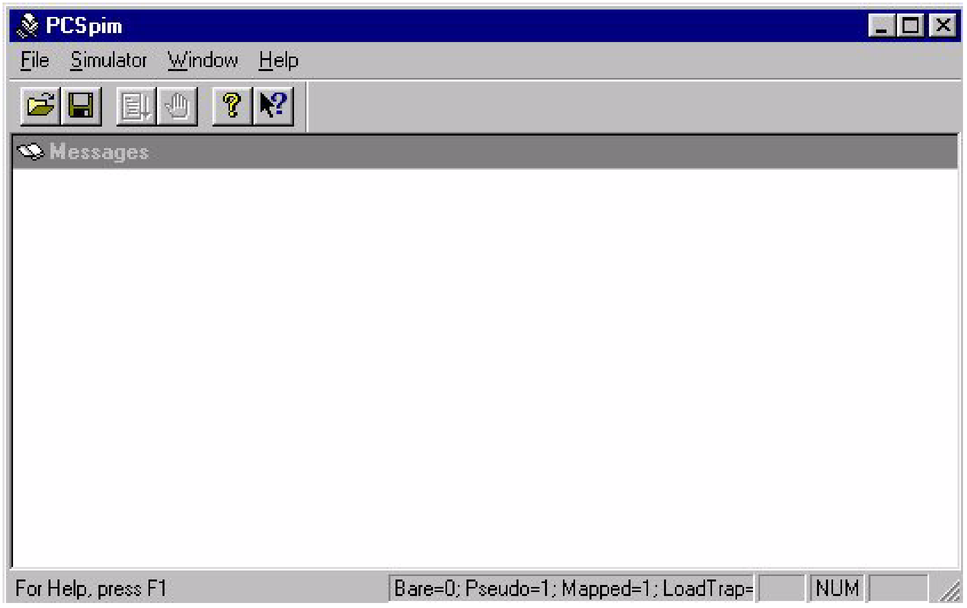
Welcome to PCSPIM Tutorial

1. Before you do anything, let's make a directory in your shared drive with the name lab230.
2. Since PCSPIM has no built in text editor, you must edit your source code in external editors. These editors are such as VI, PICO, EMACS (for UNIX lovers), or MS Notepad, MS Wordpad (for the rest of us). Remember to save your file as text only. Do not save your file as Word document or Rich Text.
   Let's do the tutorial.asm
   Using your favorite text editor, type in the following:

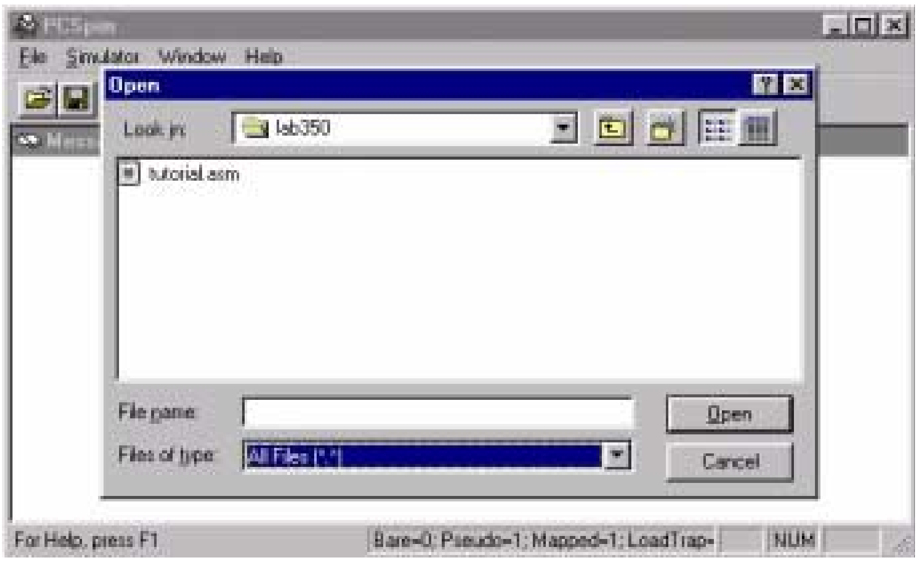
```
# tutorial.asm
                .data 0x10000000
msg1:           .asciiz "ABCDabcd"
                .text
main:            addu $s0, $ra, $0
                li $v0, 4
                la $a0, msg1
                syscall
                addu $ra, $0, $s0
                jr $ra
```

   (Warning! You must end an assembly file with a new line. This means that you must hit a carriage return after you type "jr $ra". If you forget, you'll get a compilation error on the last line.) Save this file as "tutorial.s" in your lab230 directory. This is your source code that will be used in this tutorial.

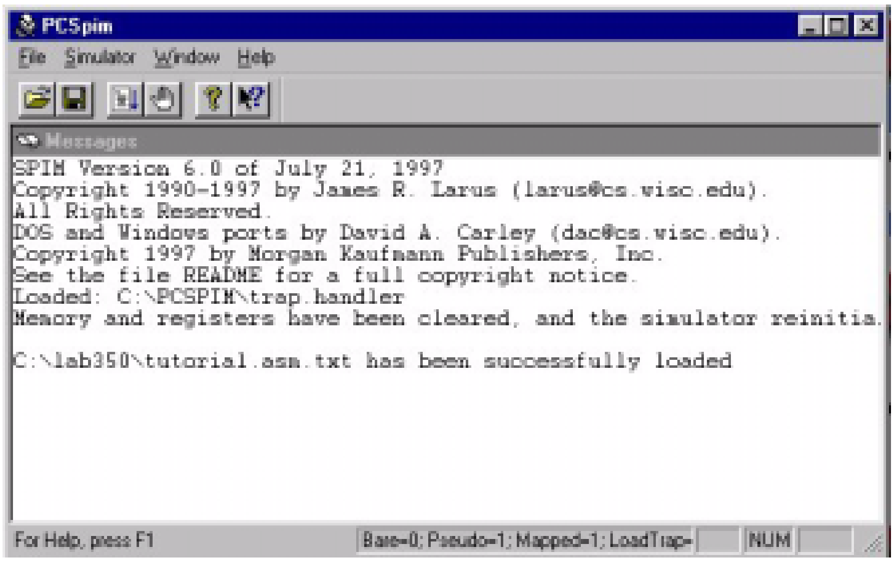

PCSpim
File Simulator Window Help
Messages
For Help, press F1
Bare=0; Pseudo=1; Mapped=1; LoadTrap=
NUM

3. Once you have your source code edited, you can start PCSPIM. The following figure is the PCSPIM interface. Pay attention to every item on the menu bar. Most likely, we will use every option available in this course.
4. Let's load in the file. Since we create a "lab230" directory, your source code (tutorial.asm)

should be in there. Go to "lab230" and select "tutorial.asm"

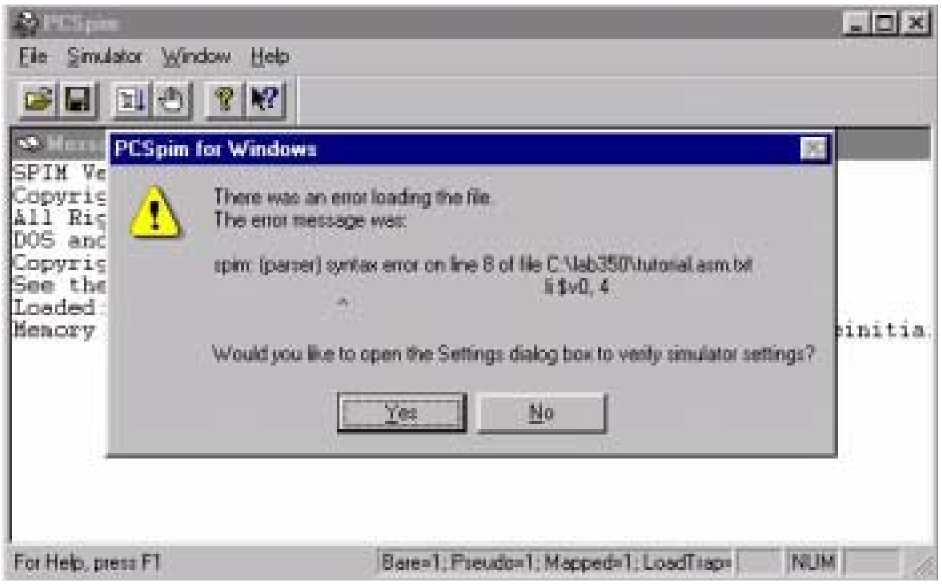

PCSpim
File Simulator Window Help
Open
Look in: lab350
tutorial.asm
File name:
Files of type: All Files (*.*)
Open
Cancel
For Help, press F1
Bare=0; Pseudo=1; Mapped=1; LoadTrap=
NUM

If your source code is correct, PCSPIM will display the following message.


PCSpim
File Simulator Window Help
Messages
For Help, press F1
Bare=0; Pseudo=1; Mapped=1; LoadTrap=
NUM

```
SPIM Version 6.0 of July 21, 1997
Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
DOS and Windows ports by David A. Carley (dac@cs.wisc.edu).
Copyright 1997 by Morgan Kaufmann Publishers, Inc.
See the file README for a full copyright notice.
Loaded: C:\PCSPIM\trap.handler
Memory and registers have been cleared, and the simulator reinitia

C:\lab350\tutorial.asm.txt has been successfully loaded
```

However, if there were any error with your source code the following screen would appear.

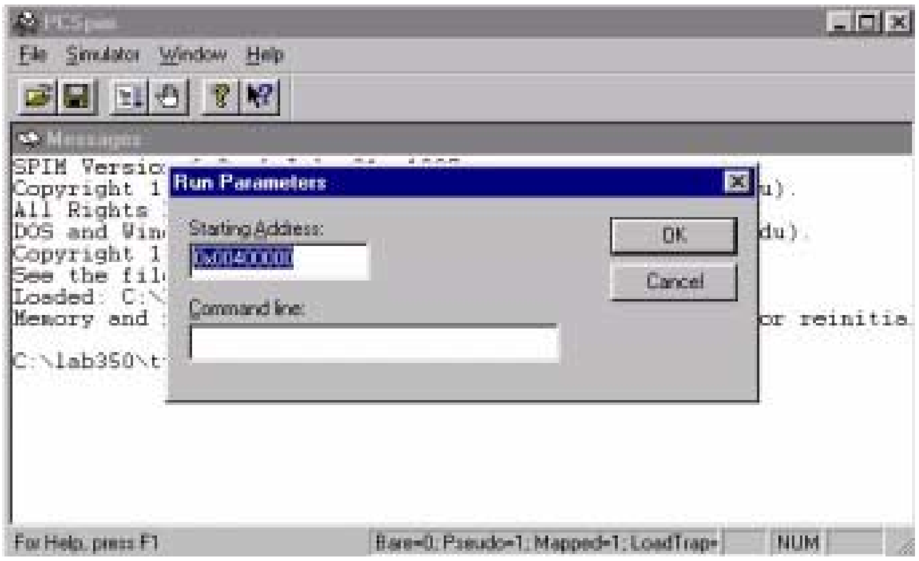

PCSpim
File Simulator Window Help
PCSpim for Windows
Yes
No
For Help, press F1
Bare=1; Pseudo=1; Mapped=1; LoadTrap=
NUM

```
There was an error loading the file.
The error message was:

spim: (parser) syntax error on line 8 of file C:\lab350\tutorial.asm.txt
                                  li $v0, 4
        ^

Would you like to open the Settings dialog box to verify simulator settings?
```

Click "NO". Go back and correct your source code using the text editor and reload the file again.

5. Once you have successfully loaded the file, we have several options. Let's first take a look at the simulator menu. By clicking the simulator menu you'll have the following option.
    - Clear registers
    - Reinitialize
    - Reload
    - Go
    - Break
    - Single step
    - Multiple step
    - Breakpoints
    - Set value
    - Display symbol table
    - Settings

   At this point, we would like to select GO. In doing so, we will execute the program. PCSPIM automatically picks the appropriate starting address for you. Therefore, you should not have to make any changes to the following setting. Just click OK.

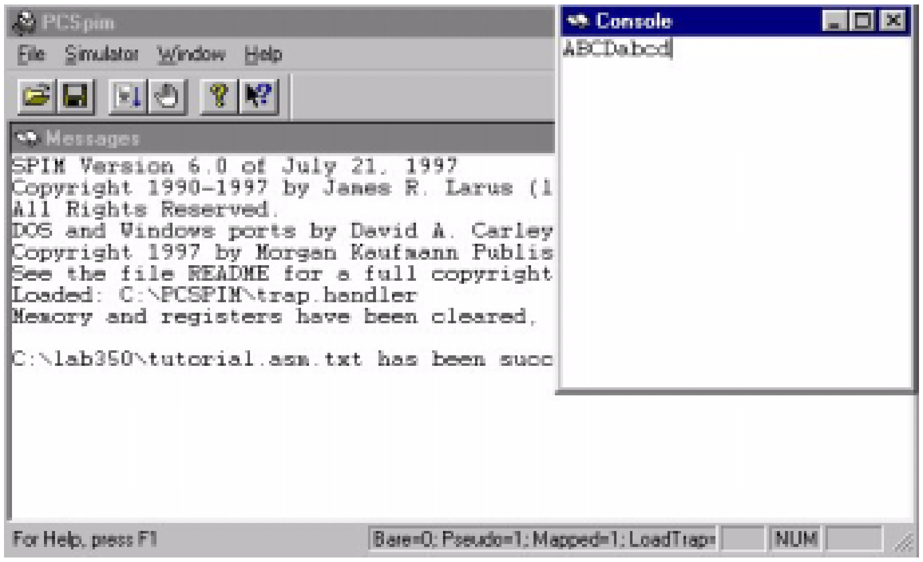

PCSpim
File Simulator Window Help
Messages
Run Parameters
OK
Cancel
For Help, press F1
Bare=0; Pseudo=1; Mapped=1; LoadTrap=
NUM

Starting Address: 0x00400000

Command line:

```
SPIM Versio
Copyright 1                                       u).
All Rights
DOS and Win                                       du).
Copyright 1
See the fil
Loaded: C:\
Memory and                                        or reinitia

C:\lab350\t
```

"Console window" is where you can enter data and monitor the program output. The following figure is what the console looks like.


PCSpim
File Simulator Window Help
Messages
Console
For Help, press F1
Bare=0; Pseudo=1; Mapped=1; LoadTrap=
NUM

```
SPIM Version 6.0 of July 21, 1997
Copyright 1990-1997 by James R. Larus (l
All Rights Reserved.
DOS and Windows ports by David A. Carley
Copyright 1997 by Morgan Kaufmann Publis
See the file README for a full copyright
Loaded: C:\PCSPIM\trap.handler
Memory and registers have been cleared.

C:\lab350\tutorial.asm.txt has been succ
```

```
ABCDabcd
```

Console can be resized to your preference. We strongly suggest that you move your console to a place where you can easily access the main interface menu bar.
Single step option allows you to execute the program one instruction at a time. Multiple step option allows you to execute specific number of instructions at a time. Other options will be explained to you later as you become more experienced.

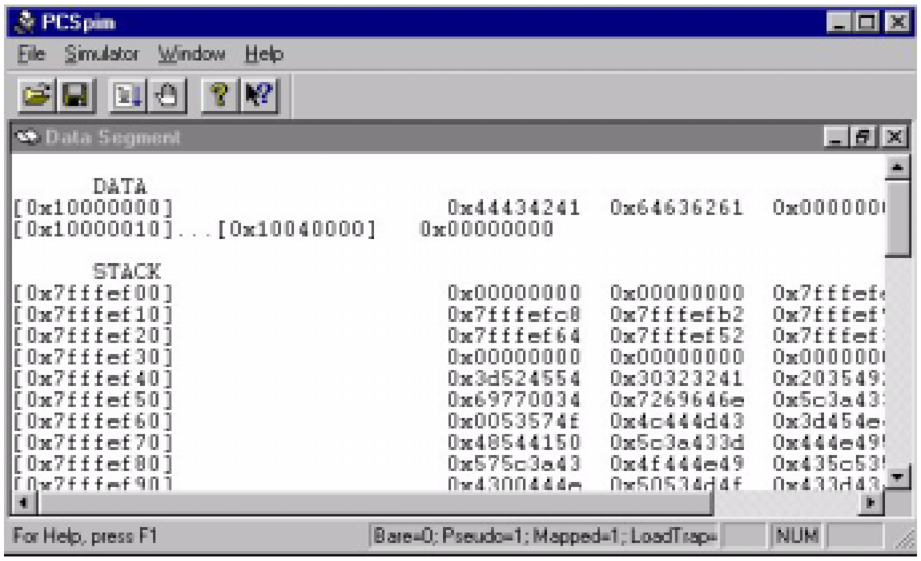
6. In "tutorial.asm", there are two lines of code that we like you to pay attention to right now.

```
.data 0x10000000          (1)
```

.text (2)

These two lines are called assembler directives. (1) specifies the starting address that data will be stored (in our case "ABCDabcd"), and (2) specifies the starting address for the sorce code. Since we specify that data should be stored at 0x10000000, we can expect to see hexadecimal representation of "ABCDabcd" at the address 0x10000000. We did not specify the location of our source code, therefore, it will be stored at system default value (0x00400000).

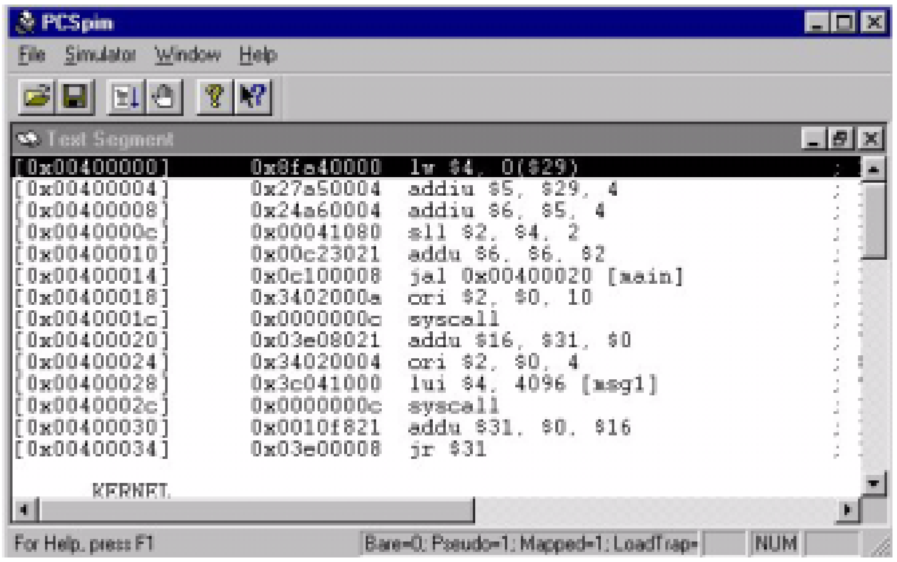
To check on the data, click on Window menu and select Data Segment. This is what you'll see.



```
     DATA
[0x10000000]                        0x44434241  0x64636261  0x000000
[0x10000010]...[0x10040000]    0x00000000

     STACK
[0x7fffef00]                        0x00000000  0x00000000  0x7fffef
[0x7fffef10]                        0x7fffefc8  0x7fffefb2  0x7fffef
[0x7fffef20]                        0x7fffef64  0x7fffef52  0x7fffef
[0x7fffef30]                        0x00000000  0x00000000  0x000000
[0x7fffef40]                        0x3d524554  0x30323241  0x203549
[0x7fffef50]                        0x69770034  0x7269646e  0x5c3a43
[0x7fffef60]                        0x0053574f  0x4c444d43  0x3d454e
[0x7fffef70]                        0x48544150  0x5c3a433d  0x444e49
[0x7fffef80]                        0x575c3a43  0x4f444e49  0x435c53
[0x7fffef90]                        0x4300444e  0x50534d4f  0x433d43
```

Look at the second and third lines carefully. Can you tell what characters are being stored in these addresses and in which order? It should be quite simple to answer.

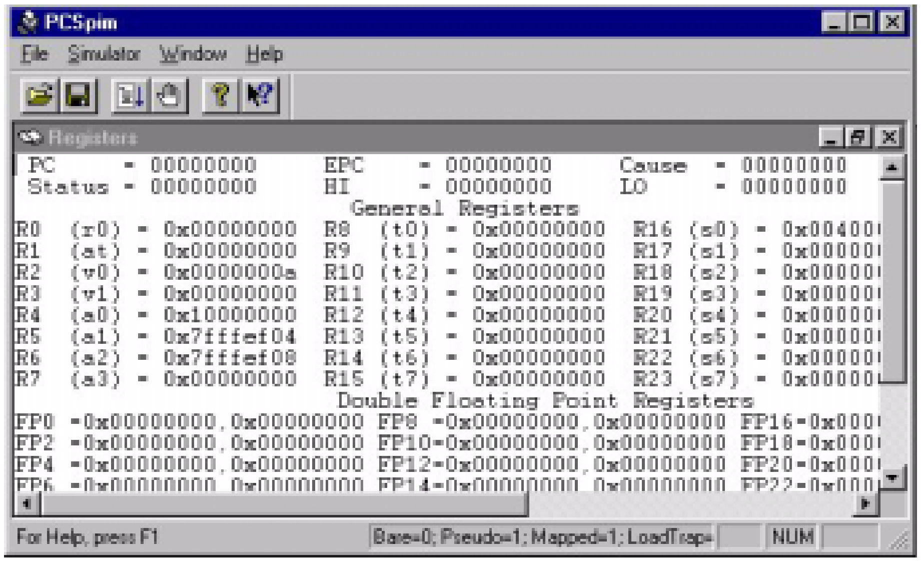
7. If you want to see the source code and the location for its storage, you'd have to select Text Segment from Window menu. If you choose to do so, the following screen would appear.



```
[0x00400000]        0x8fa40000  lw $4, 0($29)                 ;
[0x00400004]        0x27a50004  addiu $5, $29, 4              ;
[0x00400008]        0x24a60004  addiu $6, $5, 4               ;
[0x0040000c]        0x00041080  sll $2, $4, 2                 ;
[0x00400010]        0x00c23021  addu $6, $6, $2               ;
[0x00400014]        0x0c100008  jal 0x00400020 [main]         ;
[0x00400018]        0x3402000a  ori $2, $0, 10                ;
[0x0040001c]        0x0000000c  syscall                       ;
[0x00400020]        0x03e08021  addu $16, $31, $0             ;
[0x00400024]        0x34020004  ori $2, $0, 4                 ;
[0x00400028]        0x3c041000  lui $4, 4096 [msg1]           ;
[0x0040002c]        0x0000000c  syscall                       ;
[0x00400030]        0x0010f821  addu $31, $0, $16             ;
[0x00400034]        0x03e00008  jr $31                        ;

     KERNEL
```

Try to make some senses out of all this data. Let us give you a hint. The left most column is the memory location (address). The second column is the hexadecimal representation of the instructions. The third column is called the native code, and the fourth column is your source code. Note the first line of your source code does not start until address 0x00400020.

8. The other window that you ought to be familiar with is the Register segment. In this window, all the available registers are displayed. Never mind what they mean at this point, you'll have plenty of time to know them. Let us give you a glimps of what it looks like. Select Registers from Window menu and this is what you'll see.


PCSpim
File Simulator Window Help
Registers
For Help, press F1
Base=0; Pseudo=1; Mapped=1; LoadTrap=
NUM

```
 PC     = 00000000     EPC    = 00000000     Cause  = 00000000
 Status = 00000000     HI     = 00000000     LO     = 00000000
                          General Registers
R0  (r0) = 0x00000000  R8  (t0) = 0x00000000  R16 (s0) = 0x00400
R1  (at) = 0x00000000  R9  (t1) = 0x00000000  R17 (s1) = 0x00000
R2  (v0) = 0x0000000a  R10 (t2) = 0x00000000  R18 (s2) = 0x00000
R3  (v1) = 0x00000000  R11 (t3) = 0x00000000  R19 (s3) = 0x00000
R4  (a0) = 0x10000000  R12 (t4) = 0x00000000  R20 (s4) = 0x00000
R5  (a1) = 0x7fffef04  R13 (t5) = 0x00000000  R21 (s5) = 0x00000
R6  (a2) = 0x7fffef08  R14 (t6) = 0x00000000  R22 (s6) = 0x00000
R7  (a3) = 0x00000000  R15 (t7) = 0x00000000  R23 (s7) = 0x00000
                     Double Floating Point Registers
FP0 =0x00000000,0x00000000 FP8 =0x00000000,0x00000000 FP16=0x000
FP2 =0x00000000,0x00000000 FP10=0x00000000,0x00000000 FP18=0x000
FP4 =0x00000000,0x00000000 FP12=0x00000000,0x00000000 FP20=0x000
FP6 =0x00000000,0x00000000 FP14=0x00000000,0x00000000 FP22=0x000
```

At this point, we would like you to explore some other features of this software. Remember, learning is done through doing. Pay attention to all the actions you take and try to understand them.

This tutorial hopefully will give you a starting point in working with PCSPIM. As we work into the semester, you will become more experienced with the software and hopefully gain some insight to the internal working order of a computer system.