, -4 を
  JMP -4
  などに書き換えてみる.
% seconds
  > jmp.sct
  > f
  > f
```

で JMP 命令がきちんと実装できているかどうかを確認したりします.

3 ソフトウェアの実装

ソフトウェアの実装の流れは以下のようになります.

1. アセンブラを書く.
2. upa.pl でスクリプトファイルに変換する.
3. sfl ファイルのロードスクリプトを加えてスクリプトファイルを作る.
4. seconds で実行してデバッグ
5. mssg.pl でアセンブラを IMEM のデータに書き換える.
6. ボード上で実行

※各処理の詳細は実験書参照.

3.1 サンプルプログラム

実装の流れをサンプルプログラムを実行しながら見ていきましょう.
※実行の前に, pico16.sfl に LDHI 命令を追加しましょう. (これは教科書などを見れば比較的簡単に実装することができます.)
サンプルプログラムのアセンブラプログラムは, echo.asm です. この echo.asm は, 押したボタンの文字をディスプレイに表示するプログラムです.
最初に, このアセンブラを sfl の IMEM データにおきかえるスクリプトにかけます.

```
% ./upa.pl echo.asm > echo.isct
```

できあがった echo.isct をみると, アセンブラが IMEM のデータに書き換えられていることがわかると思います.
そして, これを seconds で実行できるようにするために, echo.isct に, head.sct と foot.sct を追加します.

```
% cat head.sct echo.isct foot.sct > echo.sct
```

これで, test*.sct のような echo.sct ができあがりました.
これで, seconds のシミュレーションを行うことができます.

```
% seconds
SECONDS> echo.sct
```

このスクリプトでは,

1. キーが入力されるまでループする
2. キーが入力されるとキーのアスキーコードをディスプレイに表示する
3. (1) に戻る.

という処理を繰り返します.
なので, seconds でシミュレーションする時には,

1. IF がループ状態に入るまで「f」を連打
2. 「k3」などを入力 (キーの入力をシミュレーション)
3. ループ状態に入るまで「f」を連打
4. 「mem」でメモリの内容を確認する.
5. mem[0xff] (ディスプレイの左上の行) に値が入っていることを確認
6. (2) に戻る.

とやって, mem[0xff] に値が入っていれば, きちんと動作していることになります.
それでは, 次にボードで実際に動かしてみましょう.
とりあえず, ボード上に pico16 をロードするためのデータを作ります.

```
% ssh cad0
% cd pico16
% cd src
% make distclean
% make
```

10 分ほどして, pico16.bin ができあがっていれば成功です. cad0 から exit で抜けて, pico16.bin のあるところで

```
% seyon &
```

と入力して seyon を起動します.
そして, [configuration] というボタンを押してください. すると, ppico16.bin がボード上にロードされます.
続いて, IMEM のデータに変換し,

```
% ./mssg.pl -i echo.sct > echo.imem
```

これを, [imem] のボタンで, echo.imem と入力して, IMEM 上にロードするアセンブラのプログラムをロードします.
これで実機で動作することができます. [1]

1. CLK を連打
2. K5 などを入力
3. CLK を連打
4. ディスプレイに入力したキーが表示される.
5. (1) に戻る.

[1] DMEM データを使用する場合は ./mssg.pl -d xxx.sct > xxx.dmem と入力して .dmem ファイルを作成し, [dmem] のボタンでデータをロードします.

4　実験室以外での実装

今回の実験の実装は, 矢上 ITC などでも行なうことができます. その時に使用するコマンドなどについて説明します.

矢上 ITC ⇔実験室でのデータのやりとり

scp という他のマシーンにファイルをコピーするためのコマンドを使って矢上 ITC と実験室のデータのやりとりを行なうことができます.

- 実験室にいる時に, 実験室から矢上 ITC に pico16 をコピーする.

  ```
  scp -r pico16 y12345@hinode.educ.cc.keio.ac.jp:
  ```

- 実験室にいる時に, 矢上 ITC から実験室に pico16 をコピーする.

  ```
  scp -r y12345@hinode.educ.cc.keio.ac.jp:pico16 .
  ```

- 矢上 ITC にいる時に, 実験室の pico16 を矢上 ITC にもってくる.

  ```
  scp -r j021234t@cad0.std.expr.st.keio.ac.jp:pico16 .
  ```

- 矢上 ITC にいる時に, 矢上 ITC の pico16 を実験室に送る.

  ```
  scp -r pico16 j021234t@cad0.std.expr.st.keio.ac.jp:
  ```

※「:」は忘れずに必ずつけること.
※送り先に同じ名前のファイルがある時は上書きされるので, 注意すること.

矢上 ITC での実装

scp で実験室の pico16 を取得すれば, 矢上 ITC で seconds を用いて実装を行なうことができます.

```
% scp -r j021234t@cad0.std.expr.st.keio.ac.jp:pico16 .
% cd pico16
% cd src
% seconds
  > ptest.sct
  > f
  > f
```

4.1　よくわからない時は…

質問などがある時には, 気軽に, pico-ta@am.ics.keio.ac.jp にメールをしてください.