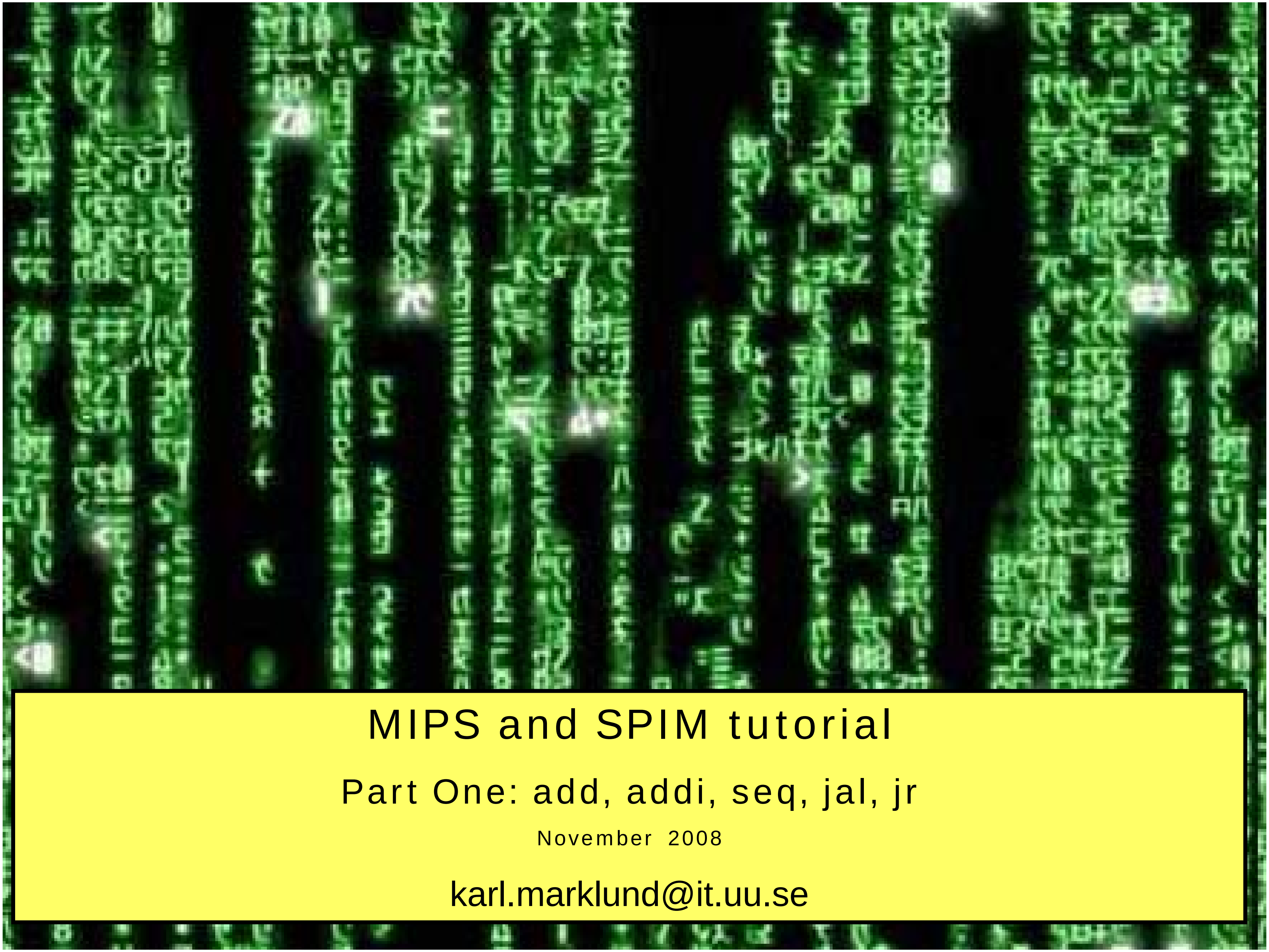


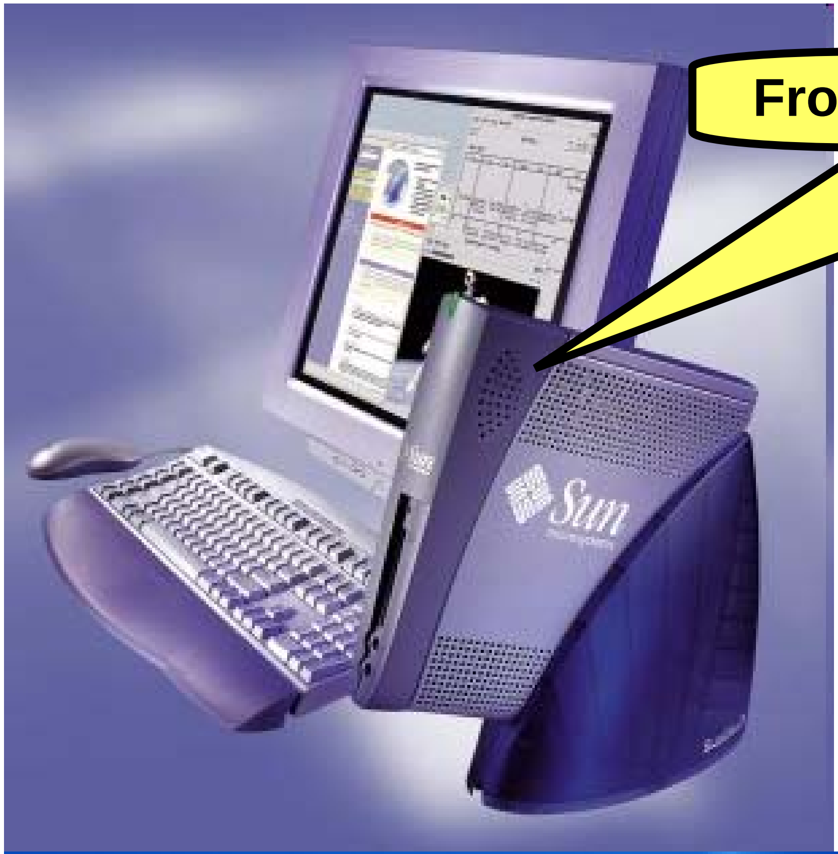
MIPS and SPIM tutorial

Part One: add, addi, seq, jal, jr

November 2008

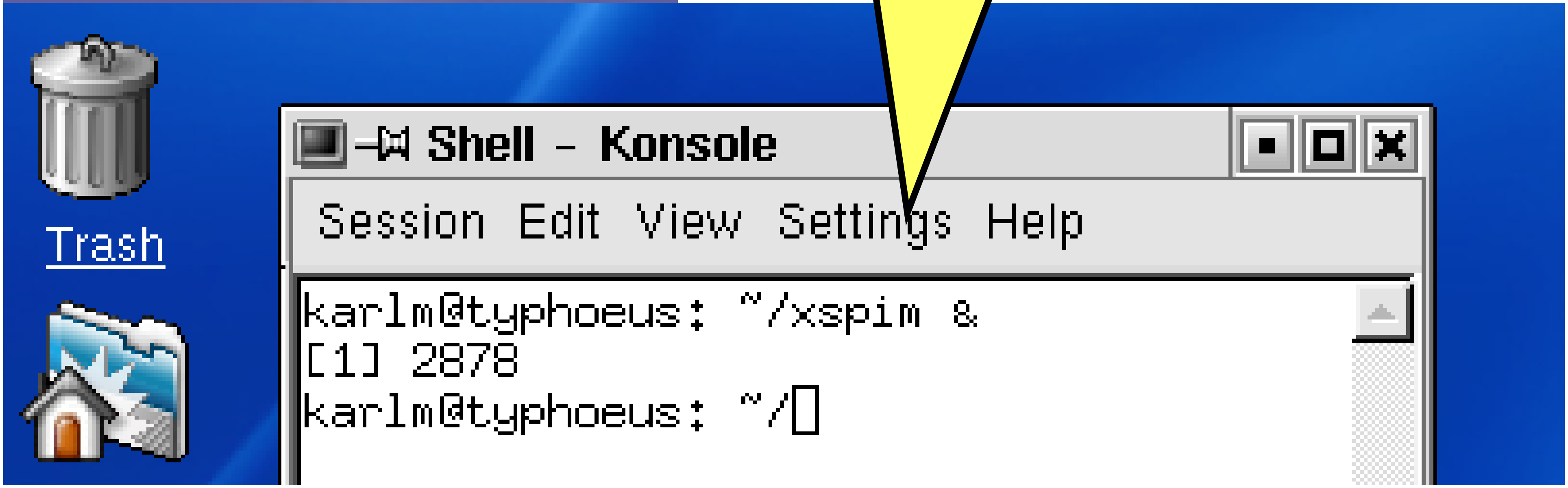
karl.marklund@it.uu.se

Your MIPS assembly programming training starts now – sit down at your computer and get ready!



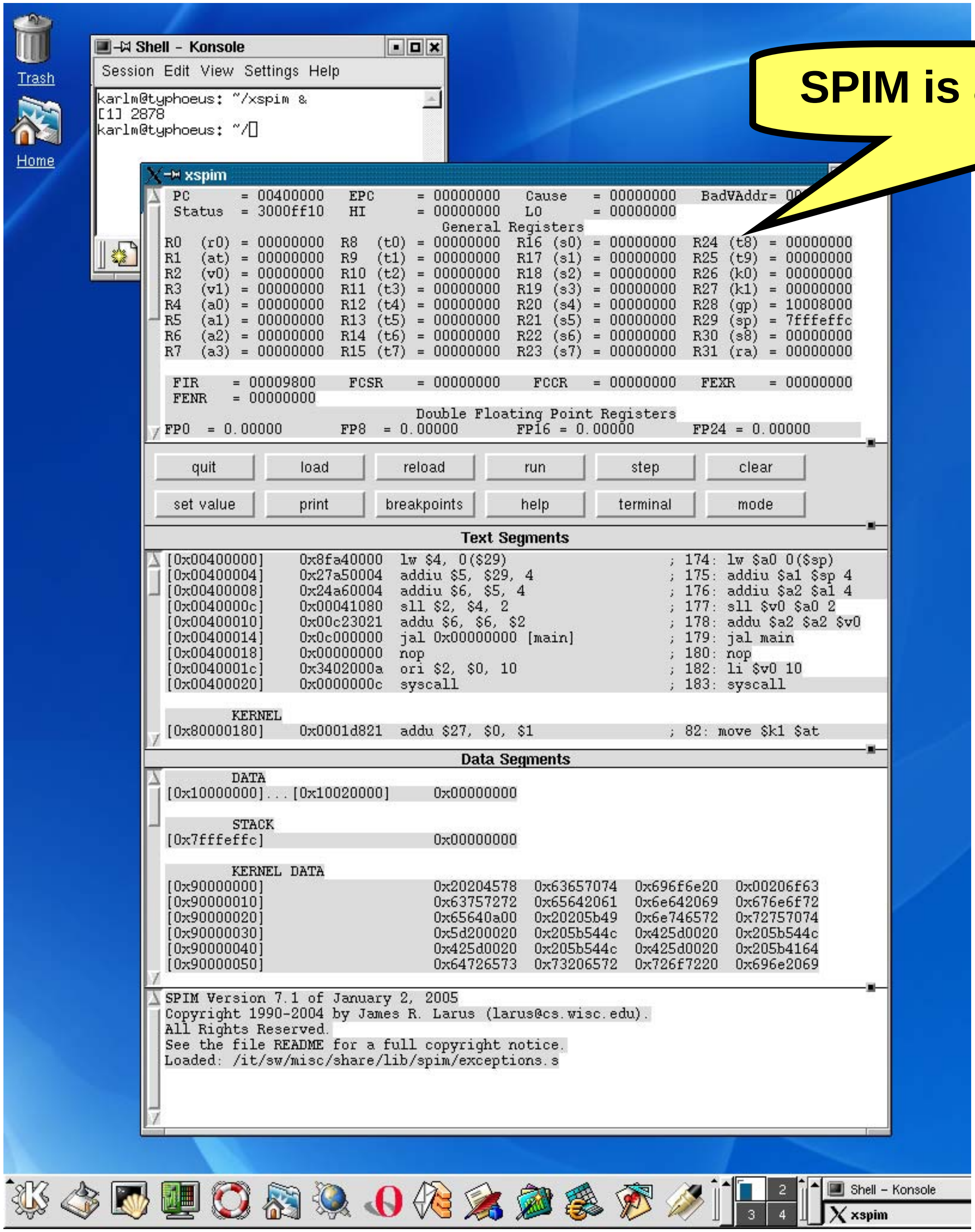
From the department Unix system...

...open up a command shell and type *xspim &* and press enter to launch SPIM.



```
karlm@typhoeus: ~/xspim &
[1] 2878
karlm@typhoeus: ~/
```

SPIM is a simulator for MIPS.



```
karlm@typhoeus: ~/xspim &
[1] 2878
karlm@typhoeus: ~/
```

```
 PC       = 00400000   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00
 Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                              General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 00000000  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffeffc
R6  (a2) = 00000000  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00000000

 FIR    = 00009800   FCSR     = 00000000   FCCR     = 00000000   FEXR     = 00000000
 FENR   = 00000000
                          Double Floating Point Registers
FP0  = 0.00000       FP8  = 0.00000       FP16 = 0.00000       FP24 = 0.00000
```

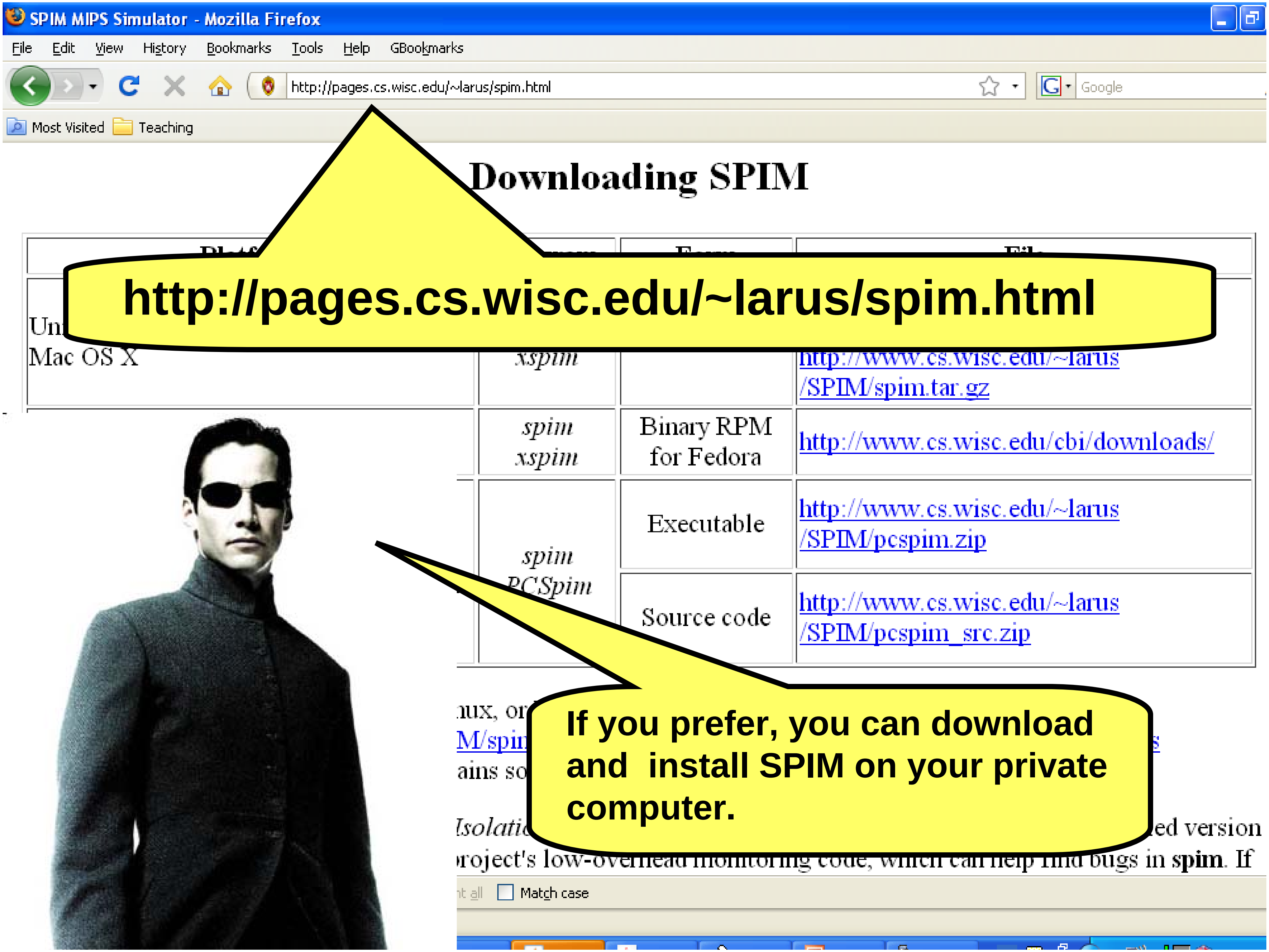
Text Segments

```
[0x00400000]    0x8fa40000  lw $4, 0($29)                 ; 174: lw $a0 0($sp)
[0x00400004]    0x27a50004  addiu $5, $29, 4              ; 175: addiu $a1 $sp 4
[0x00400008]    0x24a60004  addiu $6, $5, 4               ; 176: addiu $a2 $a1 4
[0x0040000c]    0x00041080  sll $2, $4, 2                 ; 177: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2               ; 178: addu $a2 $a2 $v0
[0x00400014]    0x0c000000  jal 0x00000000 [main]         ; 179: jal main
[0x00400018]    0x00000000  nop                           ; 180: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                ; 182: li $v0 10
[0x00400020]    0x0000000c  syscall                       ; 183: syscall

        KERNEL
[0x80000180]    0x0001d821  addu $27, $0, $1              ; 82: move $k1 $at
```

Data Segments

```
        DATA
[0x10000000]...[0x10020000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x20204578  0x63657074  0x696f6e20  0x00206f63
[0x90000010]                    0x63757272  0x65642061  0x6e642069  0x676e6f72
[0x90000020]                    0x65640a00  0x20205b49  0x6e746572  0x72757074
[0x90000030]                    0x5d200020  0x205b544c  0x425d0020  0x205b544c
[0x90000040]                    0x425d0020  0x205b544c  0x425d0020  0x205b4164
[0x90000050]                    0x64726573  0x73206572  0x726f7220  0x696e2069
```

```
SPIM Version 7.1 of January 2, 2005
Copyright 1990-2004 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /it/sw/misc/share/lib/spim/exceptions.s
```



SPIM MIPS Simulator - Mozilla Firefox

File Edit View History Bookmarks Tools Help GBookmarks

http://pages.cs.wisc.edu/~larus/spim.html

Most Visited Teaching

Downloading SPIM

Platf	ram	Form	File
Un Mac OS X	xspim		http://www.cs.wisc.edu/~larus/SPIM/spim.tar.gz
	spim xspim	Binary RPM for Fedora	http://www.cs.wisc.edu/cbi/downloads/
	spim PCSpim	Executable	http://www.cs.wisc.edu/~larus/SPIM/pcspim.zip
		Source code	http://www.cs.wisc.edu/~larus/SPIM/pcspim_src.zip

nux, or

M/spi

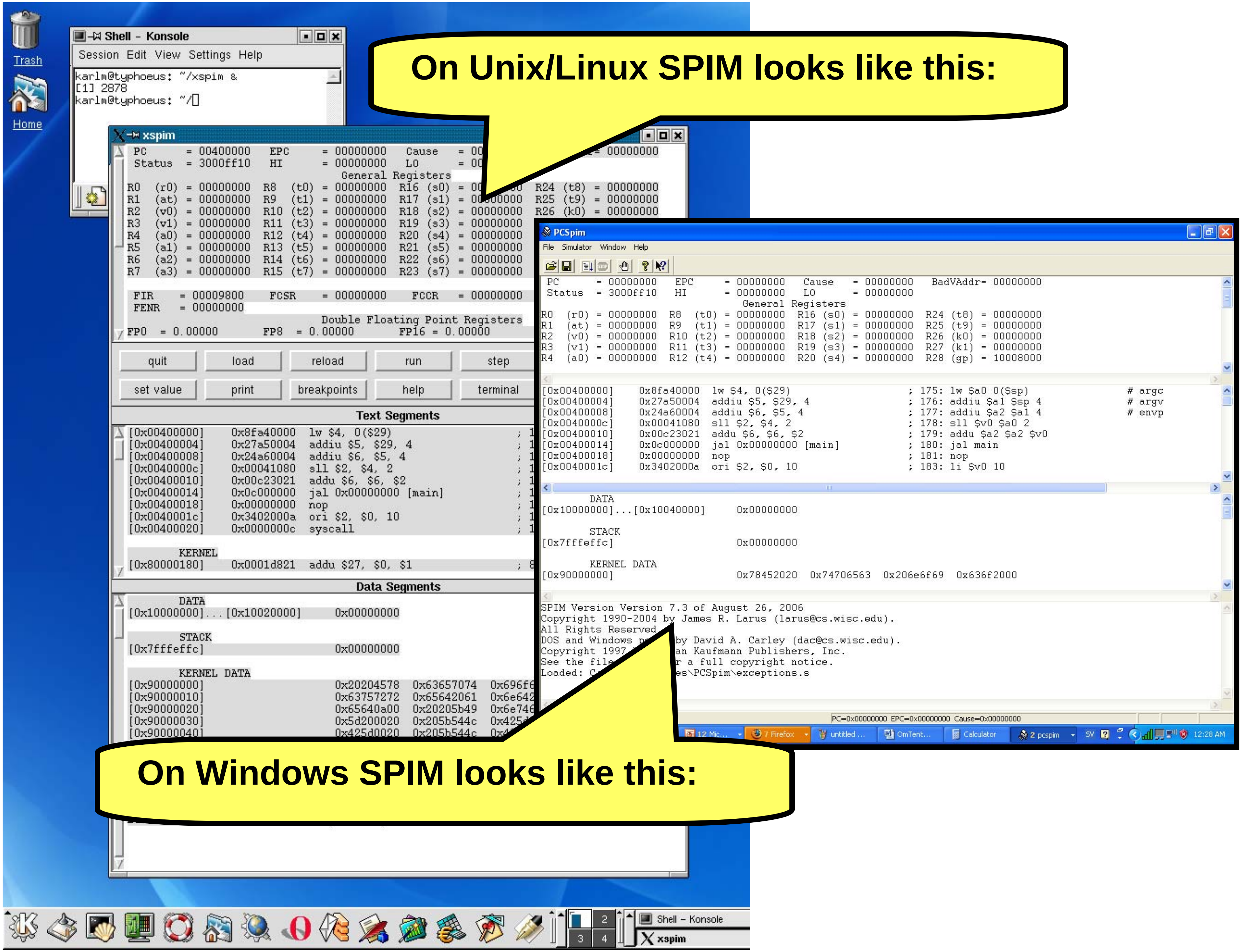
ains so

Isolati ed version

roject's low-overhead monitoring code, which can help find bugs in **spim**. If

http://pages.cs.wisc.edu/~larus/spim.html

If you prefer, you can download and install SPIM on your private computer.



On Unix/Linux SPIM looks like this:

```
karlm@typhoeus: ~/xspim &
[1] 2878
karlm@typhoeus: ~/
```

```
PC     = 00400000   EPC    = 00000000   Cause  = 00                  = 00000000
Status = 3000ff10   HI     = 00000000   LO     = 00
                          General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00        00  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00        00  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000
R5  (a1) = 00000000  R13 (t5) = 00000000  R21 (s5) = 00000000
R6  (a2) = 00000000  R14 (t6) = 00000000  R22 (s6) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000

 FIR    = 00009800   FCSR   = 00000000   FCCR   = 00000000
 FENR   = 00000000
                          Double Floating Point Registers
FP0   = 0.00000      FP8   = 0.00000     FP16 = 0.00000
```

```
                              Text Segments
[0x00400000]    0x8fa40000  lw $4, 0($29)                     ; 1
[0x00400004]    0x27a50004  addiu $5, $29, 4                  ; 1
[0x00400008]    0x24a60004  addiu $6, $5, 4                   ; 1
[0x0040000c]    0x00041080  sll $2, $4, 2                     ; 1
[0x00400010]    0x00c23021  addu $6, $6, $2                   ; 1
[0x00400014]    0x0c000000  jal 0x00000000 [main]             ; 1
[0x00400018]    0x00000000  nop                               ; 1
[0x0040001c]    0x3402000a  ori $2, $0, 10                    ; 1
[0x00400020]    0x0000000c  syscall                           ; 1

        KERNEL
[0x80000180]    0x0001d821  addu $27, $0, $1                  ; 8
```

```
                              Data Segments
        DATA
[0x10000000]...[0x10020000]      0x00000000

        STACK
[0x7fffeffc]                     0x00000000

        KERNEL DATA
[0x90000000]                     0x20204578  0x63657074  0x696f6
[0x90000010]                     0x63757272  0x65642061  0x6e642
[0x90000020]                     0x65640a00  0x20205b49  0x6e746
[0x90000030]                     0x5d200020  0x205b544c  0x425
[0x90000040]                     0x425d0020  0x205b544c  0x4
```

On Windows SPIM looks like this:

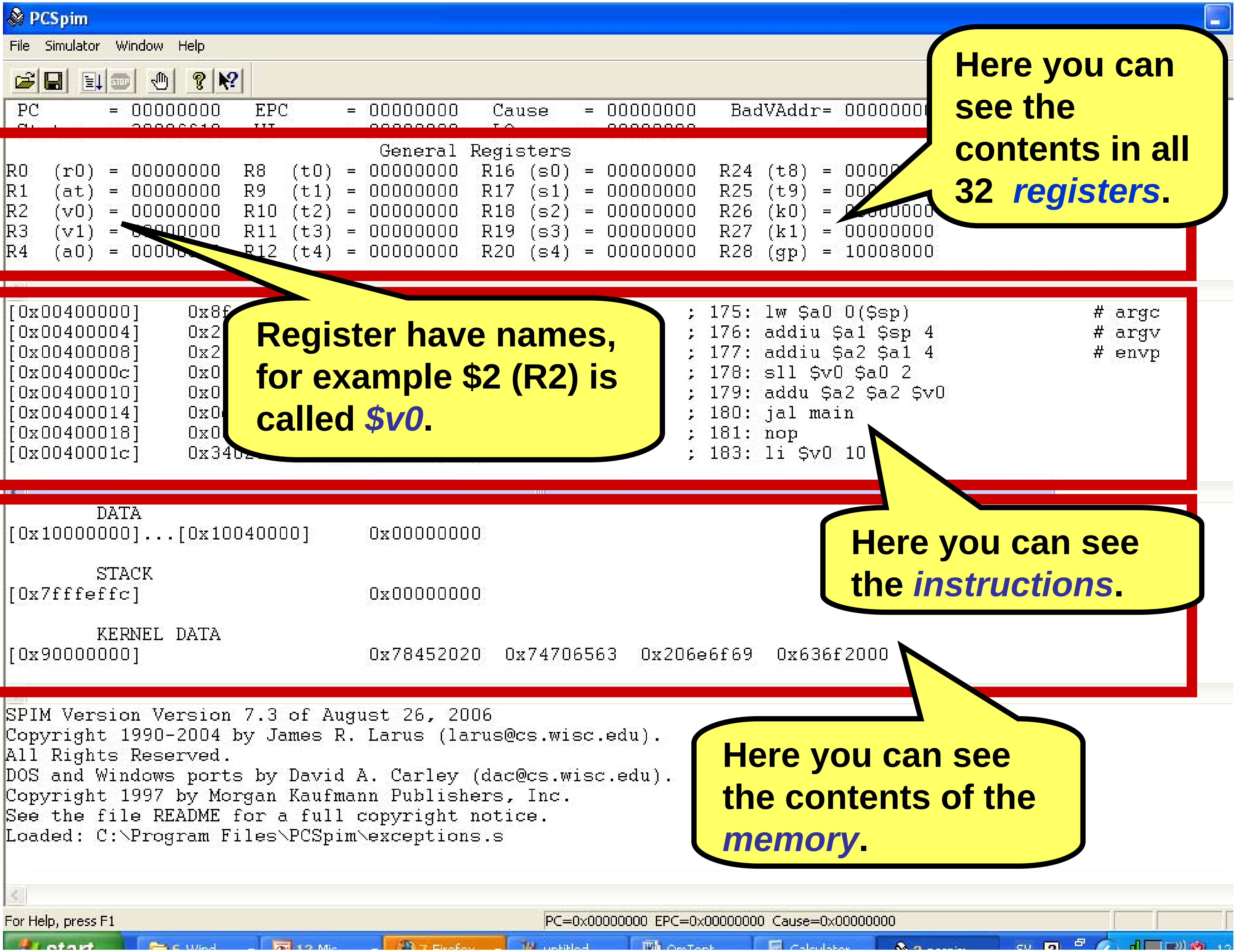
```
 PC      = 00000000   EPC     = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status  = 3000ff10   HI      = 00000000   LO       = 00000000
                             General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400000]    0x8fa40000  lw $4, 0($29)                ; 175: lw $a0 0($sp)          # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4             ; 176: addiu $a1 $sp 4        # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4              ; 177: addiu $a2 $a1 4        # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c000000  jal 0x00000000 [main]        ; 180: jal main
[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10               ; 183: li $v0 10
```

```
        DATA
[0x10000000]...[0x10040000]      0x00000000

        STACK
[0x7fffeffc]                     0x00000000

        KERNEL DATA
[0x90000000]                     0x78452020  0x74706563  0x206e6f69  0x636f2000
```

```
SPIM Version Version 7.3 of August 26, 2006
Copyright 1990-2004 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved
DOS and Windows p        by David A. Carley (dac@cs.wisc.edu).
Copyright 1997           an Kaufmann Publishers, Inc.
See the file             r a full copyright notice.
Loaded: C                es\PCSpim\exceptions.s
```

```
PC=0x00000000  EPC=0x00000000  Cause=0x00000000
```



```
PC       = 00000000   EPC      = 00000000   Cause    = 00000000   BadVAddr= 0000000
```

```
                              General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 0000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) =
R3  (v1) =           R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000     R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400000]    0x8f     ; 175: lw $a0 0($sp)           # argc
[0x00400004]    0x2      ; 176: addiu $a1 $sp 4         # argv
[0x00400008]    0x2      ; 177: addiu $a2 $a1 4         # envp
[0x0040000c]    0x0      ; 178: sll $v0 $a0 2
[0x00400010]    0x0      ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0      ; 180: jal main
[0x00400018]    0x0      ; 181: nop
[0x0040001c]    0x340    ; 183: li $v0 10
```

```
        DATA
[0x10000000]...[0x10040000]      0x00000000

        STACK
[0x7fffeffc]                     0x00000000

        KERNEL DATA
[0x90000000]                     0x78452020  0x74706563  0x206e6f69  0x636f2000
```

```
SPIM Version Version 7.3 of August 26, 2006
Copyright 1990-2004 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:\Program Files\PCSpim\exceptions.s
```

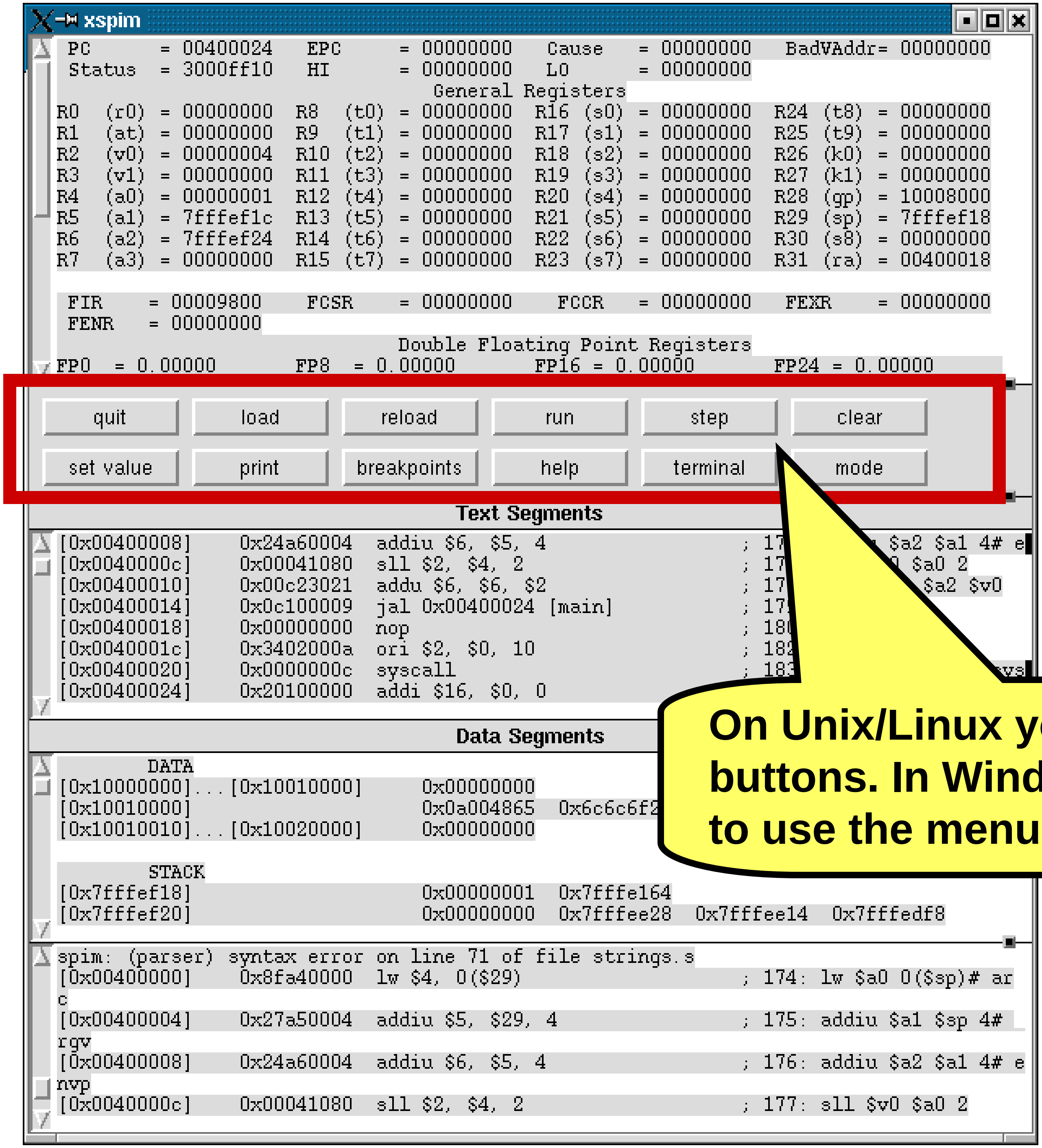
```
PC=0x00000000  EPC=0x00000000  Cause=0x00000000
```

Here you can see the contents in all 32 *registers*.

Register have names, for example $2 (R2) is called *$v0*.

Here you can see the *instructions*.

Here you can see the contents of the *memory*.



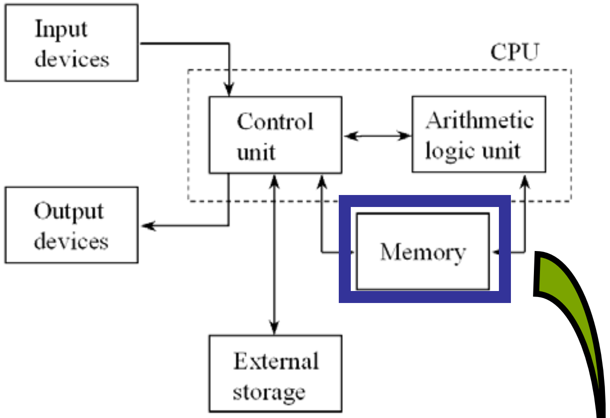
```
 PC       = 00400024   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                              General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000004  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000001  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 7fffef1c  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffef18
R6  (a2) = 7fffef24  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400018

 FIR     = 00009800    FCSR     = 00000000    FCCR     = 00000000   FEXR     = 00000000
 FENR    = 00000000
                           Double Floating Point Registers
FP0   = 0.00000        FP8   = 0.00000        FP16 = 0.00000        FP24 = 0.00000
```

```
[0x00400008]    0x24a60004  addiu $6, $5, 4                     ; 17       $a2 $a1 4# e
[0x0040000c]    0x00041080  sll $2, $4, 2                       ; 17       0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2                     ; 17        $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]               ; 17
[0x00400018]    0x00000000  nop                                 ; 180
[0x0040001c]    0x3402000a  ori $2, $0, 10                      ; 182
[0x00400020]    0x0000000c  syscall                             ; 183              sys
[0x00400024]    0x20100000  addi $16, $0, 0
```

```
        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x0a004865  0x6c6c6f2
[0x10010010]...[0x10020000]     0x00000000

        STACK
[0x7fffef18]                    0x00000001  0x7fffe164
[0x7fffef20]                    0x00000000  0x7fffee28  0x7fffee14  0x7fffedf8
```

```
spim: (parser) syntax error on line 71 of file strings.s
[0x00400000]    0x8fa40000  lw $4, 0($29)                       ; 174: lw $a0 0($sp)# ar
c
[0x00400004]    0x27a50004  addiu $5, $29, 4                    ; 175: addiu $a1 $sp 4#
rgv
[0x00400008]    0x24a60004  addiu $6, $5, 4                     ; 176: addiu $a2 $a1 4# e
nvp
[0x0040000c]    0x00041080  sll $2, $4, 2                       ; 177: sll $v0 $a0 2
```

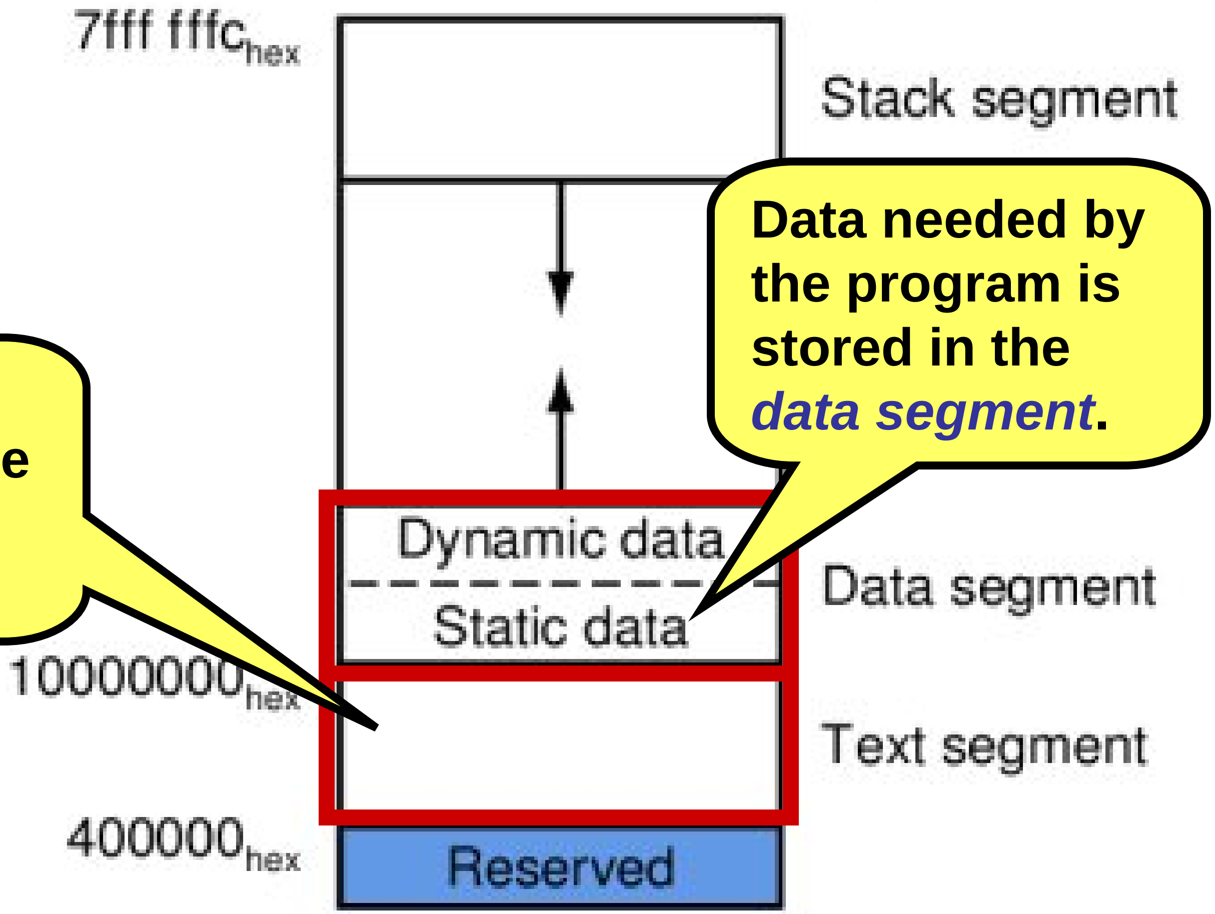
On Unix/Linux you have these buttons. In Windows you will have to use the menus.



MEMORY

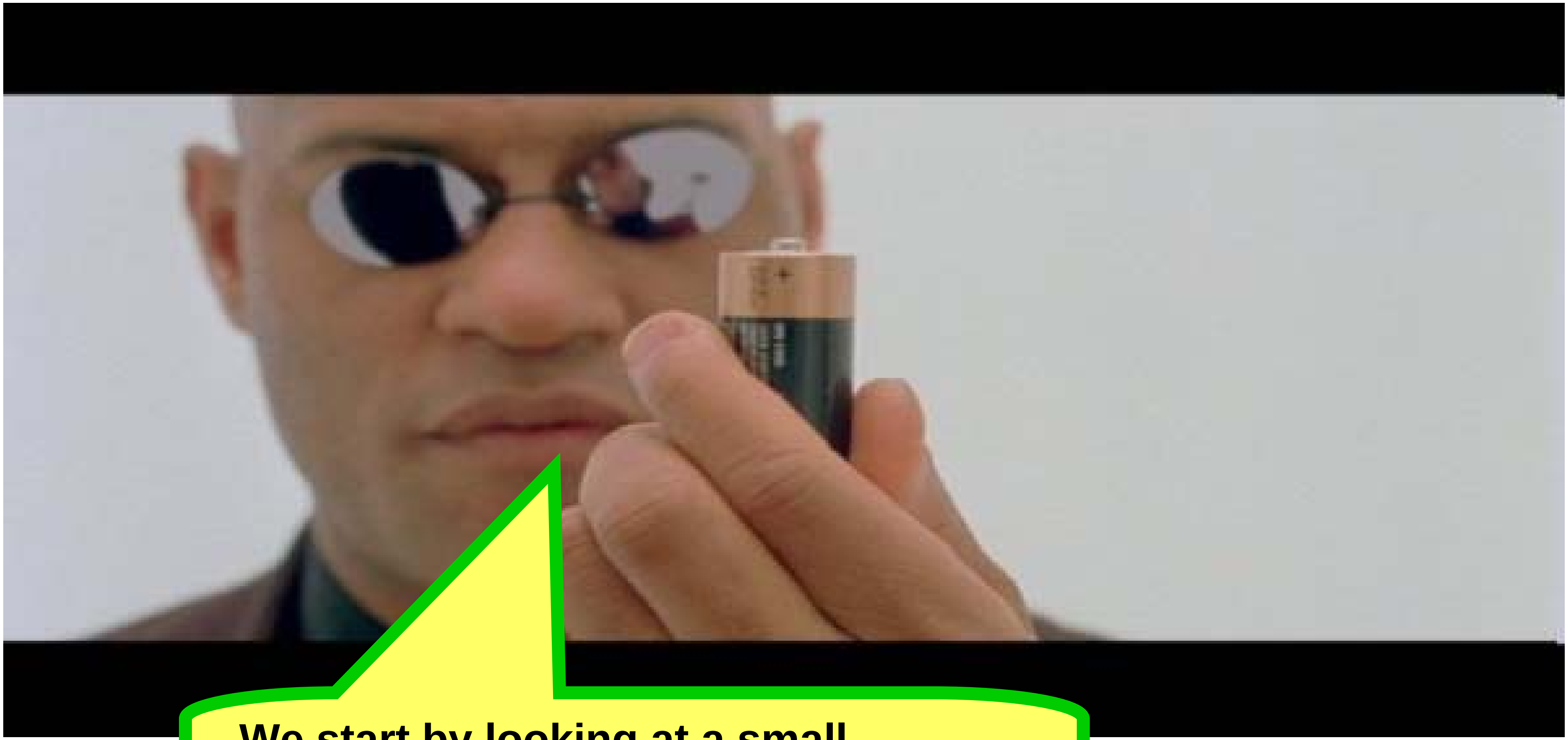
Address	Content
0xFFFFFFFF	
0xFFFFFFFE	
0xFFFFF	
0xFFFFF	
⋮	
0x00000003	
0x00000002	
0x00000001	
0x00000000	

The memory is divided into *segments*.



Data needed by the program is stored in the *data segment*.

The *machine instructions* are stored in the text segment.



We start by looking at a small example program in MIPS assembly.

first_try.s

A *label* is used to refer to places in the program.

A label is just a named address in memory (text segment).

The assembler *directive* .text instructs the assembler to treat what follows as program instructions. The assembler will translate each line (instruction) to the binary machine instruction and store the result in the *text segmentet* in memory.

The label main must be decleared global using the *.globl* directive.

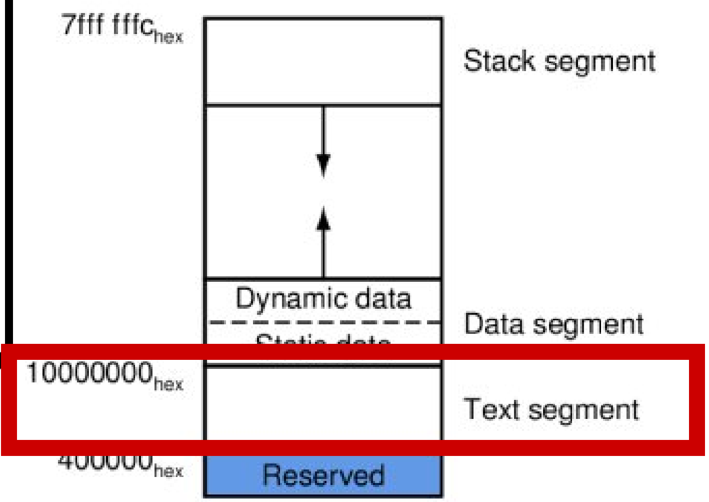
When SPIM starts, it will load your program and start executing at the label main.

Your program must allways end with the instruction:

jr $ra

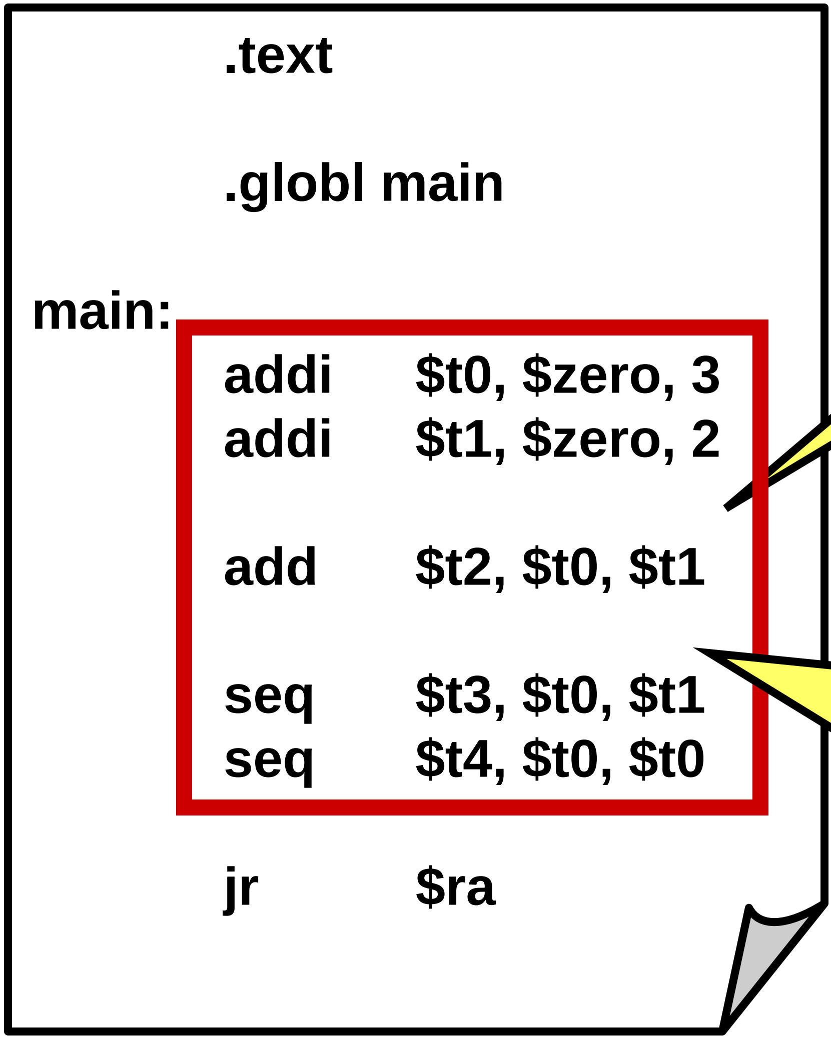
Jump Register (ra).

```
.text

.globl main

main:


jr        $ra
```



first_try.s

The file name of a MIPS assembly program got the suffix *.s*

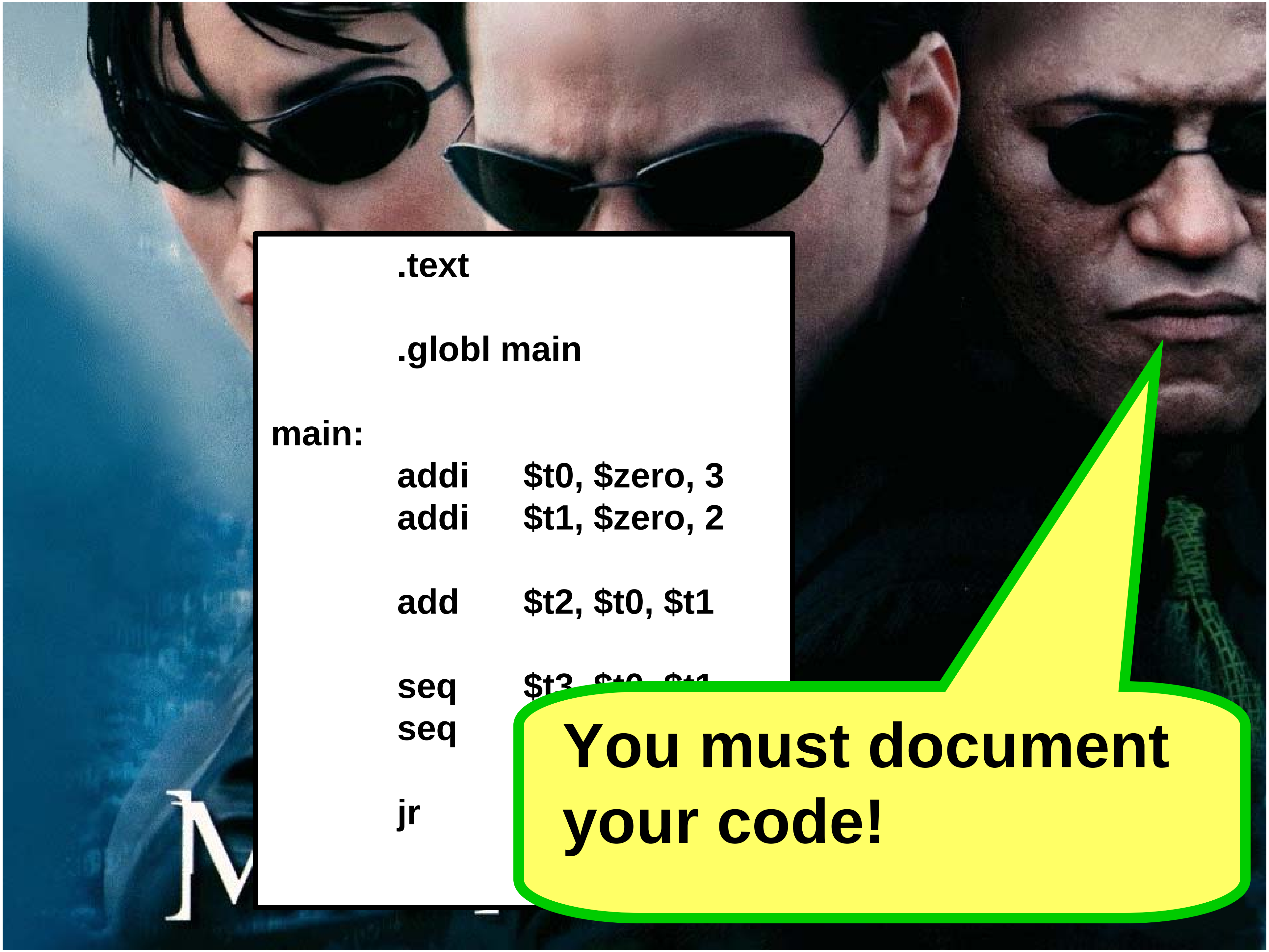


```
        .text

        .globl main

main:
        addi    $t0, $zero, 3
        addi    $t1, $zero, 2

        add     $t2, $t0, $t1

        seq     $t3, $t0, $t1
        seq     $t4, $t0, $t0

        jr      $ra
```

Lets add some instructions here...

...this is stuff we know:

addi (add immediate)
add (addition)
seq (set equal)



```
        .text

        .globl main

main:
        addi    $t0, $zero, 3
        addi    $t1, $zero, 2

        add     $t2, $t0, $t1

        seq     $t3, $t0, $t1
        seq

        jr
```

You must document your code!

first_try.s

A comment starts with #

In the comments we use normal language – *we don’t use register names!!!*

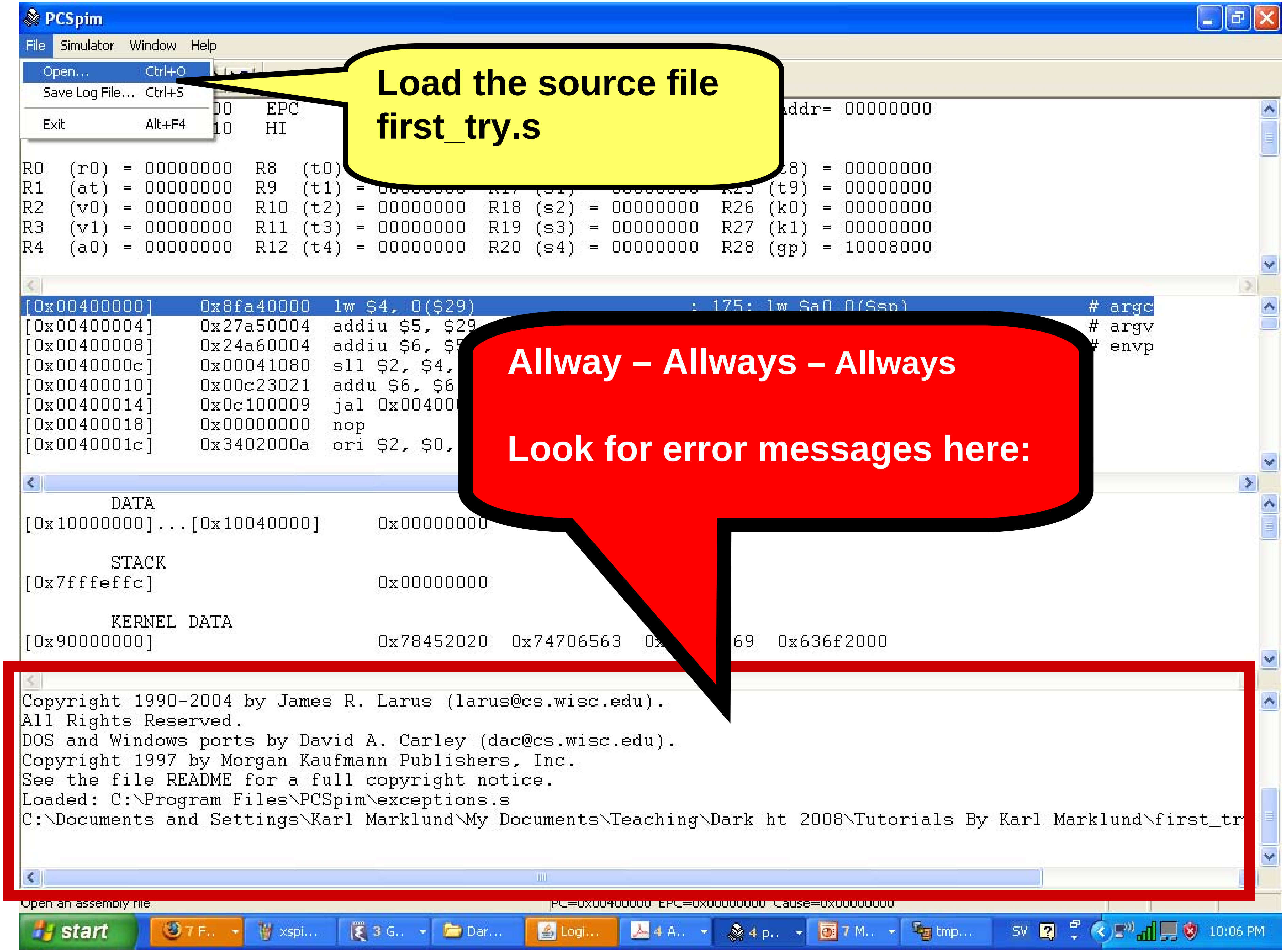
```
        .text

        .globl main

main:
        addi    $t0, $zero, 3     # a = 3
        addi    $t1, $zero, 2     # b = 2

        add     $t2, $t0, $t1     # c = a + b

        seq     $t3, $t0, $t1     # d = 1 iff a == b else d = 0
        seq     $t4, $t0, $t0     # e = 1 iff a == a (sic)

        jr      $ra               # return to caller
```

first_try.s

The comments should describe what your program does.

You should be able to understand the program without the code – only looking at the comments.

```
        .text

        .globl main

main:
                        # a = 3
                        # b = 2

                        # c = a + b

                        # d = 1 iff a == b else d = 0
                        # e = 1 iff a == a (sic)

                        # return to caller
```



Load the source file first_try.s

```
00    EPC                                          ddr= 00000000
10    HI

R0  (r0) = 00000000  R8  (t0)                           8) = 00000000
R1  (at) = 00000000  R9  (t1) =                   R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400000]    0x8fa40000  lw $4, 0($29)          ; 175: lw $a0 0($sp)       # argc
[0x00400004]    0x27a50004  addiu $5, $29                                     # argv
[0x00400008]    0x24a60004  addiu $6, $5                                      # envp
[0x0040000c]    0x00041080  sll $2, $4,
[0x00400010]    0x00c23021  addu $6, $6
[0x00400014]    0x0c100009  jal 0x00400
[0x00400018]    0x00000000  nop
[0x0040001c]    0x3402000a  ori $2, $0,
```

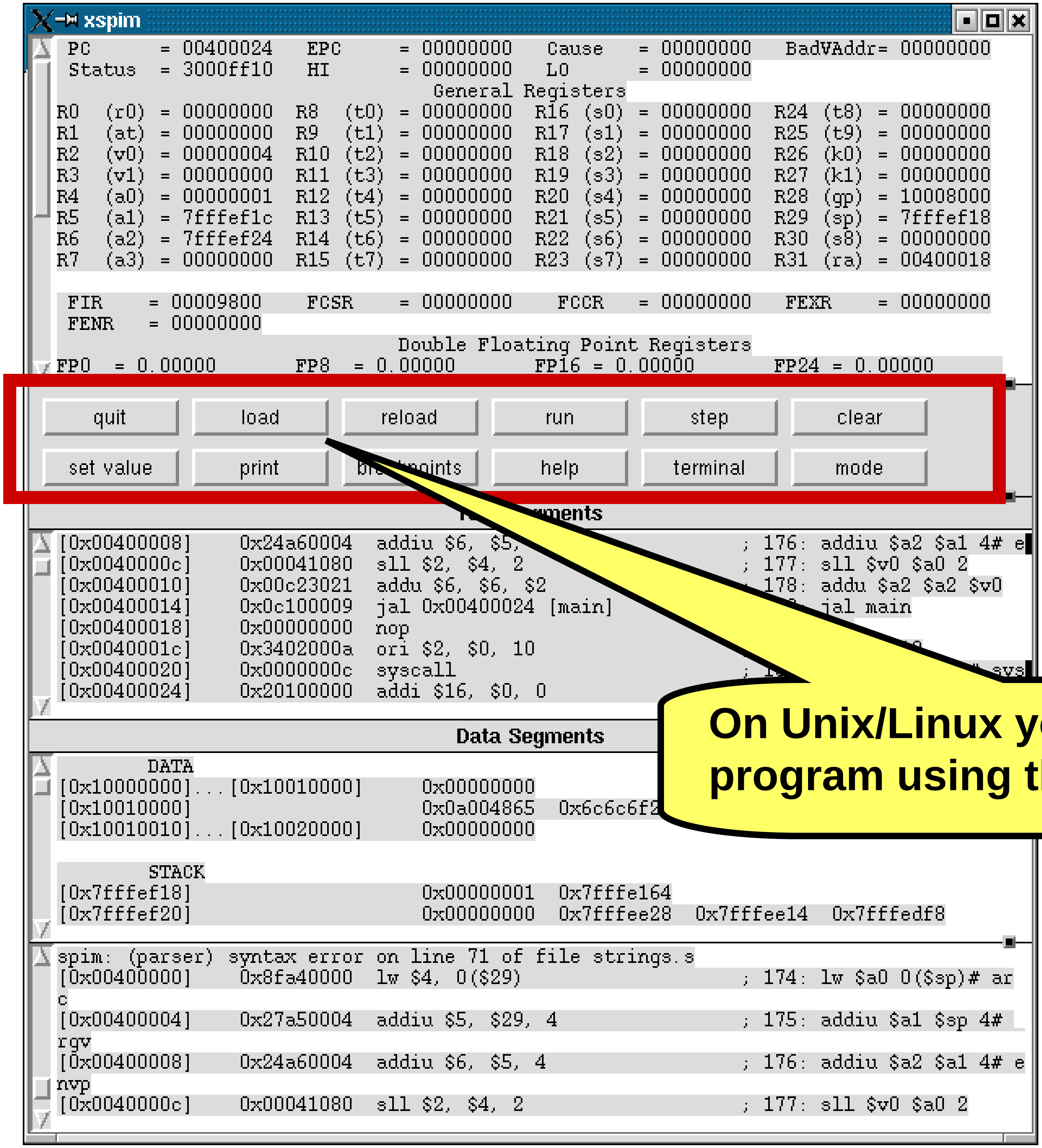
```
        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020  0x74706563  0x        69  0x636f2000
```

Allway – Allways – Allways

Look for error messages here:

```
Copyright 1990-2004 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:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_tr
```

```
Open an assembly file        PC=0x00400000  EPC=0x00000000  Cause=0x00000000
```



```
 PC       = 00400024   EPC      = 00000000    Cause    = 00000000    BadVAddr= 00000000
 Status   = 3000ff10   HI       = 00000000    LO       = 00000000
                                General Registers
R0  (r0)  = 00000000  R8  (t0)  = 00000000  R16 (s0)  = 00000000  R24 (t8)  = 00000000
R1  (at)  = 00000000  R9  (t1)  = 00000000  R17 (s1)  = 00000000  R25 (t9)  = 00000000
R2  (v0)  = 00000004  R10 (t2)  = 00000000  R18 (s2)  = 00000000  R26 (k0)  = 00000000
R3  (v1)  = 00000000  R11 (t3)  = 00000000  R19 (s3)  = 00000000  R27 (k1)  = 00000000
R4  (a0)  = 00000001  R12 (t4)  = 00000000  R20 (s4)  = 00000000  R28 (gp)  = 10008000
R5  (a1)  = 7fffef1c  R13 (t5)  = 00000000  R21 (s5)  = 00000000  R29 (sp)  = 7fffef18
R6  (a2)  = 7fffef24  R14 (t6)  = 00000000  R22 (s6)  = 00000000  R30 (s8)  = 00000000
R7  (a3)  = 00000000  R15 (t7)  = 00000000  R23 (s7)  = 00000000  R31 (ra)  = 00400018

 FIR     = 00009800    FCSR     = 00000000     FCCR     = 00000000    FEXR     = 00000000
 FENR    = 00000000
                             Double Floating Point Registers
FP0   = 0.00000        FP8   = 0.00000        FP16  = 0.00000        FP24  = 0.00000
```

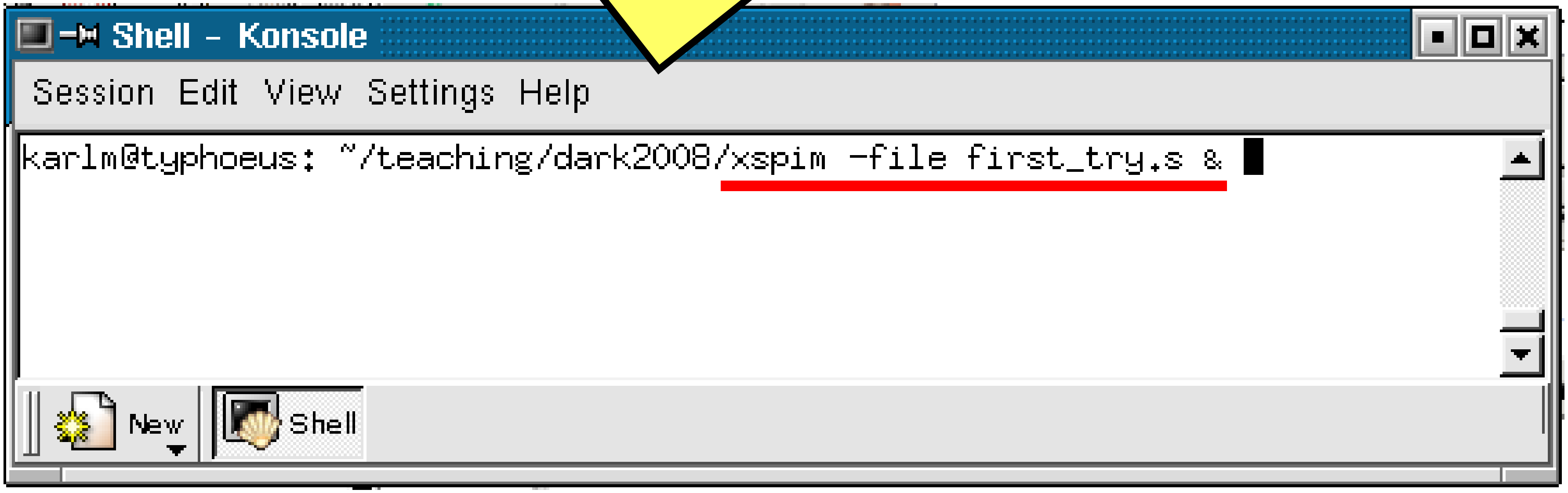
```
[0x00400008]    0x24a60004  addiu $6, $5,                              ; 176: addiu $a2 $a1 4# e
[0x0040000c]    0x00041080  sll $2, $4, 2                              ; 177: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2                              178: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]                            jal main
[0x00400018]    0x00000000  nop
[0x0040001c]    0x3402000a  ori $2, $0, 10
[0x00400020]    0x0000000c  syscall                                    ; 1                 sys
[0x00400024]    0x20100000  addi $16, $0, 0
```

```
        DATA
[0x10000000]...[0x10010000]      0x00000000
[0x10010000]                     0x0a004865  0x6c6c6f2
[0x10010010]...[0x10020000]      0x00000000

        STACK
[0x7fffef18]                     0x00000001  0x7fffe164
[0x7fffef20]                     0x00000000  0x7fffee28  0x7fffee14  0x7fffedf8
```

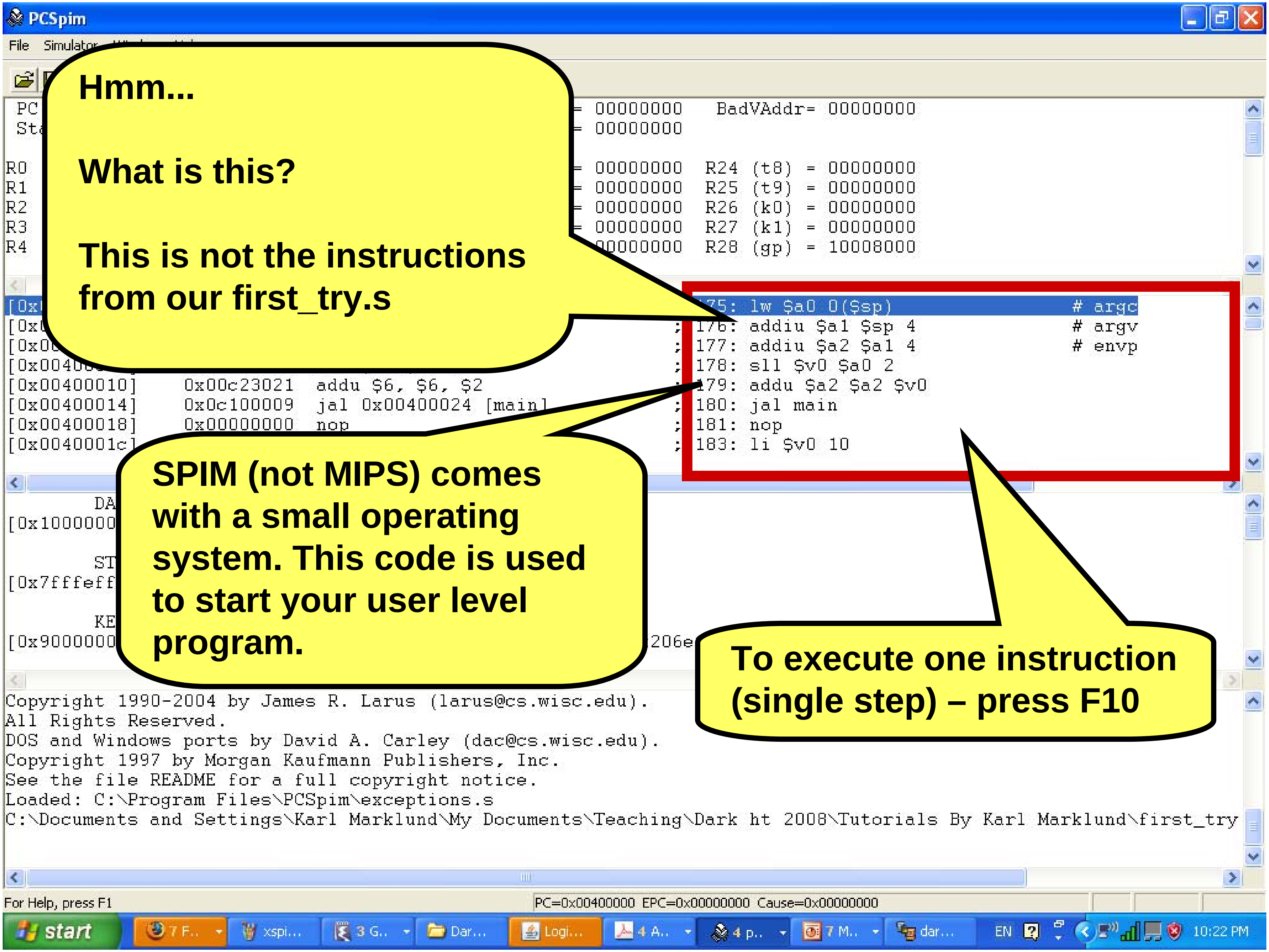
```
spim: (parser) syntax error on line 71 of file strings.s
[0x00400000]    0x8fa40000  lw $4, 0($29)                              ; 174: lw $a0 0($sp)# ar
c
[0x00400004]    0x27a50004  addiu $5, $29, 4                           ; 175: addiu $a1 $sp 4#
rgv
[0x00400008]    0x24a60004  addiu $6, $5, 4                            ; 176: addiu $a2 $a1 4# e
nvp
[0x0040000c]    0x00041080  sll $2, $4, 2                              ; 177: sll $v0 $a0 2
```

On Unix/Linux you can load the program using this button.

On Unix/Linux you can also load the program from the command line using the –file option



```
karlm@typhoeus: ~/teaching/dark2008/xspim -file first_try.s &
```



Hmm...

What is this?

This is not the instructions from our first_try.s

```
PC                                      = 00000000   BadVAddr= 00000000
Sta                                     = 00000000

R0                                      = 00000000  R24 (t8) = 00000000
R1                                      = 00000000  R25 (t9) = 00000000
R2                                      = 00000000  R26 (k0) = 00000000
R3                                      = 00000000  R27 (k1) = 00000000
R4                                      00000000  R28 (gp) = 10008000
```

```
[0x                                                 ; 175: lw $a0 0($sp)           # argc
[0x                                                 ; 176: addiu $a1 $sp 4         # argv
[0x0                                                ; 177: addiu $a2 $a1 4         # envp
[0x0040                                             ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2         ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]   ; 180: jal main
[0x00400018]    0x00000000  nop                     ; 181: nop
[0x0040001c]                                        ; 183: li $v0 10
```

SPIM (not MIPS) comes with a small operating system. This code is used to start your user level program.

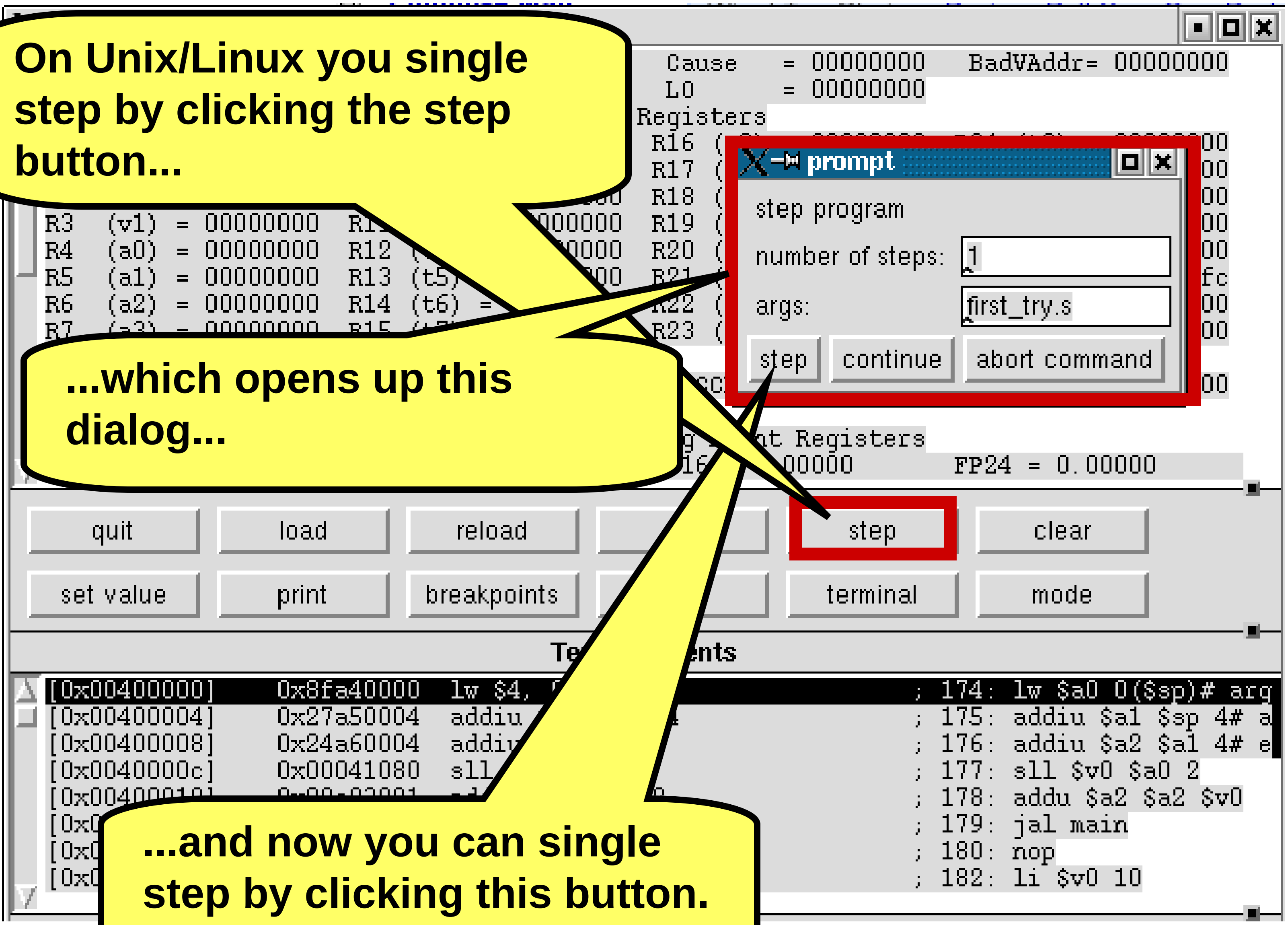
To execute one instruction (single step) – press F10

```
        DA
[0x1000000

        ST
[0x7fffeff

        KE
[0x9000000                                  206e
```

```
Copyright 1990-2004 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:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_try
```

PC=0x00400000 EPC=0x00000000 Cause=0x00000000

On Unix/Linux you single step by clicking the step button...



```
Cause     = 00000000    BadVAddr= 00000000
LO        = 00000000
Registers
R3  (v1) = 00000000
R4  (a0) = 00000000  R12
R5  (a1) = 00000000  R13 (t5)
R6  (a2) = 00000000  R14 (t6) =
FP24 = 0.00000
```

prompt

step program

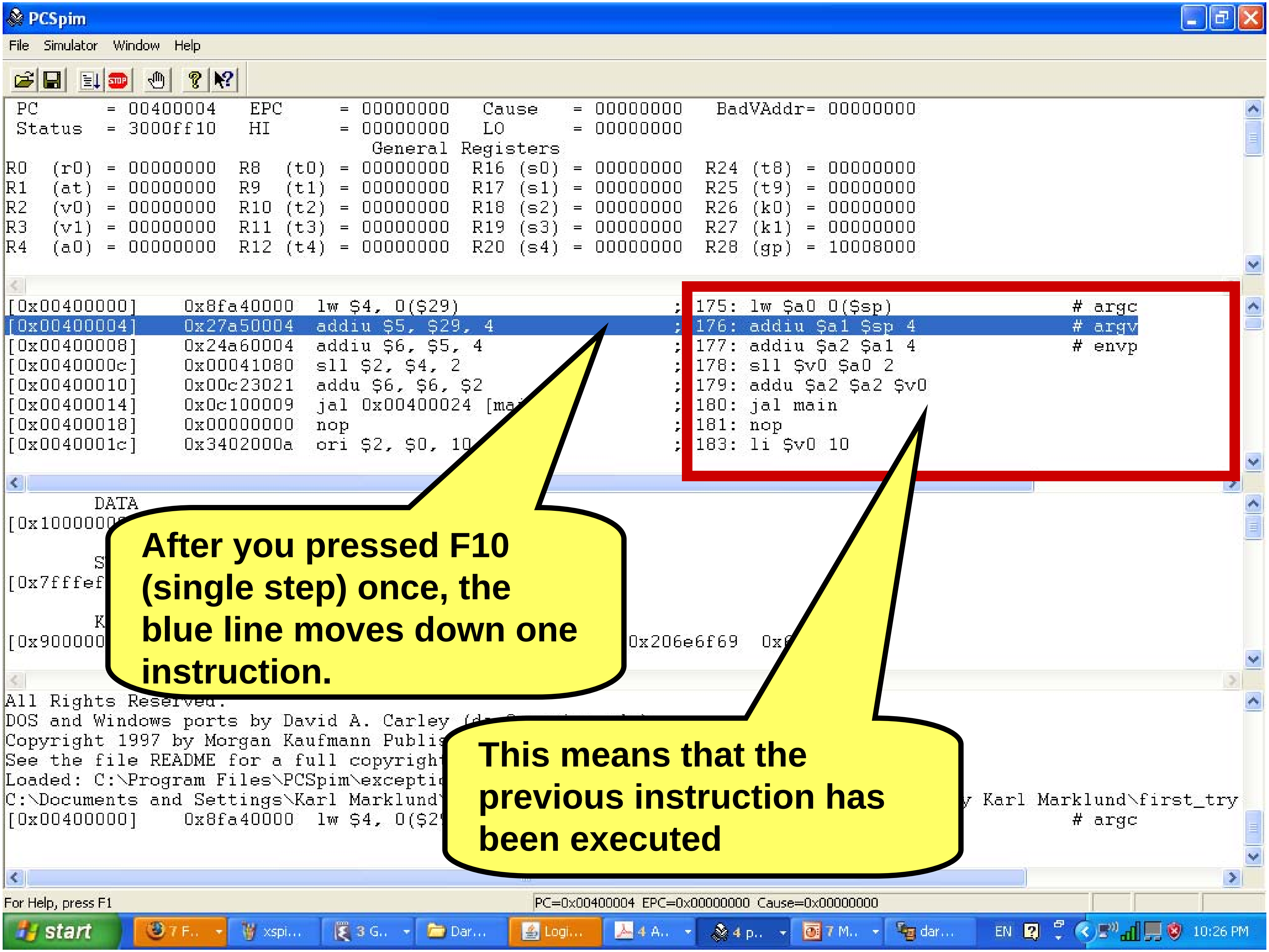
number of steps: 1

args: first_try.s

```
[0x00400000]    0x8fa40000  lw $4,          ; 174: lw $a0 0($sp)# arg
[0x00400004]    0x27a50004  addiu           ; 175: addiu $a1 $sp 4# a
[0x00400008]    0x24a60004  addiu           ; 176: addiu $a2 $a1 4# e
[0x0040000c]    0x00041080  sll             ; 177: sll $v0 $a0 2
                                            ; 178: addu $a2 $a2 $v0
                                            ; 179: jal main
                                            ; 180: nop
                                            ; 182: li $v0 10
```

...which opens up this dialog...

...and now you can single step by clicking this button.



```
 PC      = 00400004   EPC     = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status  = 3000ff10   HI      = 00000000   LO       = 00000000
                              General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400000]    0x8fa40000  lw $4, 0($29)                ; 175: lw $a0 0($sp)            # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4             ; 176: addiu $a1 $sp 4          # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4              ; 177: addiu $a2 $a1 4          # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [ma           ; 180: jal main
[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10               ; 183: li $v0 10
```

```
        DATA
[0x1000000

        S
[0x7fffef

        K
[0x900000                          0x206e6f69   0x
```

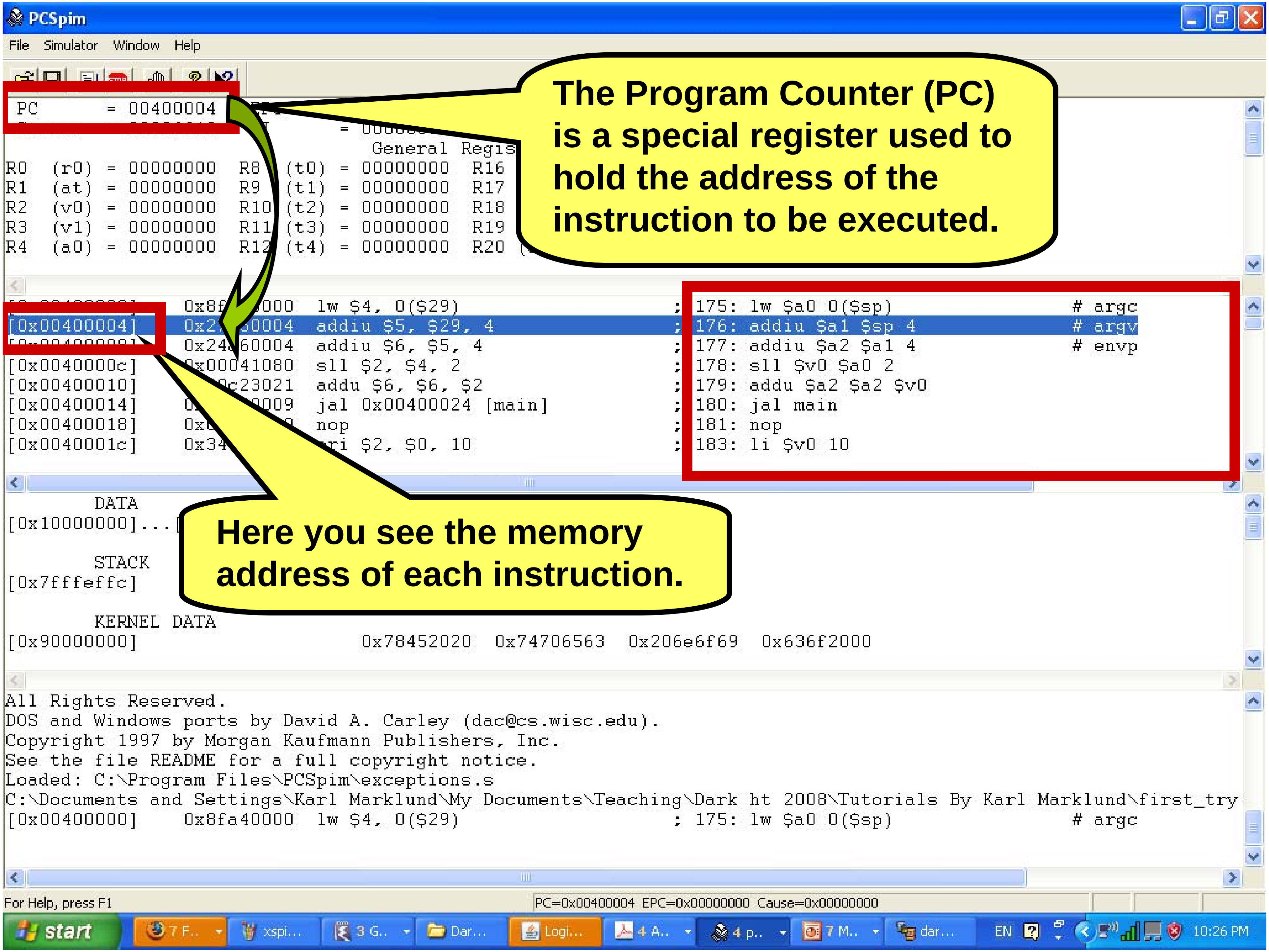
```
All Rights Reserved.
DOS and Windows ports by David A. Carley (d
Copyright 1997 by Morgan Kaufmann Publis
See the file README for a full copyrigh
Loaded: C:\Program Files\PCSpim\excepti
C:\Documents and Settings\Karl Marklund\                y Karl Marklund\first_try
[0x00400000]    0x8fa40000  lw $4, 0($2                                 # argc
```

For Help, press F1

PC=0x00400004 EPC=0x00000000 Cause=0x00000000

After you pressed F10 (single step) once, the blue line moves down one instruction.

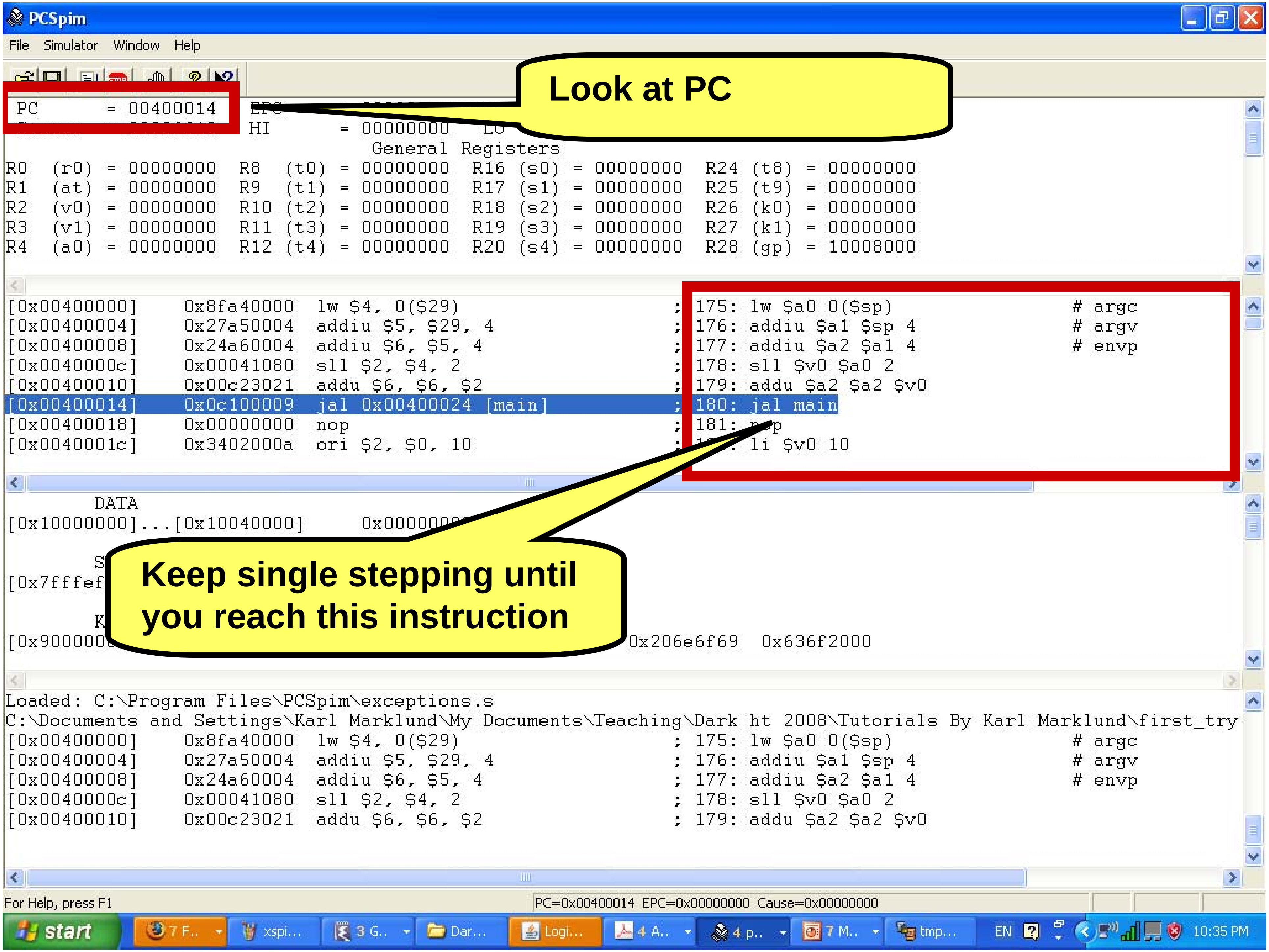
This means that the previous instruction has been executed



```
PC        = 00400004
                                  General Regis
R0  (r0) = 00000000  R8   (t0) = 00000000  R16
R1  (at) = 00000000  R9   (t1) = 00000000  R17
R2  (v0) = 00000000  R10  (t2) = 00000000  R18
R3  (v1) = 00000000  R11  (t3) = 00000000  R19
R4  (a0) = 00000000  R12  (t4) = 00000000  R20

                0x8f   000  lw $4, 0($29)              ; 175: lw $a0 0($sp)              # argc
[0x00400004]    0x2    0004  addiu $5, $29, 4          ; 176: addiu $a1 $sp 4            # argv
                0x24   60004  addiu $6, $5, 4          ; 177: addiu $a2 $a1 4            # envp
[0x0040000c]    x00041080  sll $2, $4, 2               ; 178: sll $v0 $a0 2
[0x00400010]       c23021  addu $6, $6, $2             ; 179: addu $a2 $a2 $v0
[0x00400014]    0     0009  jal 0x00400024 [main]      ; 180: jal main
[0x00400018]    0x          nop                        ; 181: nop
[0x0040001c]    0x34        ri $2, $0, 10              ; 183: li $v0 10

        DATA
[0x10000000]...[

        STACK
[0x7fffeffc]

        KERNEL  DATA
[0x90000000]                 0x78452020  0x74706563  0x206e6f69  0x636f2000

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:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_try
[0x00400000]    0x8fa40000  lw $4, 0($29)              ; 175: lw $a0 0($sp)              # argc

PC=0x00400004  EPC=0x00000000  Cause=0x00000000
```

The Program Counter (PC) is a special register used to hold the address of the instruction to be executed.

Here you see the memory address of each instruction.



Look at PC

```
PC       = 00400014   EPC
                      HI       = 00000000    LO
                          General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

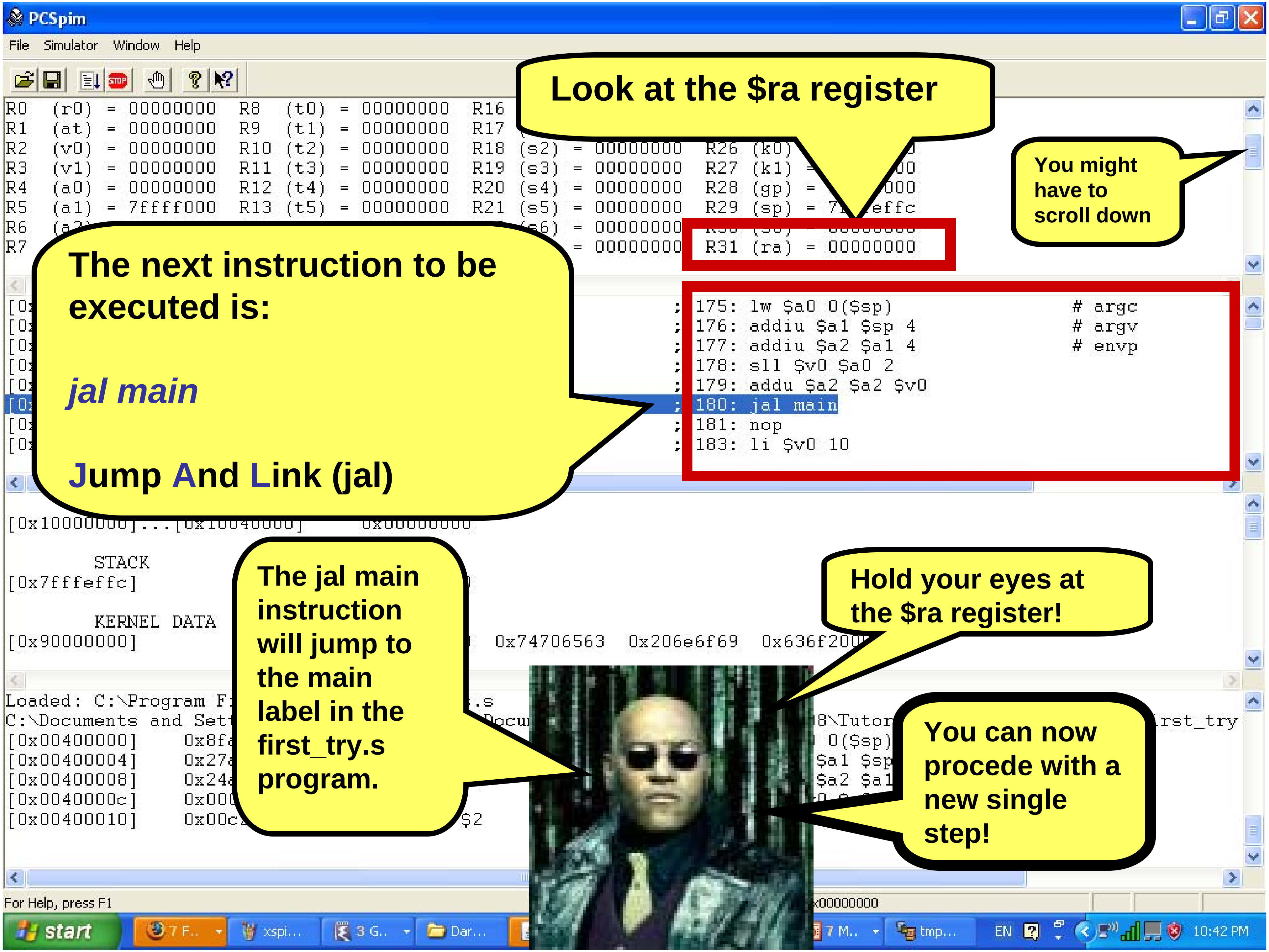
```
[0x00400000]    0x8fa40000  lw $4, 0($29)                ; 175: lw $a0 0($sp)        # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4             ; 176: addiu $a1 $sp 4      # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4              ; 177: addiu $a2 $a1 4      # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]        ; 180: jal main
[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10               ;       li $v0 10
```

```
        DATA
[0x10000000]...[0x10040000]     0x0000000

        S
[0x7fffef

        K
[0x9000000                                              0x206e6f69  0x636f2000
```

Keep single stepping until you reach this instruction

```
Loaded: C:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_try
[0x00400000]    0x8fa40000  lw $4, 0($29)                ; 175: lw $a0 0($sp)        # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4             ; 176: addiu $a1 $sp 4      # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4              ; 177: addiu $a2 $a1 4      # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
```

```
For Help, press F1                PC=0x00400014  EPC=0x00000000  Cause=0x00000000
```



```
R0  (r0) = 00000000  R8  (t0) = 00000000  R16
R1  (at) = 00000000  R9  (t1) = 00000000  R17
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0)
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1)
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp)
R5  (a1) = 7ffff000  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7    effc
R6  (a2                                      (s6) = 00000000
R7                                                = 00000000  R31 (ra) = 00000000
```

```
; 175: lw $a0 0($sp)                # argc
; 176: addiu $a1 $sp 4              # argv
; 177: addiu $a2 $a1 4              # envp
; 178: sll $v0 $a0 2
; 179: addu $a2 $a2 $v0
; 180: jal main
; 181: nop
; 183: li $v0 10
```

```
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]

        KERNEL  DATA
[0x90000000]                     0x74706563  0x206e6f69  0x636f200
```

```
Loaded: C:\Program F                        .s
C:\Documents and Sett                 Docu       8\Tutor              irst_try
[0x00400000]    0x8f                       0($sp)
[0x00400004]    0x27                      $a1 $sp
[0x00400008]    0x24                      $a2 $a1
[0x0040000c]    0x00
[0x00400010]    0x00c                $2
```

Look at the $ra register

You might have to scroll down

The next instruction to be executed is:

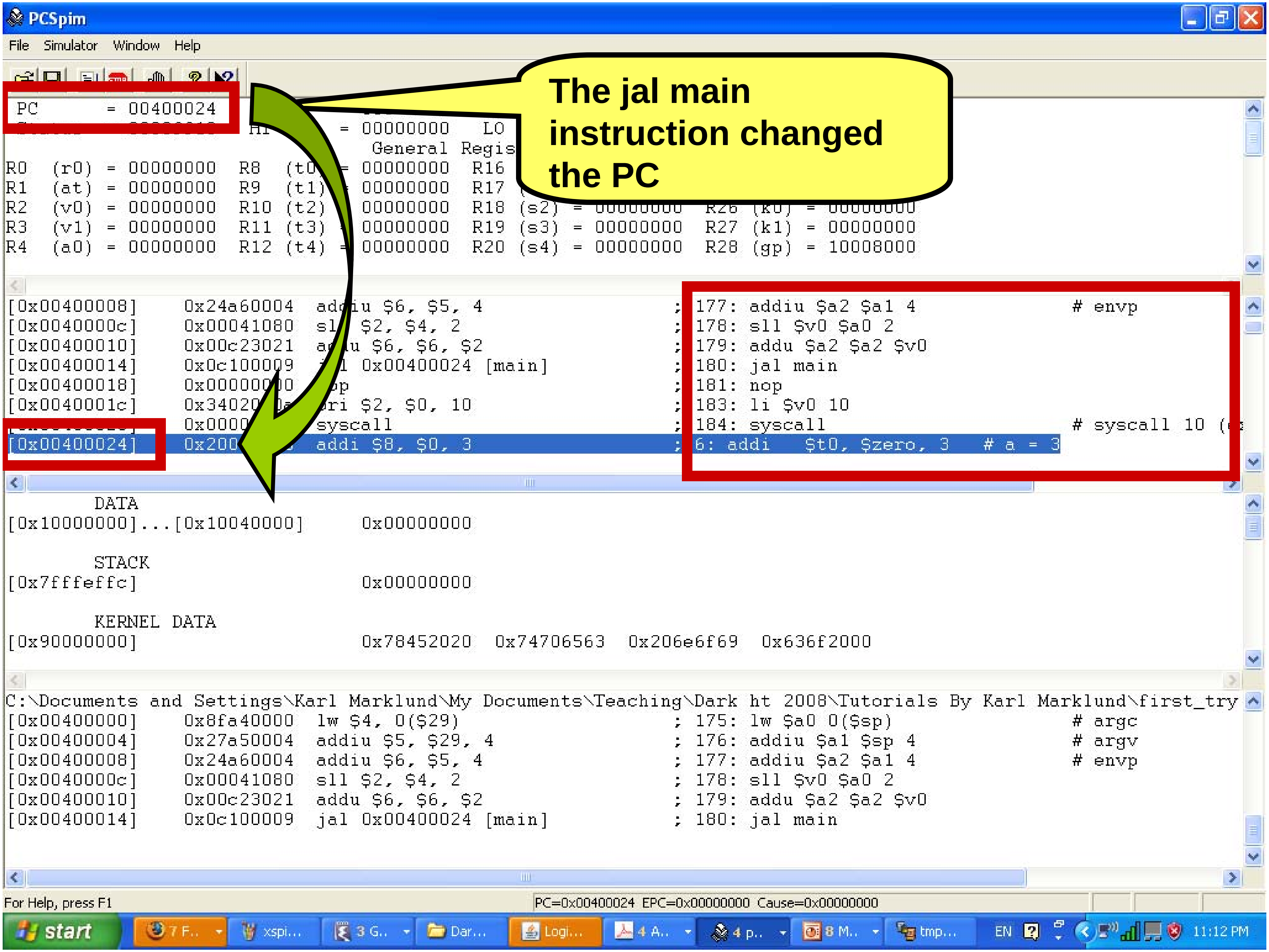
jal main

Jump And Link (jal)

The jal main instruction will jump to the main label in the first_try.s program.

Hold your eyes at the $ra register!

You can now procede with a new single step!



The jal main instruction changed the PC

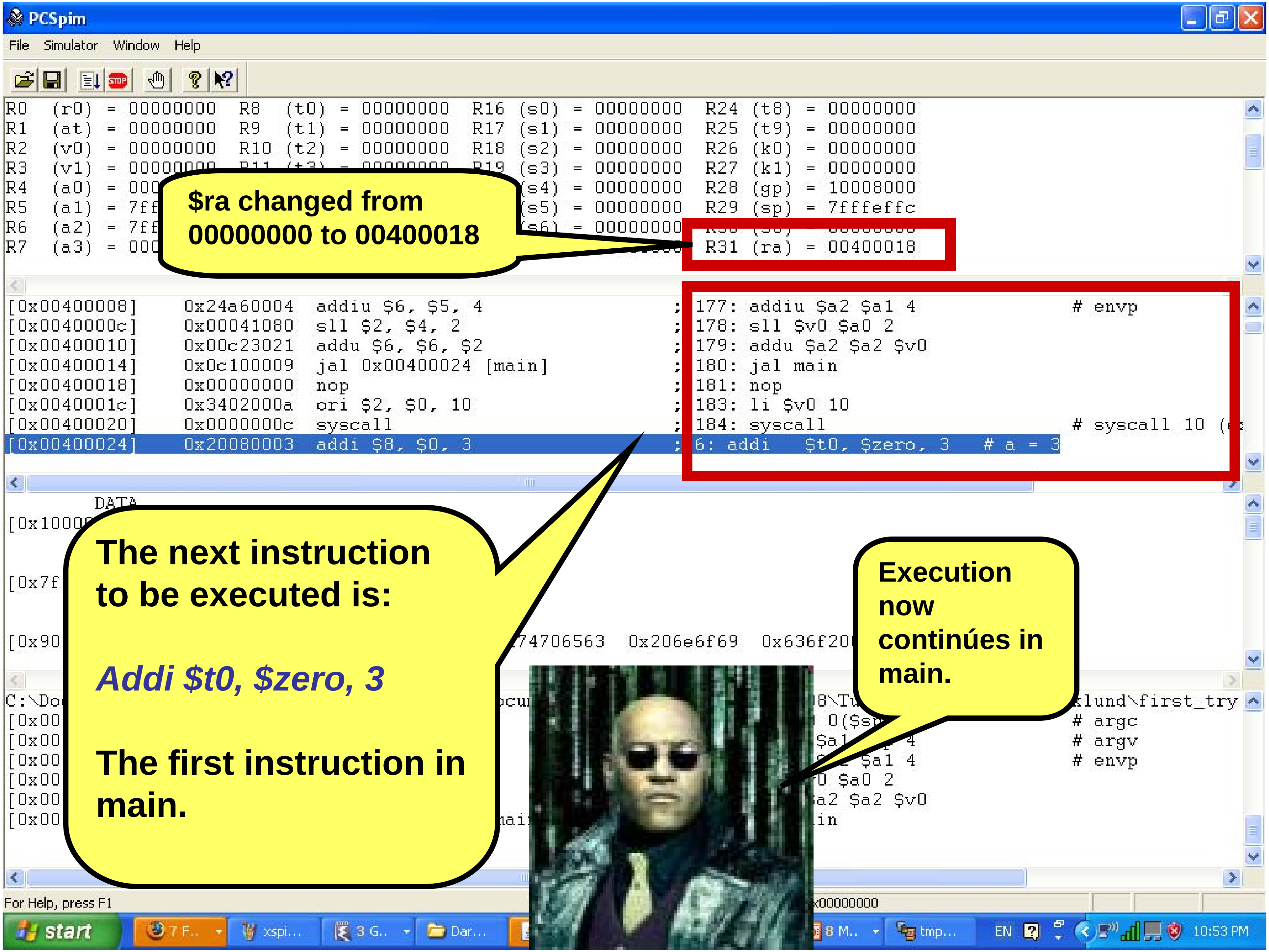
```
PC      = 00400024
                         HI     = 00000000   LO
                                 General Regis
R0  (r0) = 00000000  R8  (t0) = 00000000  R16
R1  (at) = 00000000  R9  (t1)   00000000  R17
R2  (v0) = 00000000  R10 (t2)   00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3)   00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4)   00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400008]    0x24a60004  addiu $6, $5, 4         ; 177: addiu $a2 $a1 4          # envp
[0x0040000c]    0x00041080  sll $2, $4, 2           ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2         ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]   ; 180: jal main
[0x00400018]    0x00000000  nop                     ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10          ; 183: li $v0 10
                0x0000      syscall                 ; 184: syscall                  # syscall 10 (
[0x00400024]    0x200       addi $8, $0, 3          ; 6: addi   $t0, $zero, 3    # a = 3
```

```
        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020  0x74706563  0x206e6f69  0x636f2000
```

```
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_try
[0x00400000]    0x8fa40000  lw $4, 0($29)           ; 175: lw $a0 0($sp)            # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4        ; 176: addiu $a1 $sp 4          # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4         ; 177: addiu $a2 $a1 4          # envp
[0x0040000c]    0x00041080  sll $2, $4, 2           ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2         ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]   ; 180: jal main
```

```
For Help, press F1              PC=0x00400024  EPC=0x00000000  Cause=0x00000000
```



```
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000                       R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 000                                (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 7ff                                (s5) = 00000000  R29 (sp) = 7fffeffc
R6  (a2) = 7ff                                (s6) = 00000000
R7  (a3) = 000                                                 R31 (ra) = 00400018
```

```
[0x00400008]   0x24a60004  addiu $6, $5, 4          ; 177: addiu $a2 $a1 4          # envp
[0x0040000c]   0x00041080  sll $2, $4, 2            ; 178: sll $v0 $a0 2
[0x00400010]   0x00c23021  addu $6, $6, $2          ; 179: addu $a2 $a2 $v0
[0x00400014]   0x0c100009  jal 0x00400024 [main]    ; 180: jal main
[0x00400018]   0x00000000  nop                      ; 181: nop
[0x0040001c]   0x3402000a  ori $2, $0, 10           ; 183: li $v0 10
[0x00400020]   0x0000000c  syscall                  ; 184: syscall                  # syscall 10 (
[0x00400024]   0x20080003  addi $8, $0, 3           ; 6: addi   $t0, $zero, 3   # a = 3
```

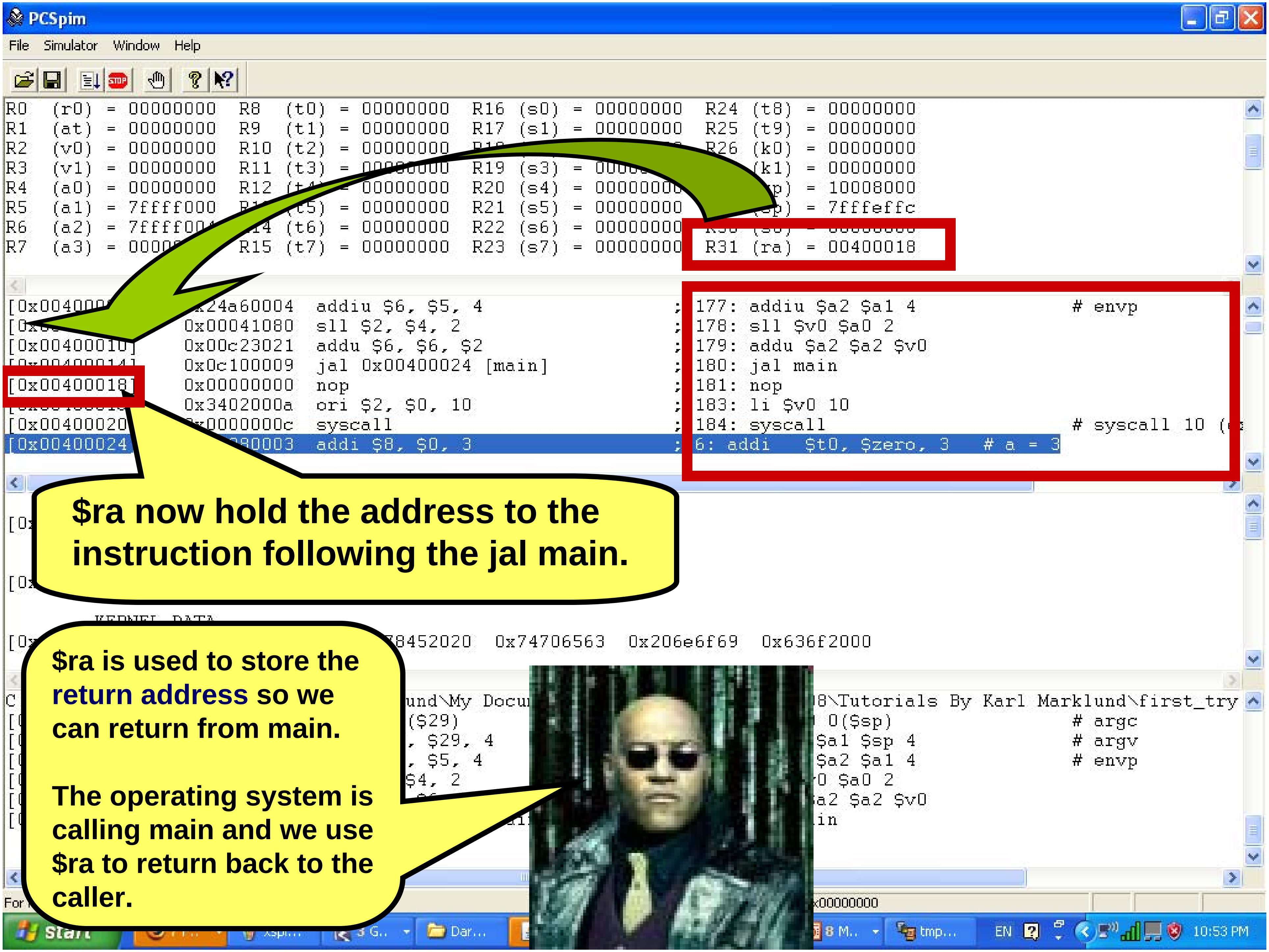
$ra changed from 00000000 to 00400018

The next instruction to be executed is:

Addi $t0, $zero, 3

The first instruction in main.

Execution now continíues in main.



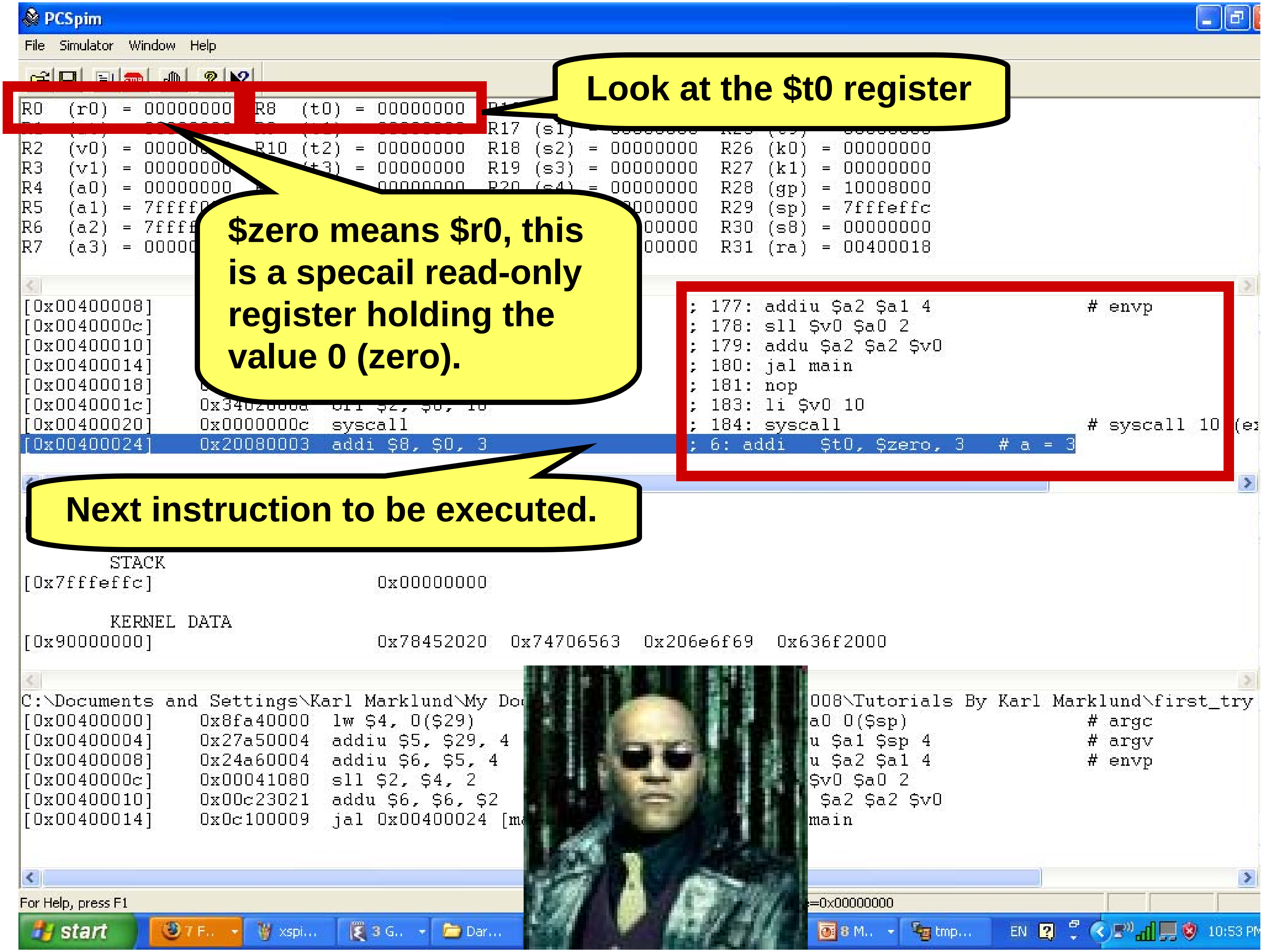
```
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000                       R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) =           R19 (s3) = 0000          (k1) = 00000000
R4  (a0) = 00000000  R12                  R20 (s4) = 00000000         p) = 10008000
R5  (a1) = 7ffff000                 t5) = 00000000  R21 (s5) = 00000000   p) = 7fffeffc
R6  (a2) = 7ffff00       4  (t6) = 00000000  R22 (s6) = 00000000
R7  (a3) = 0000          R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400018

[0x00400           24a60004  addiu $6, $5, 4        ; 177: addiu $a2 $a1 4          # envp
[0x                0x00041080  sll $2, $4, 2        ; 178: sll $v0 $a0 2
[0x00400010]       0x00c23021  addu $6, $6, $2      ; 179: addu $a2 $a2 $v0
                   0x0c100009  jal 0x00400024 [main] ; 180: jal main
[0x00400018]       0x00000000  nop                  ; 181: nop
                   0x3402000a  ori $2, $0, 10       ; 183: li $v0 10
[0x00400020]       0000000c    syscall              ; 184: syscall                  # syscall 10 (
[0x00400024        80003       addi $8, $0, 3       ; 6: addi  $t0, $zero, 3   # a = 3
```

$ra now hold the address to the instruction following the jal main.

```
         KERNEL DATA
[0x          8452020  0x74706563  0x206e6f69  0x636f2000

C                   und\My Docu            8\Tutorials By Karl Marklund\first_try
[                   ($29)                    0($sp)                # argc
[                   , $29, 4                 $a1 $sp 4             # argv
[                   , $5, 4                  $a2 $a1 4             # envp
[                   $4, 2                    0 $a0 2
[                                            a2 $a2 $v0
[                                            in
```

$ra is used to store the return address so we can return from main.

The operating system is calling main and we use $ra to return back to the caller.



Look at the $t0 register

$zero means $r0, this is a specail read-only register holding the value 0 (zero).

Next instruction to be executed.

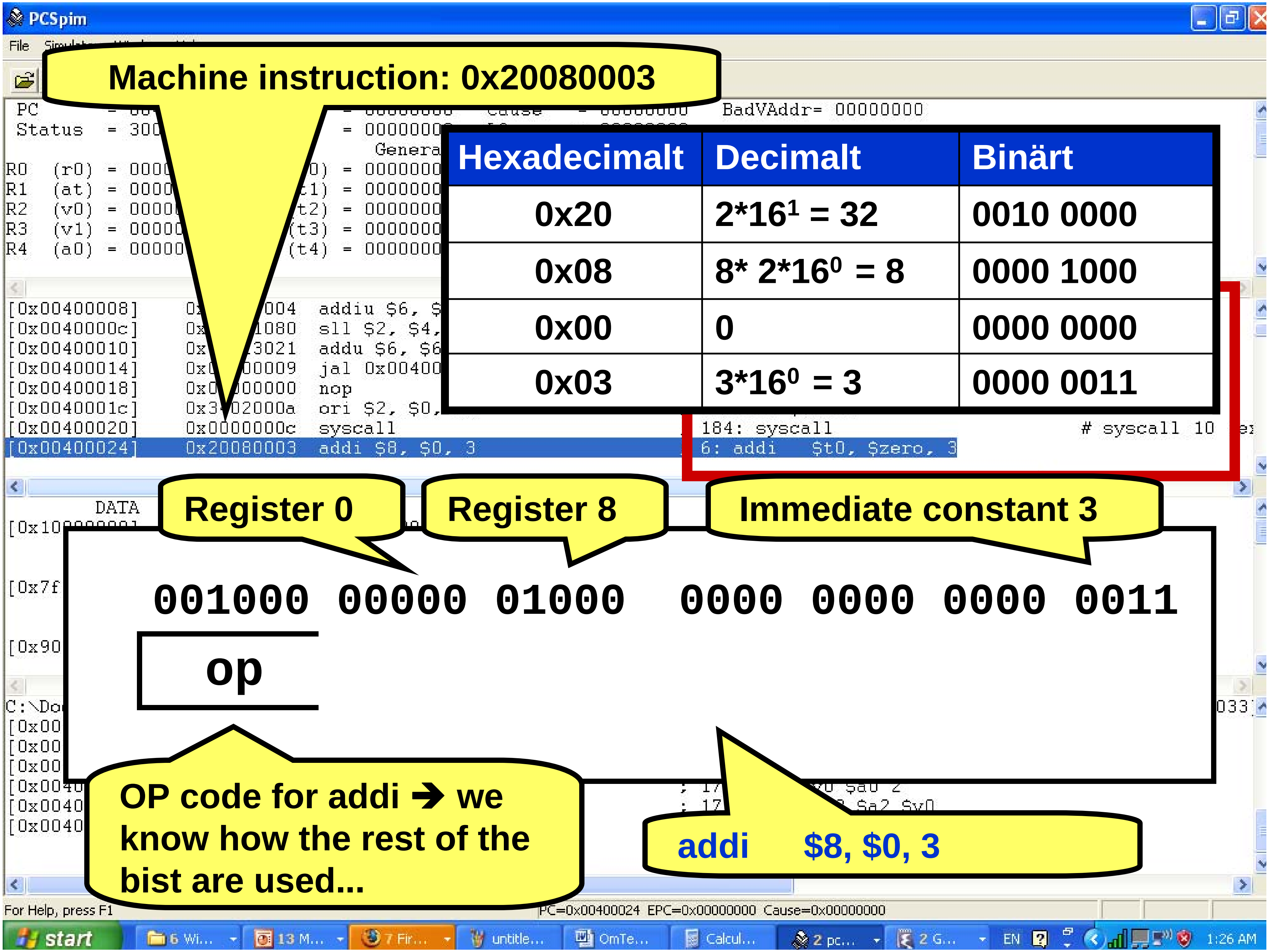
```
R0  (r0) = 00000000  R8  (t0) = 00000000
                                          R17 (s1)
R2  (v0) = 00000     R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000           = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000                                            R28 (gp) = 10008000
R5  (a1) = 7ffff                                      000000   R29 (sp) = 7fffeffc
R6  (a2) = 7fff                                        00000   R30 (s8) = 00000000
R7  (a3) = 0000                                        00000   R31 (ra) = 00400018
```

```
[0x00400008]                                       ; 177: addiu $a2 $a1 4               # envp
[0x0040000c]                                       ; 178: sll $v0 $a0 2
[0x00400010]                                       ; 179: addu $a2 $a2 $v0
[0x00400014]                                       ; 180: jal main
[0x00400018]                                       ; 181: nop
[0x0040001c]                                       ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                ; 184: syscall                       # syscall 10 (e
[0x00400024]    0x20080003  addi $8, $0, 3         ; 6: addi    $t0, $zero, 3   # a = 3
```

```
        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020  0x74706563  0x206e6f69  0x636f2000
```

```
C:\Documents and Settings\Karl Marklund\My Do                008\Tutorials By Karl Marklund\first_try
[0x00400000]    0x8fa40000  lw $4, 0($29)                    a0 0($sp)                   # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4                 u $a1 $sp 4                 # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4                  u $a2 $a1 4                 # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                    $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2                   $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [m                main
```

=0x00000000



```
PC       = 00                                                  BadVAddr= 00000000
Status   = 300                                  = 0000000
                                                  Genera
R0  (r0) = 0000                     0)  = 0000000
R1  (at) = 0000                     t1) = 0000000
R2  (v0) = 0000                     t2) = 0000000
R3  (v1) = 0000                     (t3) = 0000000
R4  (a0) = 0000                     (t4) = 0000000

[0x00400008]   0x        004   addiu $6, $
[0x0040000c]   0x        1080  sll $2, $4,
[0x00400010]   0x        3021  addu $6, $6
[0x00400014]   0x0       00009 jal 0x00400
[0x00400018]   0x0       00000 nop
[0x0040001c]   0x3   02000a    ori $2, $0,
[0x00400020]   0x0000000c      syscall                 184: syscall                        # syscall 10 ex
[0x00400024]   0x20080003      addi $8, $0, 3          6: addi    $t0, $zero, 3

        DATA
[0x10
[0x7f
[0x90
C:\Do                                                                                           033]
[0x00
[0x00
[0x00
[0x0040
[0x0040
[0x0040
                                                         ; 17          0 $a0 2
                                                         ; 17          2 $a2 $v0
For Help, press F1                         PC=0x00400024  EPC=0x00000000  Cause=0x00000000
```

Machine instruction: 0x20080003

Hexadecimalt	Decimalt	Binärt
0x20	$2*16^1 = 32$	0010 0000
0x08	$8* 2*16^0 = 8$	0000 1000
0x00	0	0000 0000
0x03	$3*16^0 = 3$	0000 0011

Register 0

Register 8

Immediate constant 3

001000 00000 01000 0000 0000 0000 0011

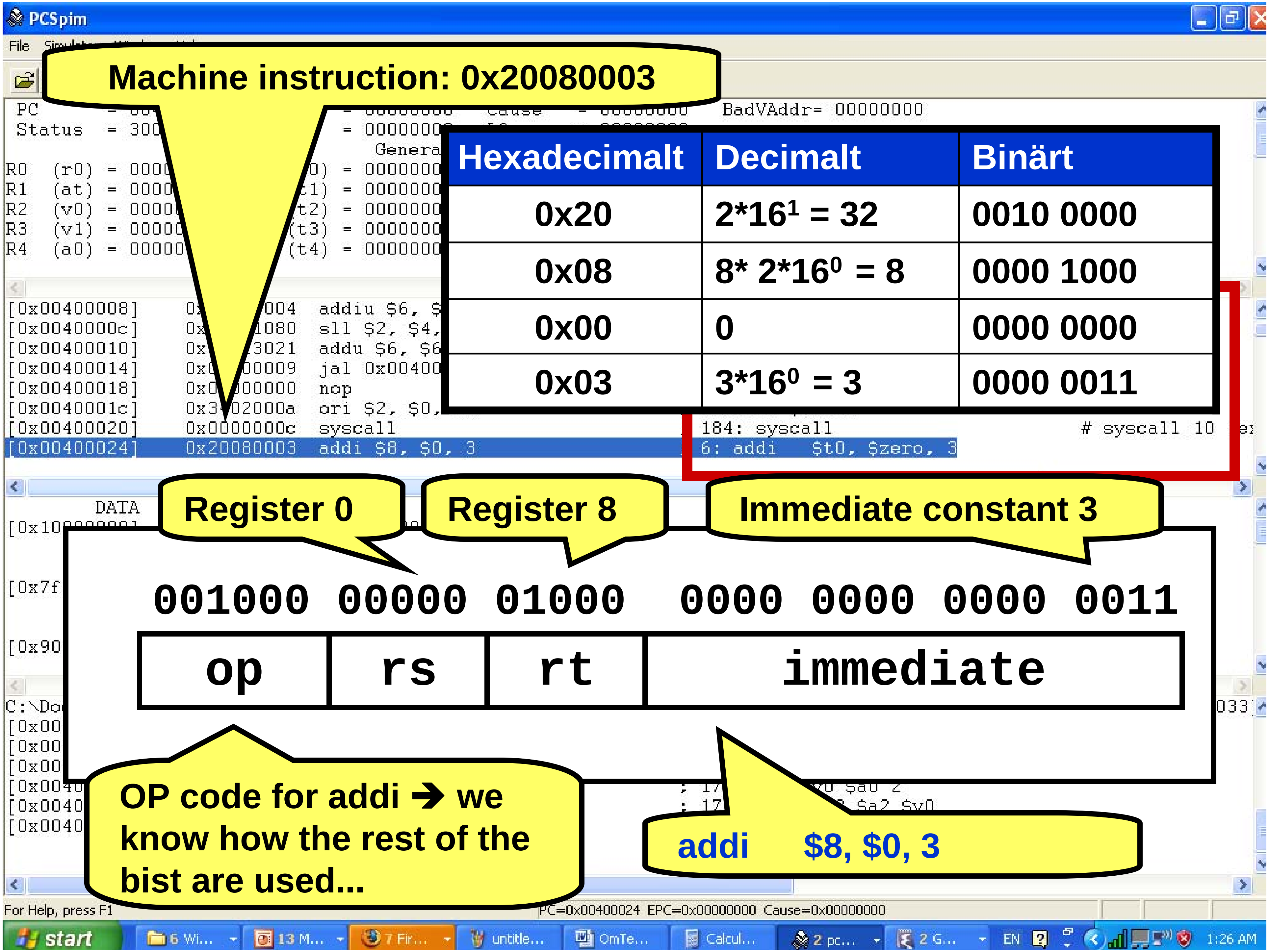
op

OP code for addi → we know how the rest of the bist are used...

addi $8, $0, 3

Machine instruction: 0x20080003

Hexadecimalt	Decimalt	Binärt
0x20	$2*16^1 = 32$	0010 0000
0x08	$8* 2*16^0 = 8$	0000 1000
0x00	0	0000 0000
0x03	$3*16^0 = 3$	0000 0011



```
PC       =                         BadVAddr= 00000000
Status   = 300                   = 0000000
                         Genera
R0  (r0) = 0000          0)  = 0000000
R1  (at) = 0000          t1) = 0000000
R2  (v0) = 0000          t2) = 0000000
R3  (v1) = 0000          (t3) = 0000000
R4  (a0) = 00000         (t4) = 0000000

[0x00400008]   0x     004   addiu $6, $
[0x0040000c]   0x    1080   sll $2, $4,
[0x00400010]   0x   3021    addu $6, $6
[0x00400014]   0x0  00009   jal 0x00400
[0x00400018]   0x0  00000   nop
[0x0040001c]   0x3 02000a   ori $2, $0,
[0x00400020]   0x0000000c   syscall                     184: syscall                  # syscall 10
[0x00400024]   0x20080003   addi $8, $0, 3              6: addi    $t0, $zero, 3

[0x10
[0x7f
[0x90
C:\Do
[0x00
[0x00
[0x00
[0x0040
[0x0040
[0x0040
```

For Help, press F1

PC=0x00400024 EPC=0x00000000 Cause=0x00000000

Register 0

Register 8

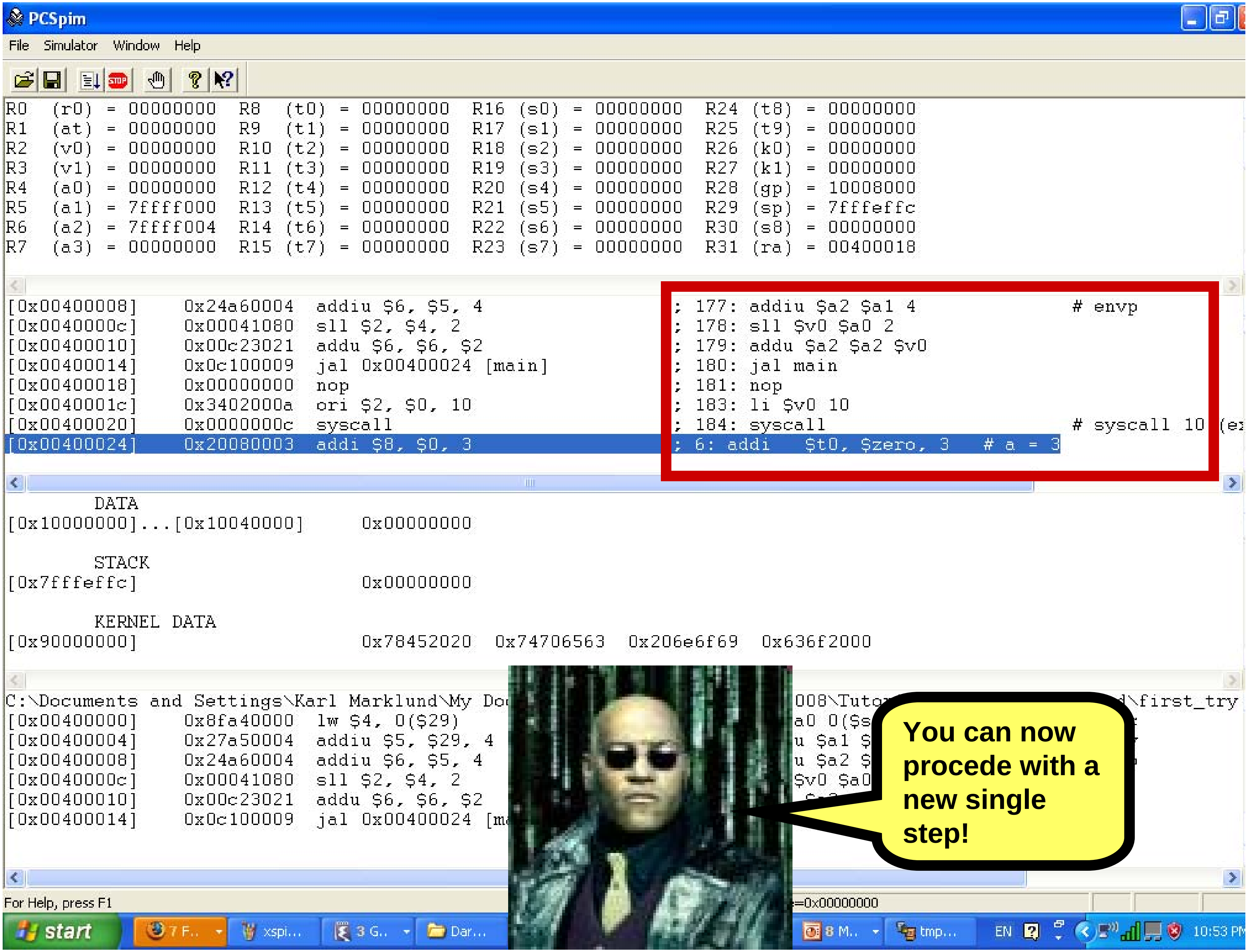
Immediate constant 3

```
001000  00000  01000   0000 0000 0000 0011
```

op	rs	rt	immediate

OP code for addi → we know how the rest of the bist are used...

addi $8, $0, 3



```
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 7ffff000  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffeffc
R6  (a2) = 7ffff004  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400018
```

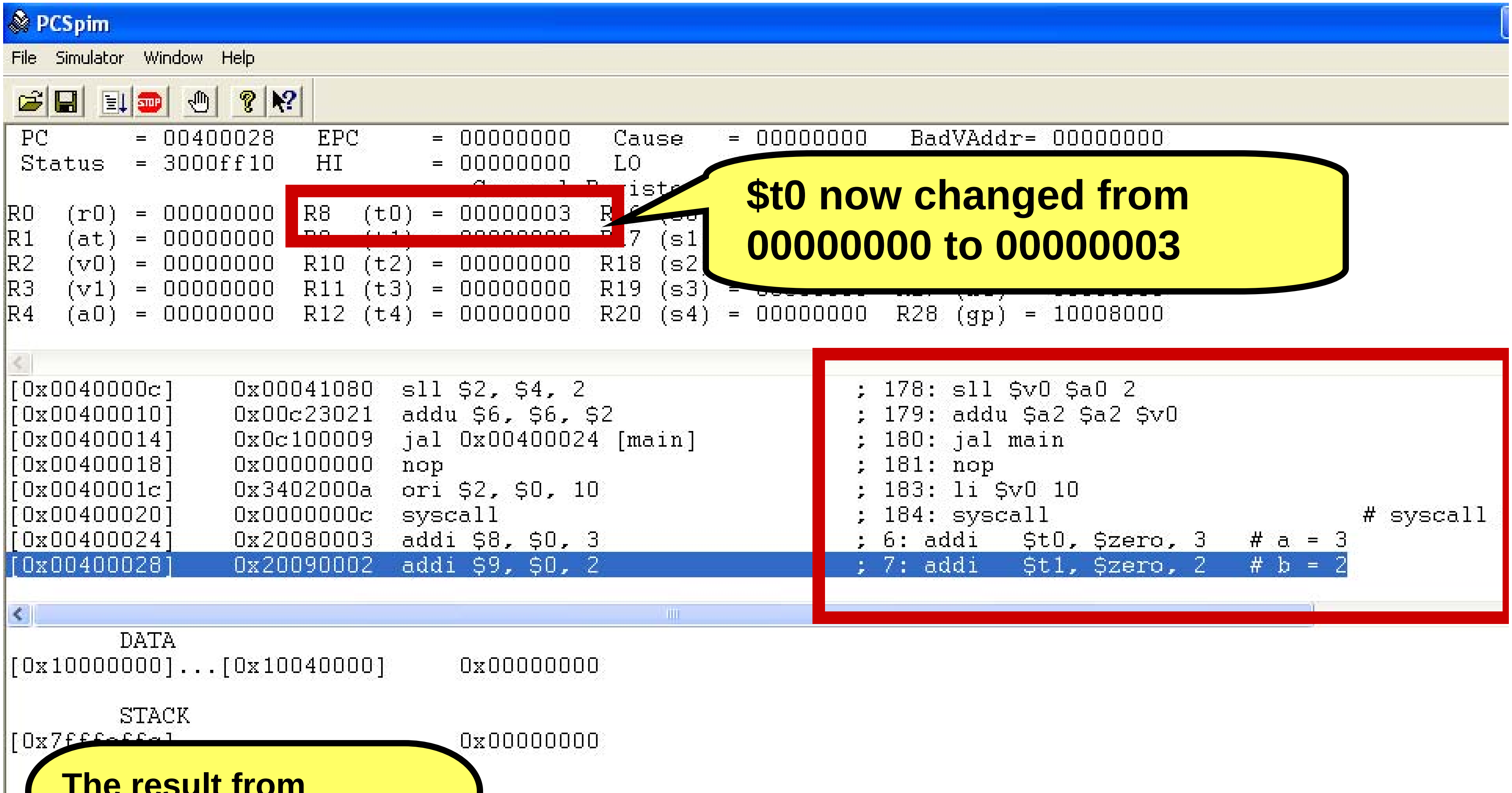
```
[0x00400008]    0x24a60004  addiu $6, $5, 4          ; 177: addiu $a2 $a1 4            # envp
[0x0040000c]    0x00041080  sll $2, $4, 2            ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2          ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]    ; 180: jal main
[0x00400018]    0x00000000  nop                      ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10           ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                  ; 184: syscall                    # syscall 10 (ex
[0x00400024]    0x20080003  addi $8, $0, 3           ; 6: addi    $t0, $zero, 3    # a = 3
```

```
        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020  0x74706563  0x206e6f69  0x636f2000
```

```
C:\Documents and Settings\Karl Marklund\My Doc            008\Tuto                d\first_try
[0x00400000]    0x8fa40000  lw $4, 0($29)                a0 0($s
[0x00400004]    0x27a50004  addiu $5, $29, 4             u $a1 $
[0x00400008]    0x24a60004  addiu $6, $5, 4              u $a2 $
[0x0040000c]    0x00041080  sll $2, $4, 2                $v0 $a0
[0x00400010]    0x00c23021  addu $6, $6, $2
[0x00400014]    0x0c100009  jal 0x00400024 [ma
```

```
For Help, press F1                                       =0x00000000
```

You can now procede with a new single step!



```
PC       = 00400028    EPC      = 00000000    Cause    = 00000000    BadVAddr= 00000000
Status   = 3000ff10    HI       = 00000000    LO

R0  (r0) = 00000000  R8  (t0) = 00000003  R
R1  (at) = 00000000                         R17 (s1
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) =
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]        ; 180: jal main
[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10               ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                      ; 184: syscall                      # syscall
[0x00400024]    0x20080003  addi $8, $0, 3               ; 6: addi    $t0, $zero, 3   # a = 3
[0x00400028]    0x20090002  addi $9, $0, 2               ; 7: addi    $t1, $zero, 2   # b = 2
```

```
        DATA
[0x10000000]...[0x10040000]      0x00000000

        STACK
                                 0x00000000

                                 78452020                          0x636f2000

                                 0($29)                            w $a0 0            argc
                                                                   ddiu $a            argv
                                                                   ddiu $a            envp
                                 6, $6, $2                         ddu
                                 0x00400024                        al main
[0x00400024]    0x20080003  addi $8, $0, 3                         i    $t0, $zero, 3   # a = 3
```

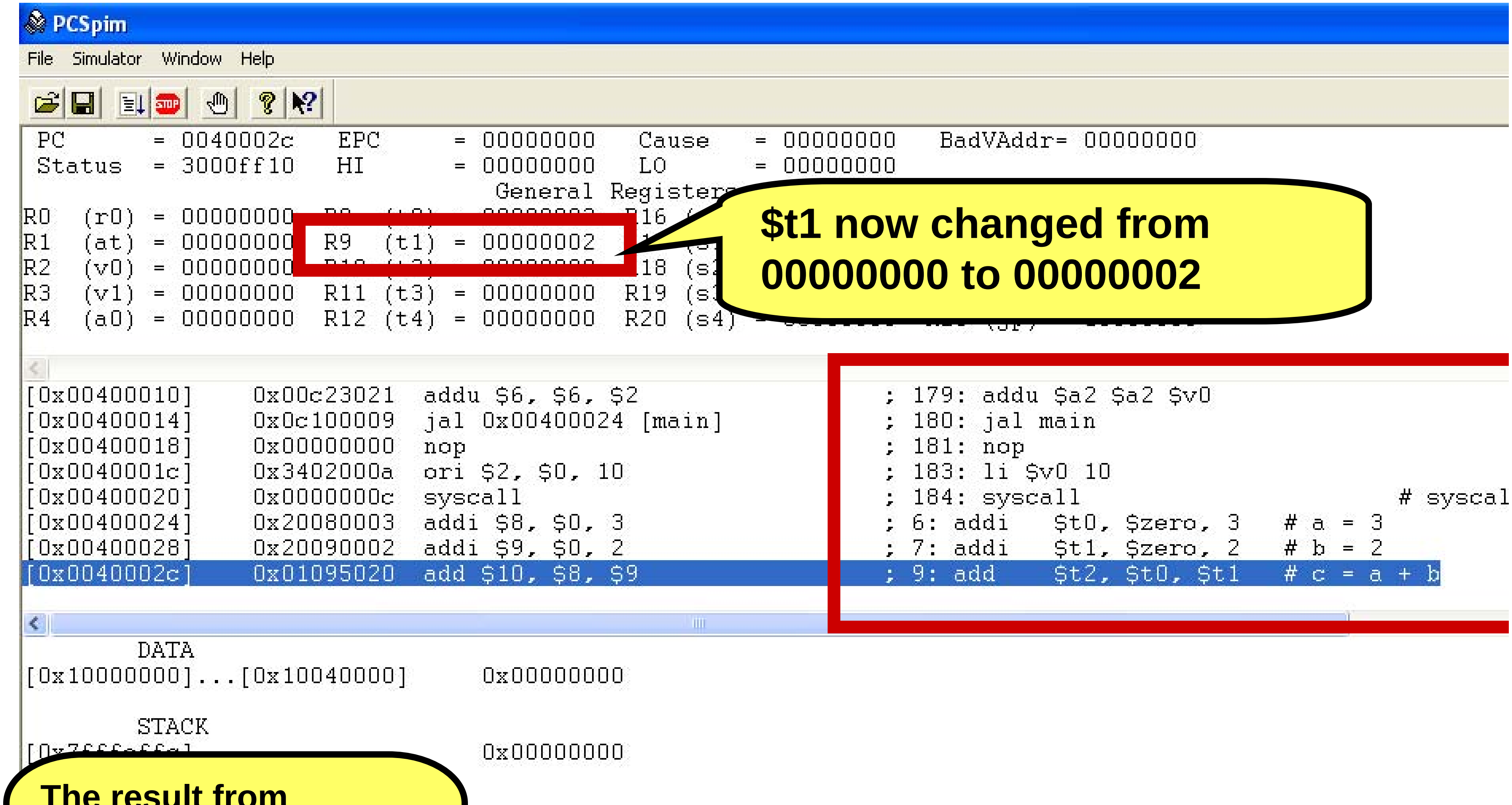
$t0 now changed from 00000000 to 00000003

The result from

addi $t0, $zero, 3

, can now be seen in the $t0 register.

You can now procede with a new single step!



```
PC       = 0040002c   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                General Registers
R0  (r0) = 00000000
R1  (at) = 00000000   R9  (t1) = 00000002
R2  (v0) = 00000000                          R18 (s
R3  (v1) = 00000000   R11 (t3) = 00000000   R19 (s
R4  (a0) = 00000000   R12 (t4) = 00000000   R20 (s4)

[0x00400010]    0x00c23021  addu $6, $6, $2          ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]    ; 180: jal main
[0x00400018]    0x00000000  nop                      ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10           ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                  ; 184: syscall                    # syscal
[0x00400024]    0x20080003  addi $8, $0, 3           ; 6: addi   $t0, $zero, 3   # a = 3
[0x00400028]    0x20090002  addi $9, $0, 2           ; 7: addi   $t1, $zero, 2   # b = 2
[0x0040002c]    0x01095020  add $10, $8, $9          ; 9: add    $t2, $t0, $t1   # c = a + b

        DATA
[0x10000000]...[0x10040000]      0x00000000

        STACK
                                 0x00000000

                                 0x7845202                0x636f2000

                                iu $5, $29,           addiu                        # argv
                                                      addiu                        # envp
                                                      sll
                                 0x00400024           jal
                                adi $8, $0,           addi    $                          3
[0x00400028]    0x20090002  addi $9, $0, 2          addi    $t1, $zero, 2     # b = 2
```

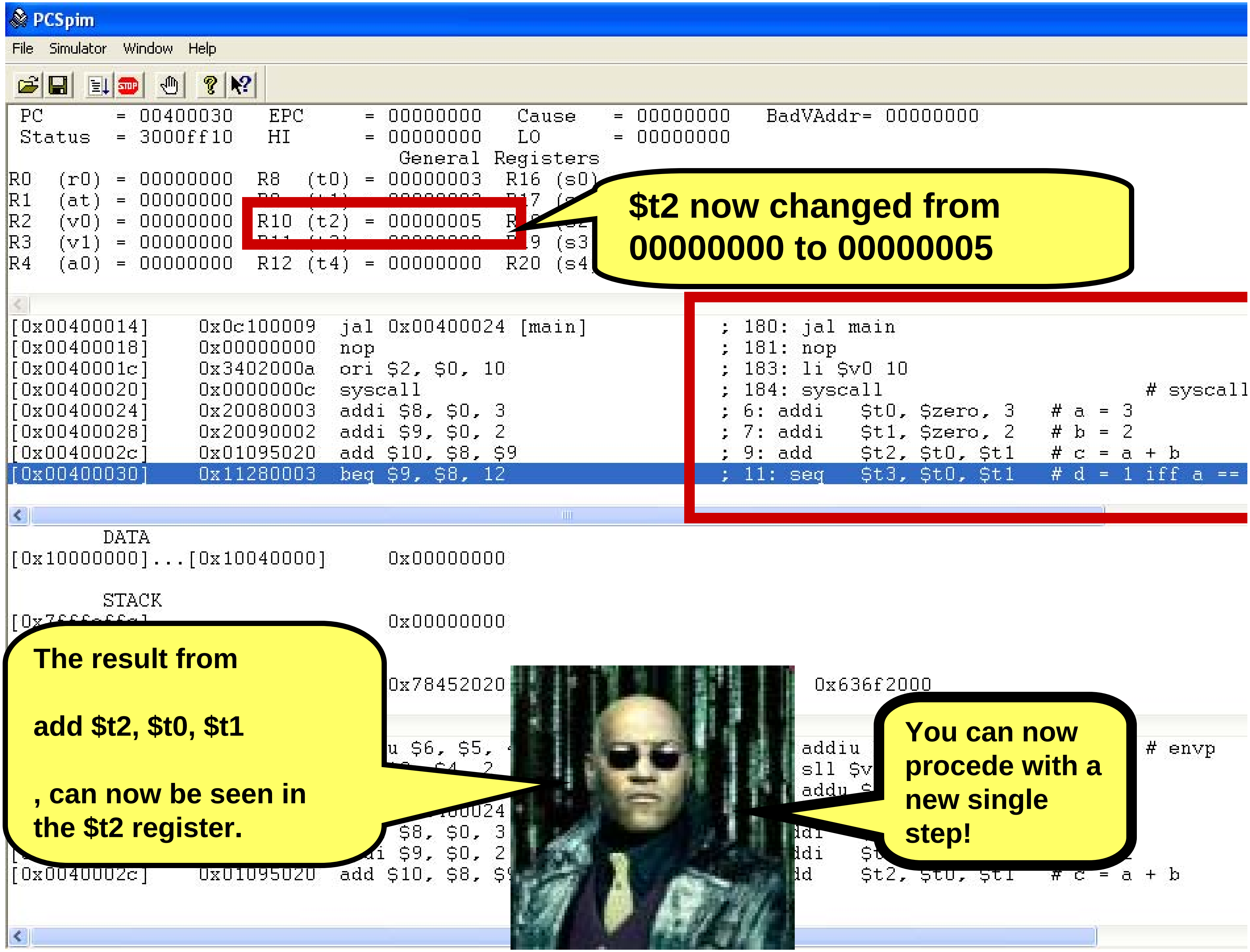
$t1 now changed from 00000000 to 00000002

The result from

addi $t1, $zero, 2

, can now be seen in the $t1 register.

You can now procede with a new single step!



```
 PC       = 00400030   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                General Registers
R0  (r0) = 00000000   R8  (t0) = 00000003  R16 (s0)
R1  (at) = 00000000
R2  (v0) = 00000000   R10 (t2) = 00000005
R3  (v1) = 00000000                          9 (s3
R4  (a0) = 00000000   R12 (t4) = 00000000  R20 (s4

[0x00400014]    0x0c100009  jal 0x00400024 [main]      ; 180: jal main
[0x00400018]    0x00000000  nop                        ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10             ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                    ; 184: syscall                     # syscall
[0x00400024]    0x20080003  addi $8, $0, 3             ; 6: addi    $t0, $zero, 3   # a = 3
[0x00400028]    0x20090002  addi $9, $0, 2             ; 7: addi    $t1, $zero, 2   # b = 2
[0x0040002c]    0x01095020  add $10, $8, $9            ; 9: add     $t2, $t0, $t1   # c = a + b
[0x00400030]    0x11280003  beq $9, $8, 12             ; 11: seq    $t3, $t0, $t1   # d = 1 iff a ==

        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
                                0x00000000

                                0x78452020                        0x636f2000

[0x0040002c]    0x01095020  add $10, $8, $9                           $t2, $t0, $t1   # c = a + b
```

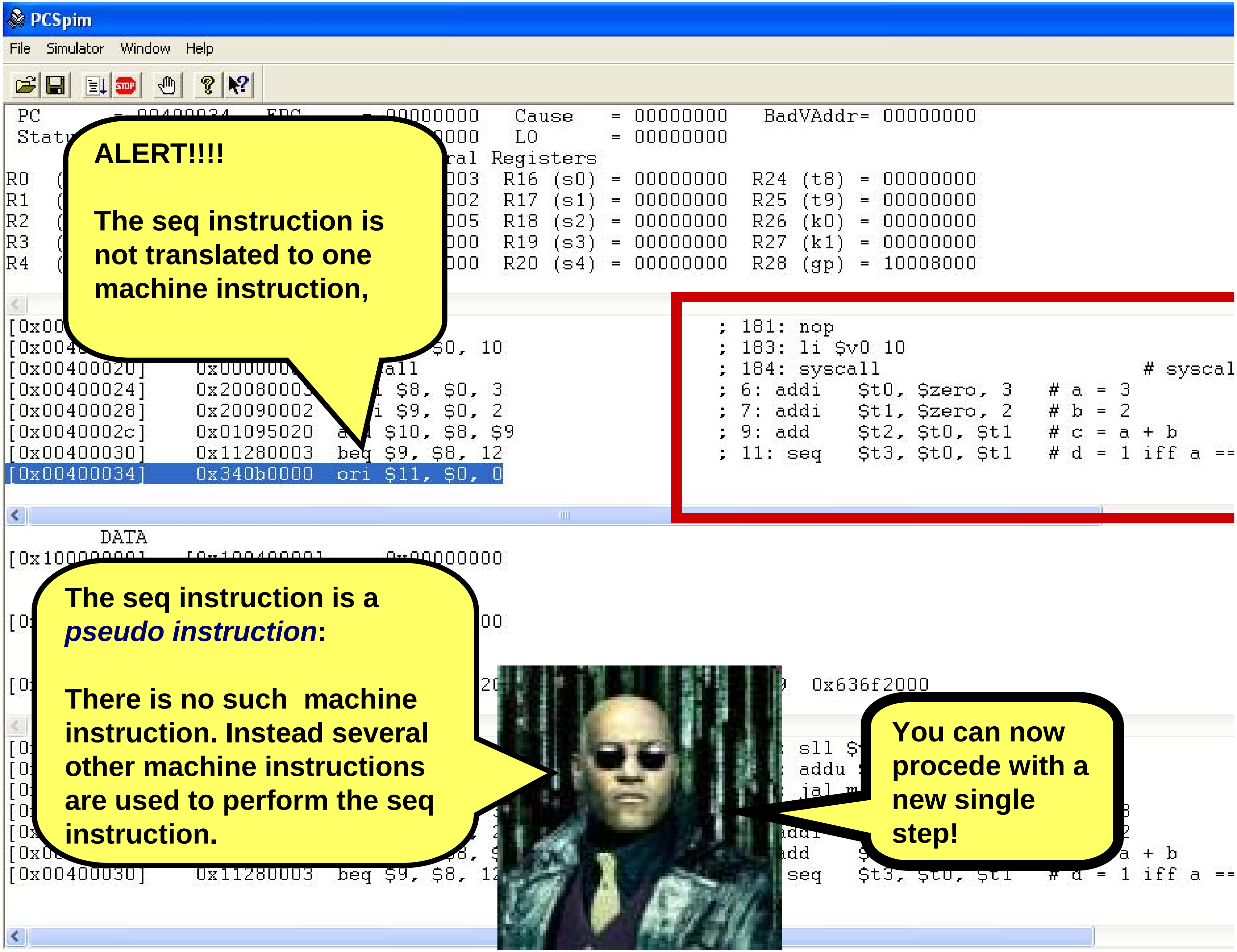
$t2 now changed from 00000000 to 00000005

The result from

add $t2, $t0, $t1

, can now be seen in the $t2 register.

You can now procede with a new single step!



```
PC         = 00400034   EPC        = 00000000   Cause     = 00000000   BadVAddr= 00000000
Stat                                     0000   LO        = 00000000
                                        ral Registers
R0    (                                  003   R16 (s0) = 00000000   R24 (t8) = 00000000
R1    (                                  002   R17 (s1) = 00000000   R25 (t9) = 00000000
R2    (                                  005   R18 (s2) = 00000000   R26 (k0) = 00000000
R3    (                                  000   R19 (s3) = 00000000   R27 (k1) = 00000000
R4    (                                  000   R20 (s4) = 00000000   R28 (gp) = 10008000
```

```
[0x00                                                ; 181: nop
[0x004                               $0, 10          ; 183: li $v0 10
[0x00400020]      0x000000       all                 ; 184: syscall                            # syscal
[0x00400024]      0x2008000      i $8, $0, 3         ; 6: addi     $t0, $zero, 3     # a = 3
[0x00400028]      0x20090002     i $9, $0, 2         ; 7: addi     $t1, $zero, 2     # b = 2
[0x0040002c]      0x01095020  a  d $10, $8, $9       ; 9: add      $t2, $t0, $t1     # c = a + b
[0x00400030]      0x11280003  beq $9, $8, 12         ; 11: seq     $t3, $t0, $t1     # d = 1 iff a ==
[0x00400034]      0x340b0000  ori $11, $0, 0
```

```
        DATA
[0x10000000]    [0x10040000]       0x00000000
```

```
[0x00400030]      0x11280003  beq $9, $8, 12                 seq      $t3, $t0, $t1     # d = 1 iff a ==
```

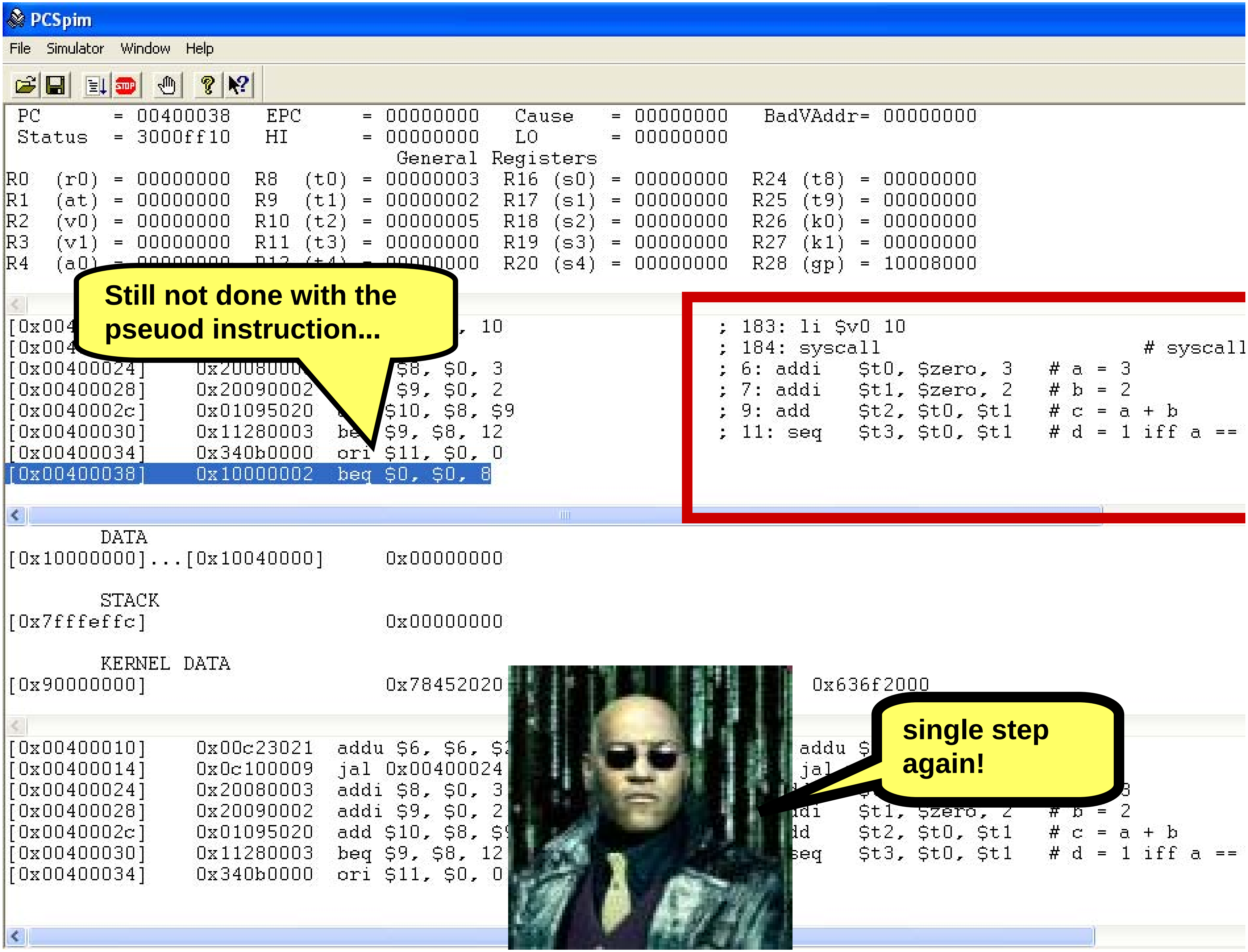
ALERT!!!!

The seq instruction is not translated to one machine instruction,

The seq instruction is a *pseudo instruction*:

There is no such machine instruction. Instead several other machine instructions are used to perform the seq instruction.

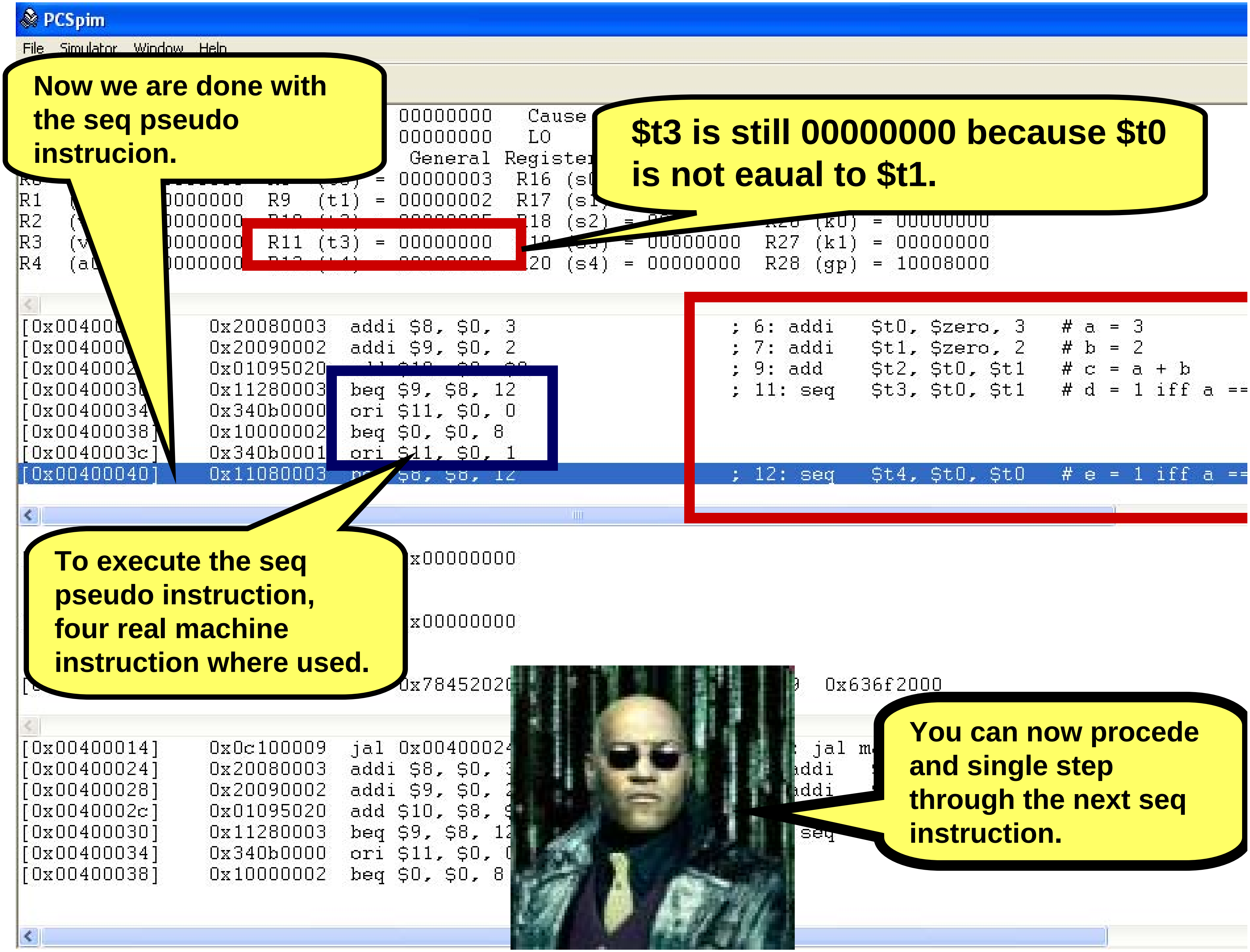
You can now procede with a new single step!



```
 PC       = 00400038    EPC      = 00000000    Cause    = 00000000    BadVAddr= 00000000
 Status   = 3000ff10    HI       = 00000000    LO       = 00000000
                                   General  Registers
R0   (r0) = 00000000   R8  (t0) = 00000003   R16 (s0) = 00000000   R24 (t8) = 00000000
R1   (at) = 00000000   R9  (t1) = 00000002   R17 (s1) = 00000000   R25 (t9) = 00000000
R2   (v0) = 00000000   R10 (t2) = 00000005   R18 (s2) = 00000000   R26 (k0) = 00000000
R3   (v1) = 00000000   R11 (t3) = 00000000   R19 (s3) = 00000000   R27 (k1) = 00000000
R4   (a0)                                    R20 (s4) = 00000000   R28 (gp) = 10008000

[0x004                                , 10              ; 183: li $v0 10
[0x004                                                  ; 184: syscall                        # syscall
[0x00400024]     0x2008000         $8, $0, 3            ; 6: addi    $t0, $zero, 3    # a = 3
[0x00400028]     0x20090002        $9, $0, 2            ; 7: addi    $t1, $zero, 2    # b = 2
[0x0040002c]     0x01095020       $10, $8, $9           ; 9: add     $t2, $t0, $t1    # c = a + b
[0x00400030]     0x11280003   be  $9, $8, 12            ; 11: seq    $t3, $t0, $t1    # d = 1 iff a ==
[0x00400034]     0x340b0000   ori $11, $0, 0
[0x00400038]     0x10000002   beq $0, $0, 8

         DATA
[0x10000000]...[0x10040000]       0x00000000

         STACK
[0x7fffeffc]                      0x00000000

         KERNEL  DATA
[0x90000000]                      0x78452020                      0x636f2000

[0x00400010]     0x00c23021   addu $6, $6, $                      addu $
[0x00400014]     0x0c100009   jal 0x00400024                      jal
[0x00400024]     0x20080003   addi $8, $0, 3                                                  3
[0x00400028]     0x20090002   addi $9, $0, 2                      ddi    $t1, $zero, 2    # b = 2
[0x0040002c]     0x01095020   add $10, $8, $                      dd     $t2, $t0, $t1    # c = a + b
[0x00400030]     0x11280003   beq $9, $8, 12                      seq    $t3, $t0, $t1    # d = 1 iff a ==
[0x00400034]     0x340b0000   ori $11, $0, 0
```

Still not done with the pseuod instruction...

single step again!



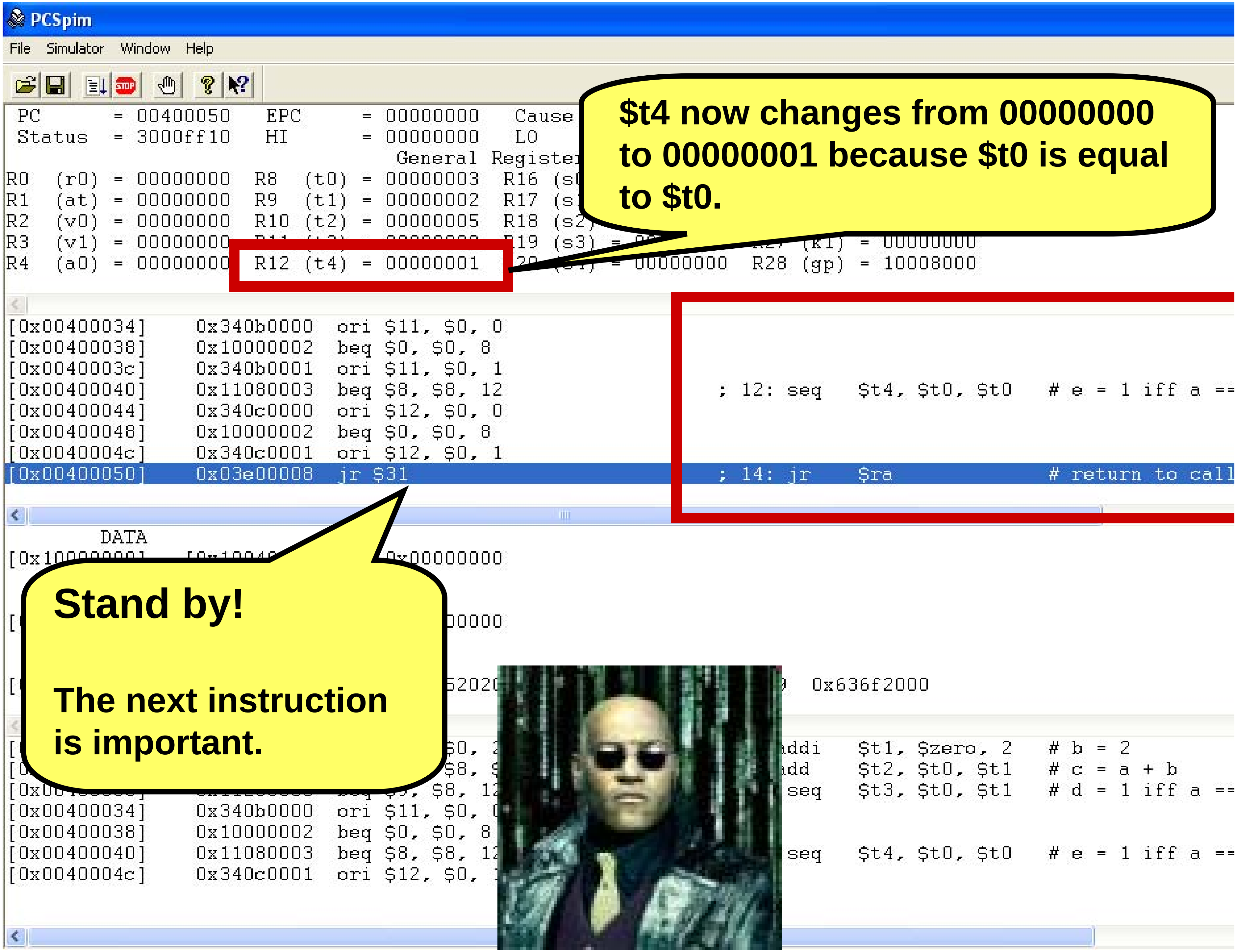
```
                    00000000   Cause
                    00000000   LO
                     General Register
R0                  R8  (t0) = 00000003  R16 (s0
R1   (    0000000   R9  (t1) = 00000002  R17 (s1
R2   (    0000000   R10 (t2) = 00000005  R18 (s2) = 00              R26 (k0) = 00000000
R3   (v   0000000   R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4   (a   0000000   R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400         0x20080003  addi $8, $0, 3        ; 6: addi   $t0, $zero, 3    # a = 3
[0x00400         0x20090002  addi $9, $0, 2        ; 7: addi   $t1, $zero, 2    # b = 2
[0x0040002       0x01095020  add $10, $8, $9       ; 9: add    $t2, $t0, $t1    # c = a + b
[0x0040003       0x11280003  beq $9, $8, 12        ; 11: seq   $t3, $t0, $t1    # d = 1 iff a ==
[0x00400034      0x340b0000  ori $11, $0, 0
[0x00400038]     0x10000002  beq $0, $0, 8
[0x0040003c]     0x340b0001  ori $11, $0, 1
[0x00400040]     0x11080003  beq $8, $8, 12        ; 12: seq   $t4, $t0, $t0    # e = 1 iff a ==

                  x00000000

                  x00000000

[                 0x7845202             0x636f2000

[0x00400014]     0x0c100009  jal 0x0040002           jal m
[0x00400024]     0x20080003  addi $8, $0, 3          addi
[0x00400028]     0x20090002  addi $9, $0, 2          addi
[0x0040002c]     0x01095020  add $10, $8, $
[0x00400030]     0x11280003  beq $9, $8, 12          seq
[0x00400034]     0x340b0000  ori $11, $0, 0
[0x00400038]     0x10000002  beq $0, $0, 8
```

Now we are done with the seq pseudo instrucion.

$t3 is still 00000000 because $t0 is not eaual to $t1.

To execute the seq pseudo instruction, four real machine instruction where used.

You can now procede and single step through the next seq instruction.



```
 PC       = 00400050   EPC      = 00000000   Cause
 Status   = 3000ff10   HI       = 00000000   LO
                                   General  Registe
R0  (r0) = 00000000  R8  (t0) = 00000003  R16 (s
R1  (at) = 00000000  R9  (t1) = 00000002  R17 (s
R2  (v0) = 00000000  R10 (t2) = 00000005  R18 (s2
R3  (v1) = 00000000                       R19 (s3) =              (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000001      (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400034]    0x340b0000  ori $11, $0, 0
[0x00400038]    0x10000002  beq $0, $0, 8
[0x0040003c]    0x340b0001  ori $11, $0, 1
[0x00400040]    0x11080003  beq $8, $8, 12                ; 12: seq    $t4, $t0, $t0    # e = 1 iff a ==
[0x00400044]    0x340c0000  ori $12, $0, 0
[0x00400048]    0x10000002  beq $0, $0, 8
[0x0040004c]    0x340c0001  ori $12, $0, 1
[0x00400050]    0x03e00008  jr $31                        ; 14: jr     $ra              # return to call
```

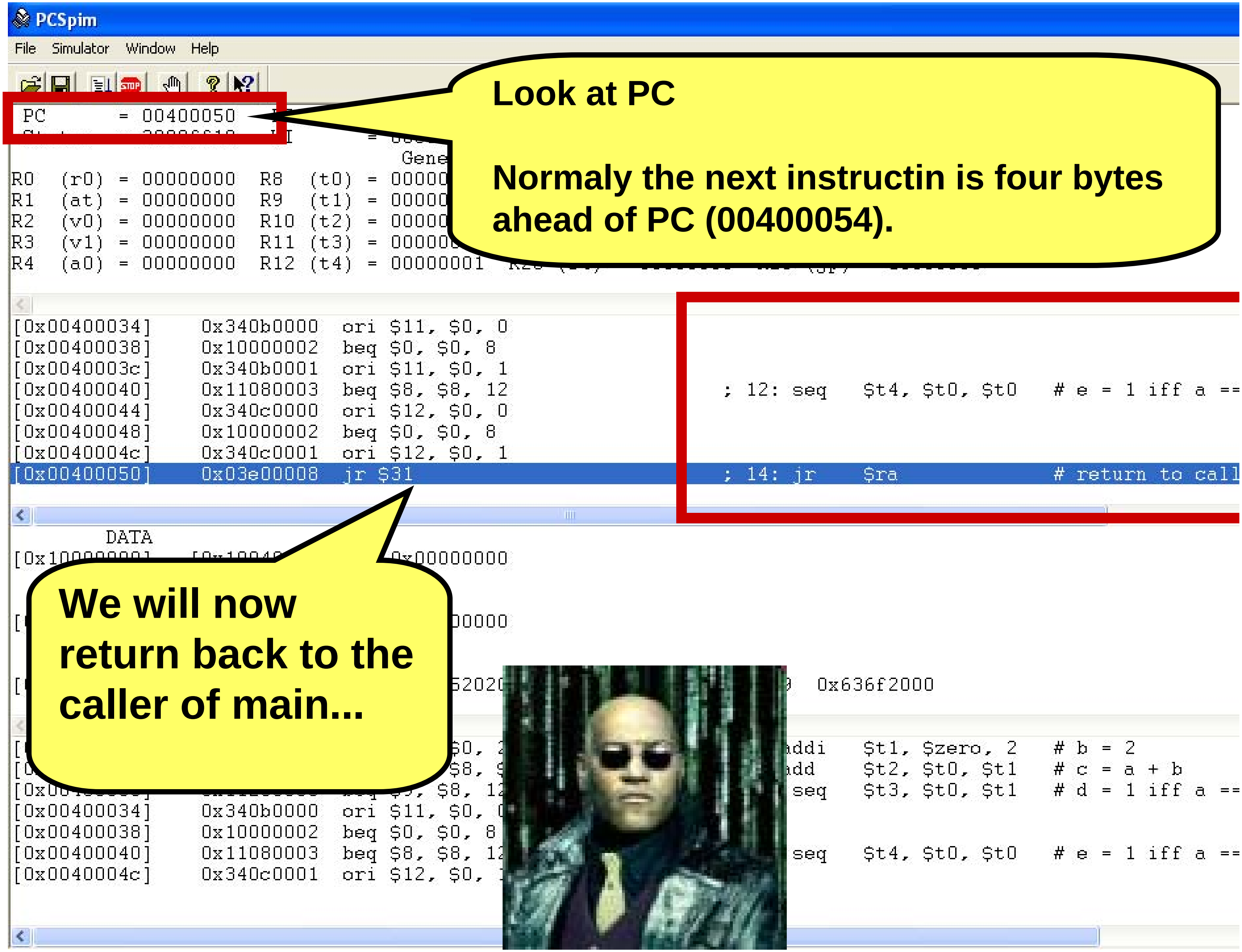
```
      DATA
[0x10000000]   [0x1004             0x00000000
                                   00000
                                   52020                          0x636f2000
```

```
                               $0, 2                     addi   $t1, $zero, 2      # b = 2
                               $8, 9                     add    $t2, $t0, $t1      # c = a + b
                               $8, 12                    seq    $t3, $t0, $t1      # d = 1 iff a ==
[0x00400034]    0x340b0000  ori $11, $0, 0
[0x00400038]    0x10000002  beq $0, $0, 8
[0x00400040]    0x11080003  beq $8, $8, 12               seq    $t4, $t0, $t0      # e = 1 iff a ==
[0x0040004c]    0x340c0001  ori $12, $0, 1
```

$t4 now changes from 00000000 to 00000001 because $t0 is equal to $t0.

Stand by!

The next instruction is important.

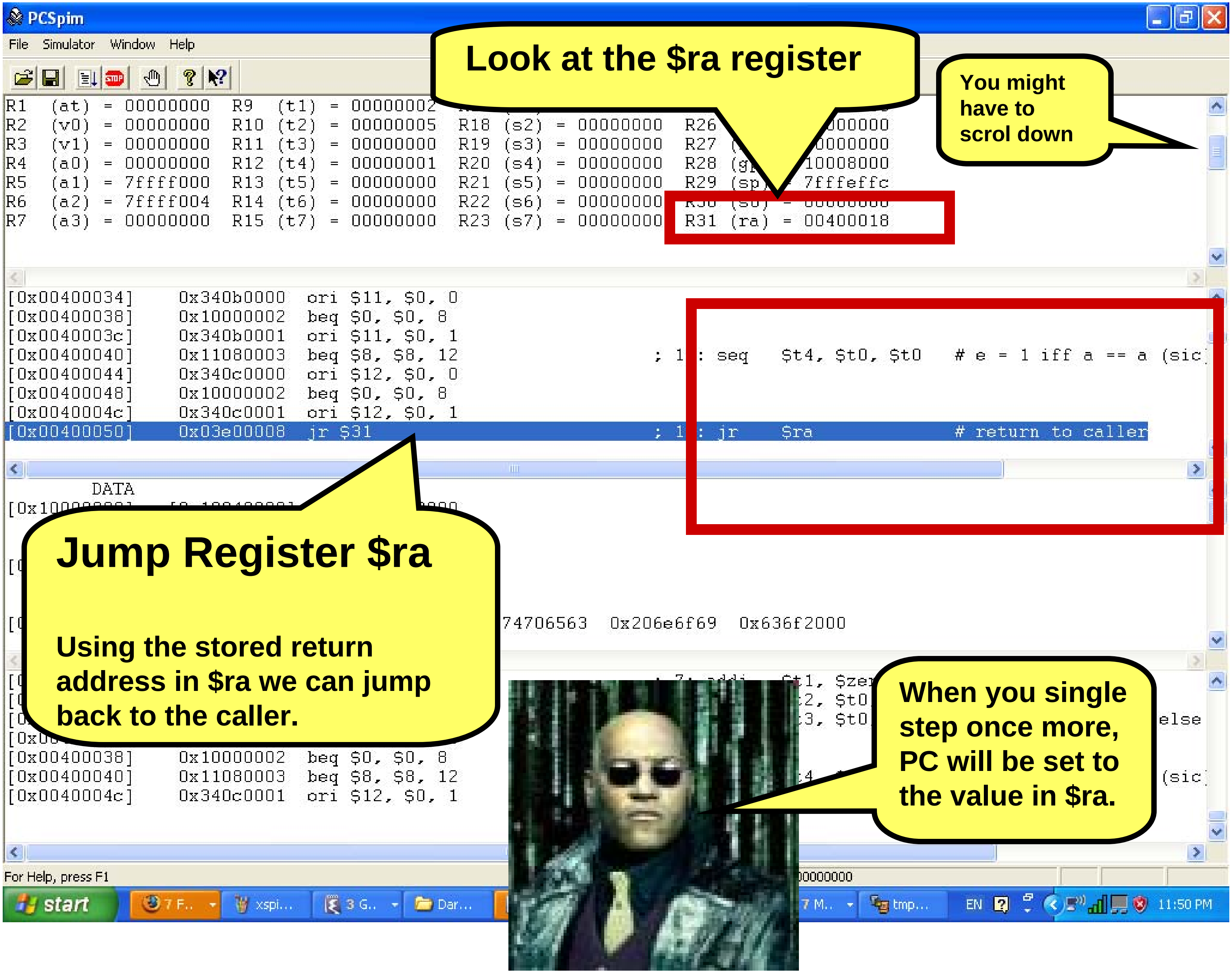


```
PC        = 00400050
                                   Gene
R0  (r0) = 00000000  R8  (t0) = 00000
R1  (at) = 00000000  R9  (t1) = 00000
R2  (v0) = 00000000  R10 (t2) = 00000
R3  (v1) = 00000000  R11 (t3) = 00000
R4  (a0) = 00000000  R12 (t4) = 00000001

[0x00400034]    0x340b0000  ori $11, $0, 0
[0x00400038]    0x10000002  beq $0, $0, 8
[0x0040003c]    0x340b0001  ori $11, $0, 1
[0x00400040]    0x11080003  beq $8, $8, 12                ; 12: seq    $t4, $t0, $t0    # e = 1 iff a ==
[0x00400044]    0x340c0000  ori $12, $0, 0
[0x00400048]    0x10000002  beq $0, $0, 8
[0x0040004c]    0x340c0001  ori $12, $0, 1
[0x00400050]    0x03e00008  jr $31                        ; 14: jr     $ra              # return to call

      DATA
[0x10000000]    [0x1004                0x00000000

                                       00000

                                       52020                      0x636f2000

                                       $0, 2                 addi   $t1, $zero, 2    # b = 2
                                       $8, $                 add    $t2, $t0, $t1    # c = a + b
                                       $8, 12                seq    $t3, $t0, $t1    # d = 1 iff a ==
[0x00400034]    0x340b0000  ori $11, $0, 0
[0x00400038]    0x10000002  beq $0, $0, 8
[0x00400040]    0x11080003  beq $8, $8, 12                 seq    $t4, $t0, $t0    # e = 1 iff a ==
[0x0040004c]    0x340c0001  ori $12, $0, 1
```

Look at PC

Normaly the next instructin is four bytes ahead of PC (00400054).

We will now return back to the caller of main...



```
R1  (at) = 00000000  R9  (t1) = 00000002
R2  (v0) = 00000000  R10 (t2) = 00000005  R18 (s2) = 00000000  R26
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27
R4  (a0) = 00000000  R12 (t4) = 00000001  R20 (s4) = 00000000  R28 (g   10008000
R5  (a1) = 7ffff000  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp)  7fffeffc
R6  (a2) = 7ffff004  R14 (t6) = 00000000  R22 (s6) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400018

[0x00400034]   0x340b0000  ori $11, $0, 0
[0x00400038]   0x10000002  beq $0, $0, 8
[0x0040003c]   0x340b0001  ori $11, $0, 1
[0x00400040]   0x11080003  beq $8, $8, 12          ; 1 : seq   $t4, $t0, $t0   # e = 1 iff a == a (sic
[0x00400044]   0x340c0000  ori $12, $0, 0
[0x00400048]   0x10000002  beq $0, $0, 8
[0x0040004c]   0x340c0001  ori $12, $0, 1
[0x00400050]   0x03e00008  jr $31                  ; 1 : jr    $ra             # return to caller

     DATA
[0x1

                                 74706563  0x206e6f69  0x636f2000

[0x00400038]   0x10000002  beq $0, $0, 8
[0x00400040]   0x11080003  beq $8, $8, 12
[0x0040004c]   0x340c0001  ori $12, $0, 1
```

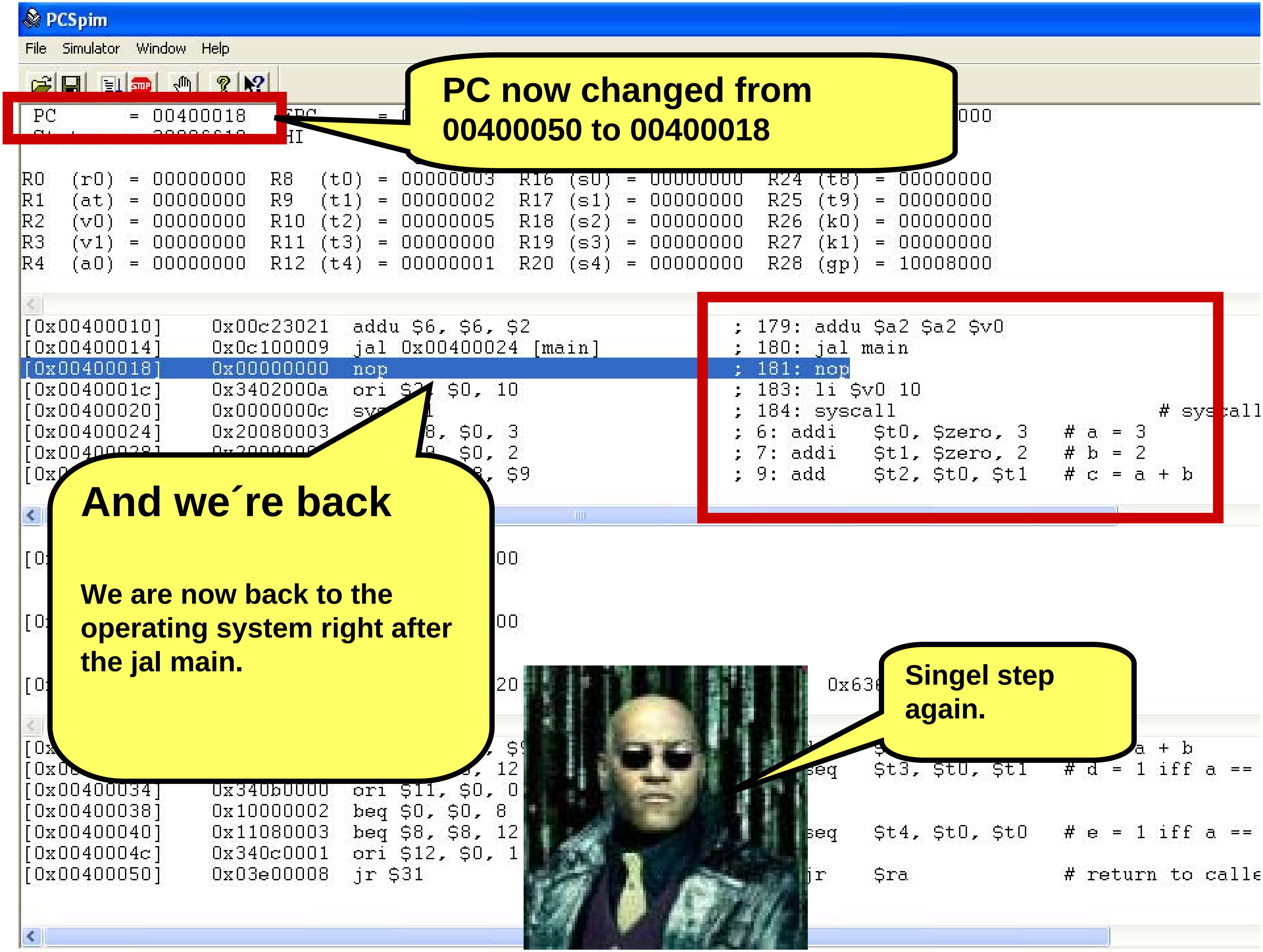
Look at the $ra register

You might have to scrol down

Jump Register $ra

Using the stored return address in $ra we can jump back to the caller.

When you single step once more, PC will be set to the value in $ra.



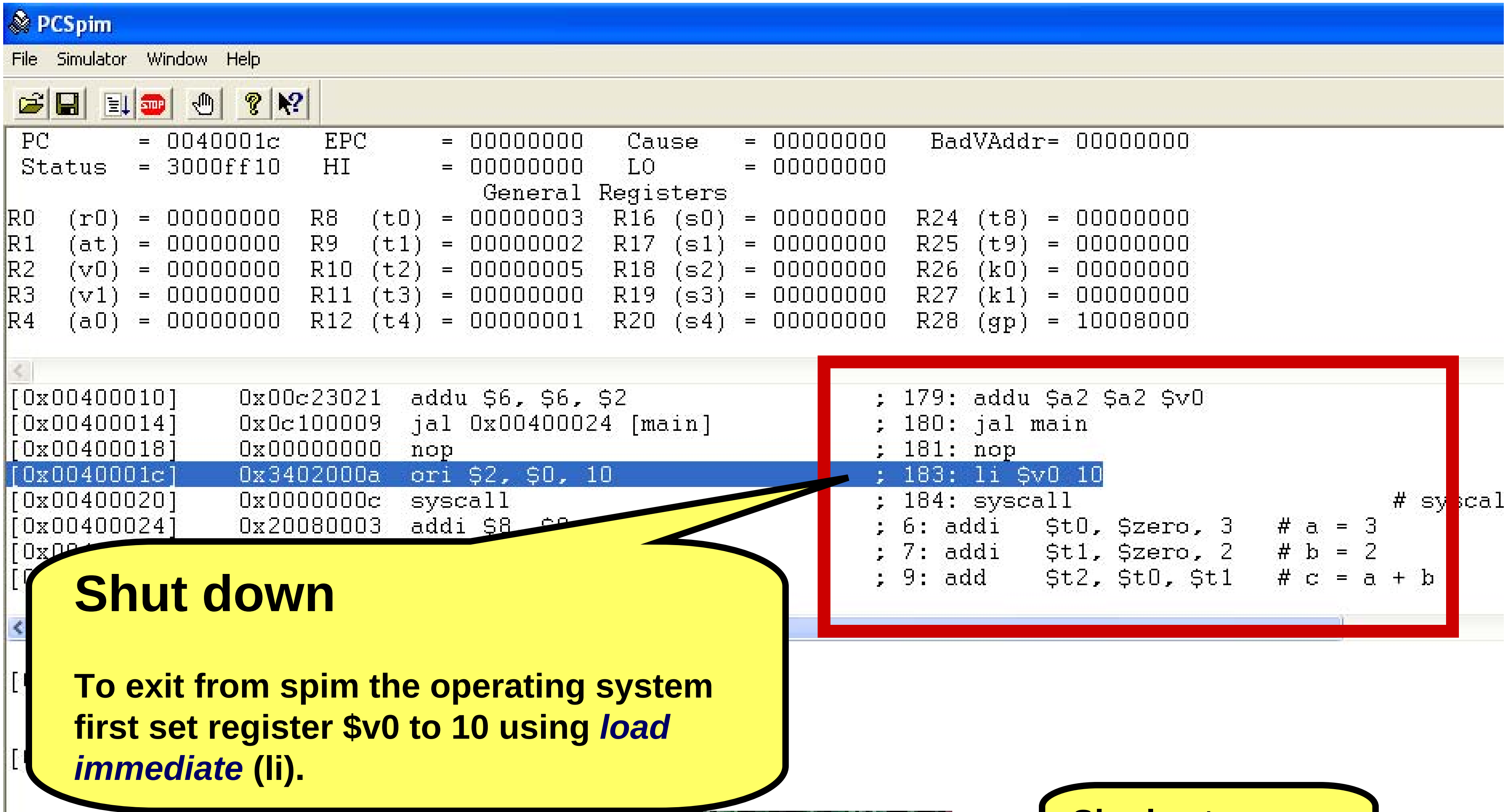
```
PC      = 00400018   EPC                                   000
                     HI

R0  (r0) = 00000000  R8  (t0) = 00000003  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000002  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000005  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000001  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400010]    0x00c23021  addu $6, $6, $2            ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]      ; 180: jal main
[0x00400018]    0x00000000  nop                        ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10             ; 183: li $v0 10
[0x00400020]    0x0000000c  sy         l               ; 184: syscall                    # sys  call
[0x00400024]    0x20080003       8, $0, 3              ; 6: addi   $t0, $zero, 3   # a = 3
[0x00400028]    0x2000000       9, $0, 2               ; 7: addi   $t1, $zero, 2   # b = 2
[0x                             8, $9                  ; 9: add    $t2, $t0, $t1   # c = a + b

[0                              00

[0                              00

[0                              20                    0x63

[0x                             , $9                                               a + b
[0x0                            , 12                  seq    $t3, $t0, $t1         # d = 1 iff a ==
[0x00400034]    0x340b0000  ori $11, $0, 0
[0x00400038]    0x10000002  beq $0, $0, 8
[0x00400040]    0x11080003  beq $8, $8, 12            seq    $t4, $t0, $t0         # e = 1 iff a ==
[0x0040004c]    0x340c0001  ori $12, $0, 1
[0x00400050]    0x03e00008  jr $31                    jr     $ra                   # return to calle
```

PC now changed from 00400050 to 00400018

And we´re back

We are now back to the operating system right after the jal main.

Singel step again.

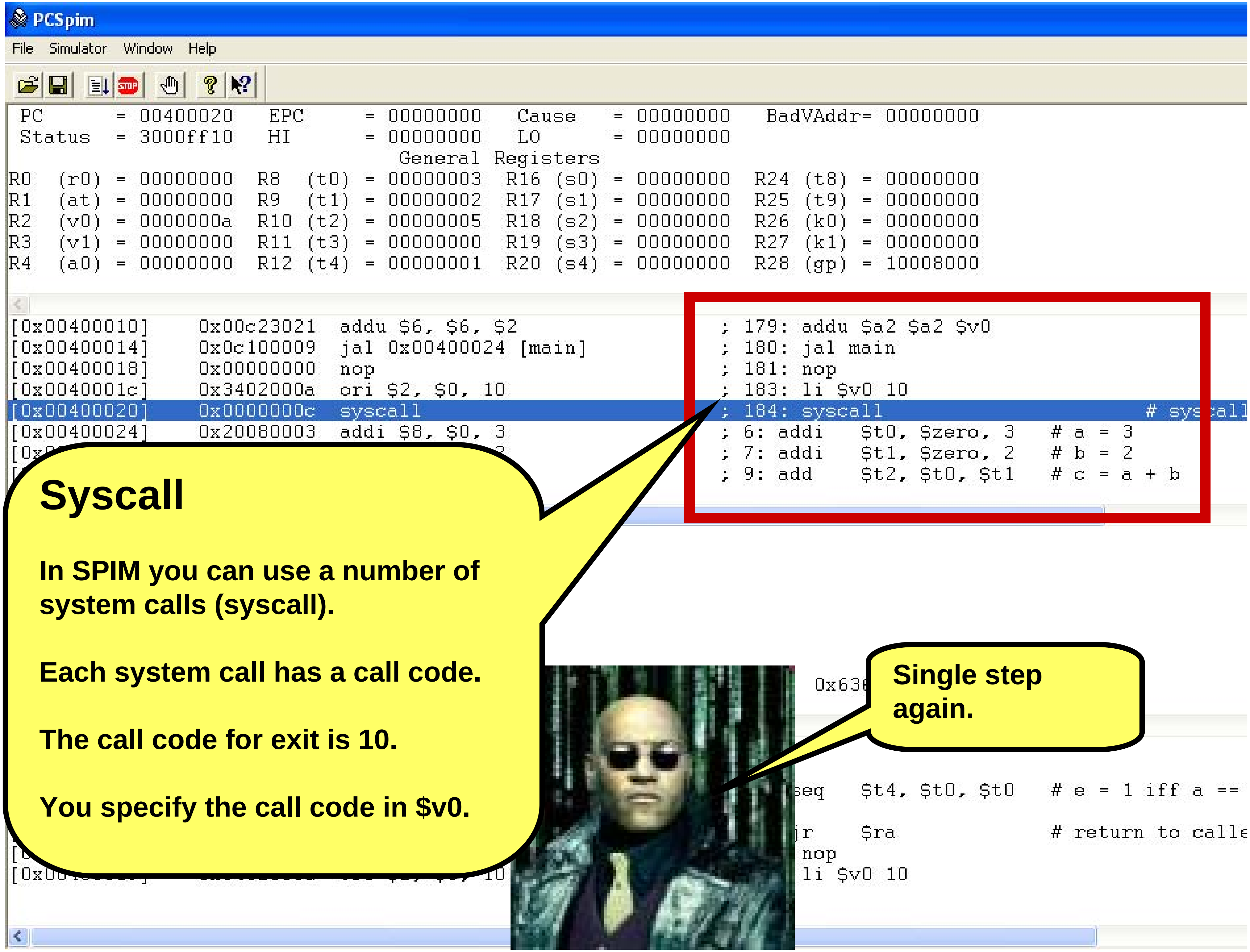


```
 PC       = 0040001c   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                General Registers
R0  (r0) = 00000000  R8  (t0) = 00000003  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000002  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000005  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000001  R20 (s4) = 00000000  R28 (gp) = 10008000

[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]        ; 180: jal main
[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10               ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                      ; 184: syscall                        # sy
[0x00400024]    0x20080003  addi $8, $0                  ; 6: addi   $t0, $zero, 3    # a = 3
                                                         ; 7: addi   $t1, $zero, 2    # b = 2
                                                         ; 9: add    $t2, $t0, $t1    # c = a + b

[0x90000000]                0x78452020                   0x63

[0x00400030]    0x11280003  beq $9, $8, 12                                            1 iff a ==
[0x00400034]    0x340b0000  ori $11, $0,
[0x00400038]    0x10000002  beq $0, $0, 8
[0x00400040]    0x11080003  beq $8, $8, 12                 seq    $t4, $t0, $t0    # e = 1 iff a ==
[0x0040004c]    0x340c0001  ori $12, $0,
[0x00400050]    0x03e00008  jr $31                         jr     $ra              # return to call
[0x00400018]    0x00000000  nop                          : nop
```

Shut down

To exit from spim the operating system first set register $v0 to 10 using *load immediate* (li).

Single step again.



```
PC       = 00400020   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                          General Registers
R0  (r0) = 00000000  R8  (t0) = 00000003  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000002  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 0000000a  R10 (t2) = 00000005  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000001  R20 (s4) = 00000000  R28 (gp) = 10008000
```

```
[0x00400010]    0x00c23021  addu $6, $6, $2             ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]       ; 180: jal main
[0x00400018]    0x00000000  nop                         ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10              ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                     ; 184: syscall                # syscall
[0x00400024]    0x20080003  addi $8, $0, 3              ; 6: addi   $t0, $zero, 3   # a = 3
                                                        ; 7: addi   $t1, $zero, 2   # b = 2
                                                        ; 9: add    $t2, $t0, $t1   # c = a + b
```

```
                                                        seq    $t4, $t0, $t0   # e = 1 iff a ==
                                                        jr     $ra             # return to calle
                                                        nop
[0x00                       ori $2, $0, 10              li $v0 10
```

Syscall

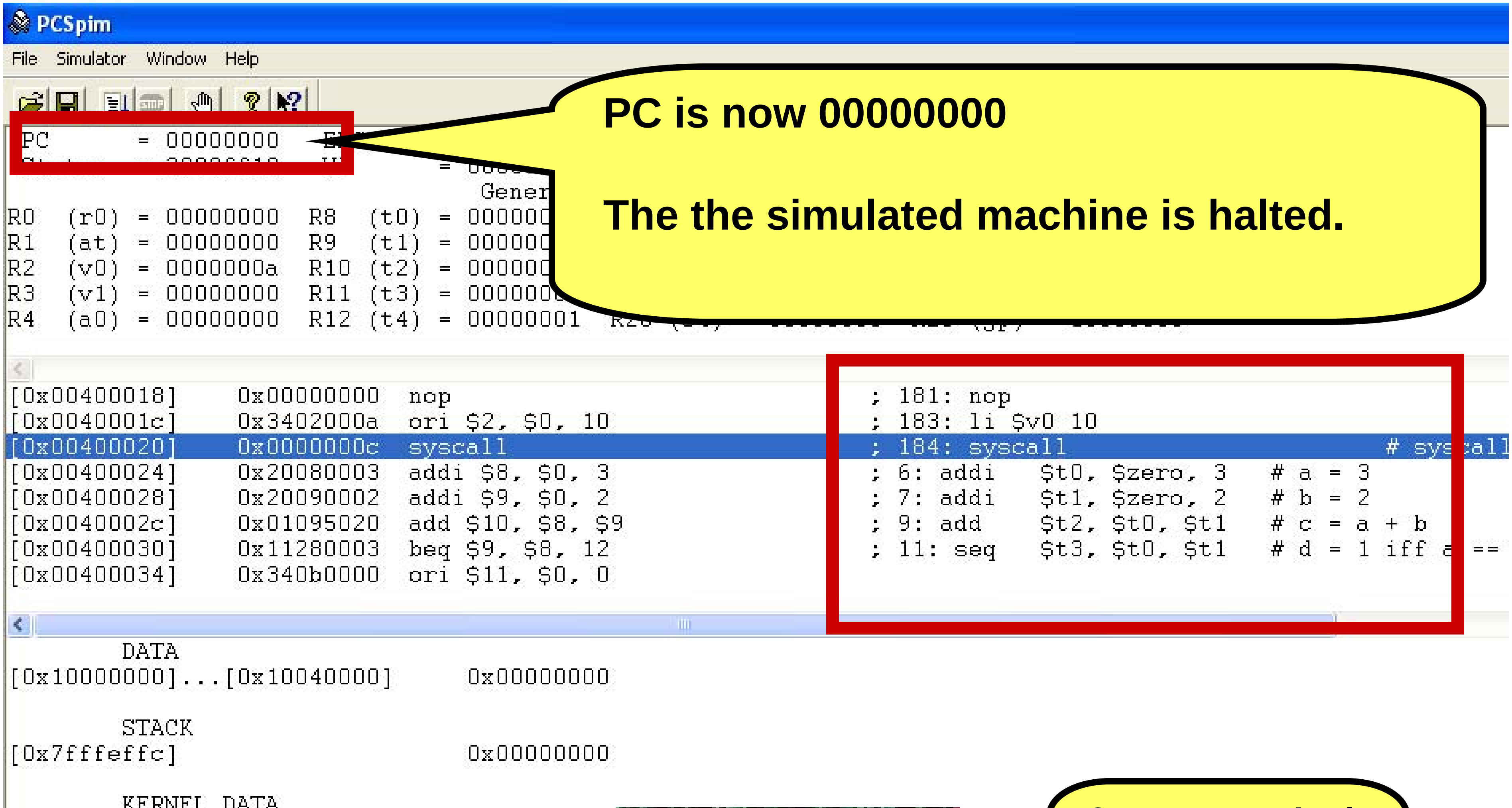
In SPIM you can use a number of system calls (syscall).

Each system call has a call code.

The call code for exit is 10.

You specify the call code in $v0.

Single step again.

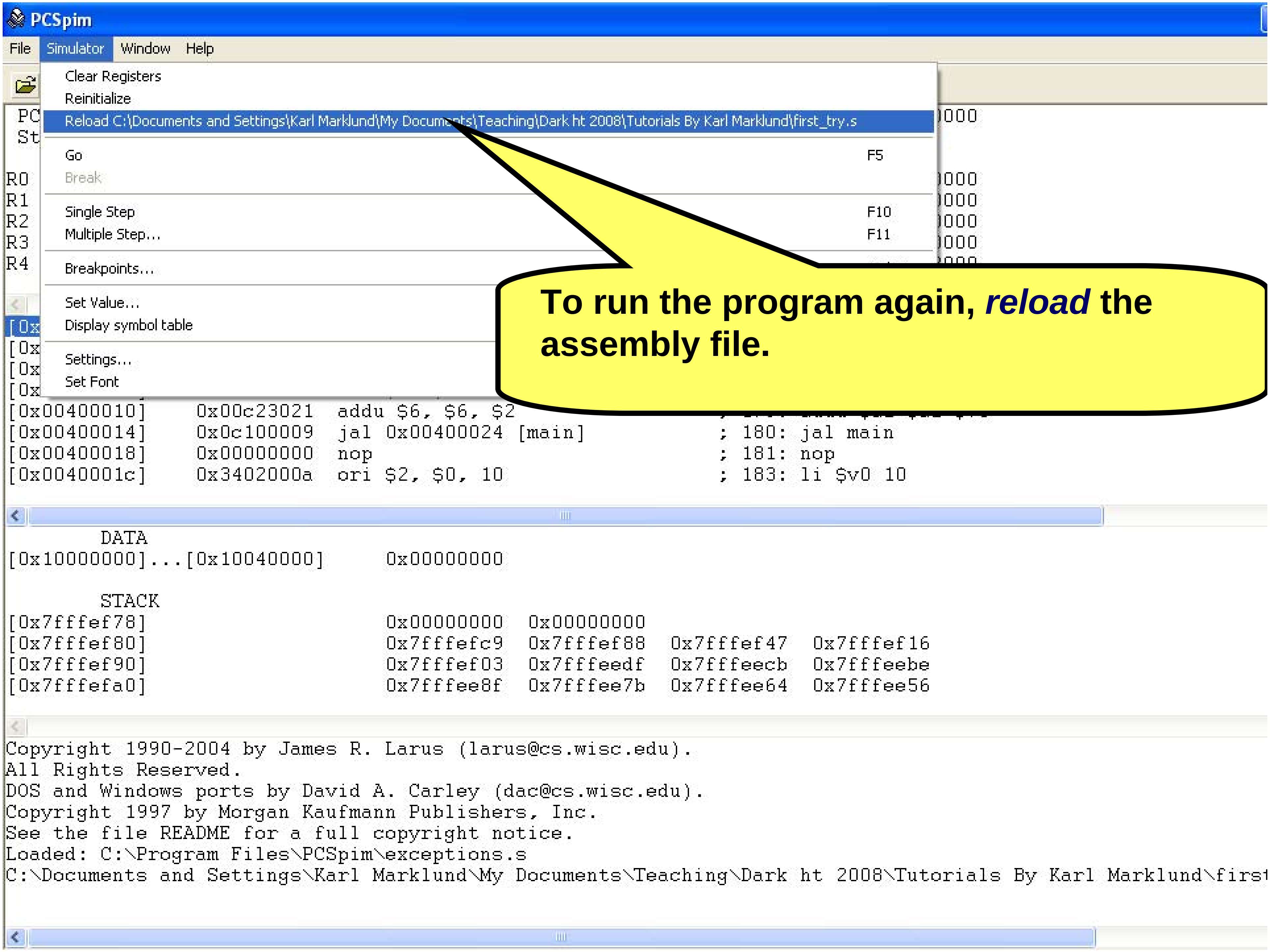


```
PC       = 00000000   EP
                      = 0000
                        Gener
R0  (r0) = 00000000  R8  (t0) = 000000
R1  (at) = 00000000  R9  (t1) = 000000
R2  (v0) = 0000000a  R10 (t2) = 000000
R3  (v1) = 00000000  R11 (t3) = 000000
R4  (a0) = 00000000  R12 (t4) = 00000001

[0x00400018]    0x00000000  nop                        ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10             ; 183: li $v0 10
[0x00400020]    0x0000000c  syscall                    ; 184: syscall                  # syscall
[0x00400024]    0x20080003  addi $8, $0, 3             ; 6: addi   $t0, $zero, 3   # a = 3
[0x00400028]    0x20090002  addi $9, $0, 2             ; 7: addi   $t1, $zero, 2   # b = 2
[0x0040002c]    0x01095020  add $10, $8, $9            ; 9: add    $t2, $t0, $t1   # c = a + b
[0x00400030]    0x11280003  beq $9, $8, 12             ; 11: seq   $t3, $t0, $t1   # d = 1 iff a ==
[0x00400034]    0x340b0000  ori $11, $0, 0

        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000

        KERNEL DATA
[0x90000000]                    0x78452020          0x63

[0x00400038]    0x10000002  beq $0, $0, 8
[0x00400040]    0x11080003  beq $8, $8, 12                                    f a ==
[0x0040004c]    0x340c0001  ori $12, $0, 1
[0x00400050]    0x03e00008  jr $31          jr    $ra                         to calle
[0x00400018]    0x00000000  nop             nop
[0x0040001c]    0x3402000a  ori $2, $0, 10  li $v0 10
[0x00400020]    0x0000000c  syscall         syscall                           # syscall
```

PC is now 00000000

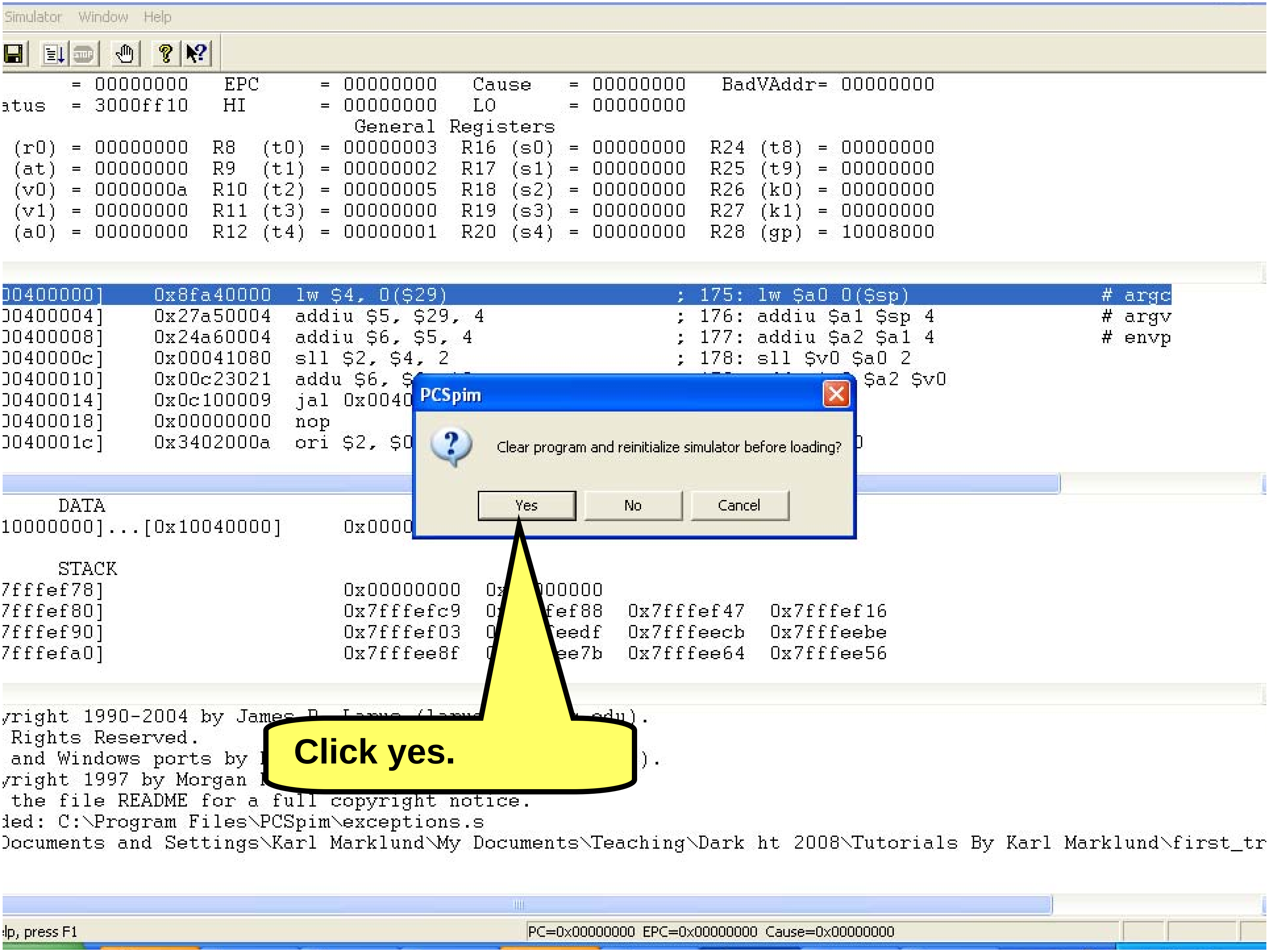
The the simulated machine is halted.

If you try to single step again, nothing will happen.



To run the program again, *reload* the assembly file.

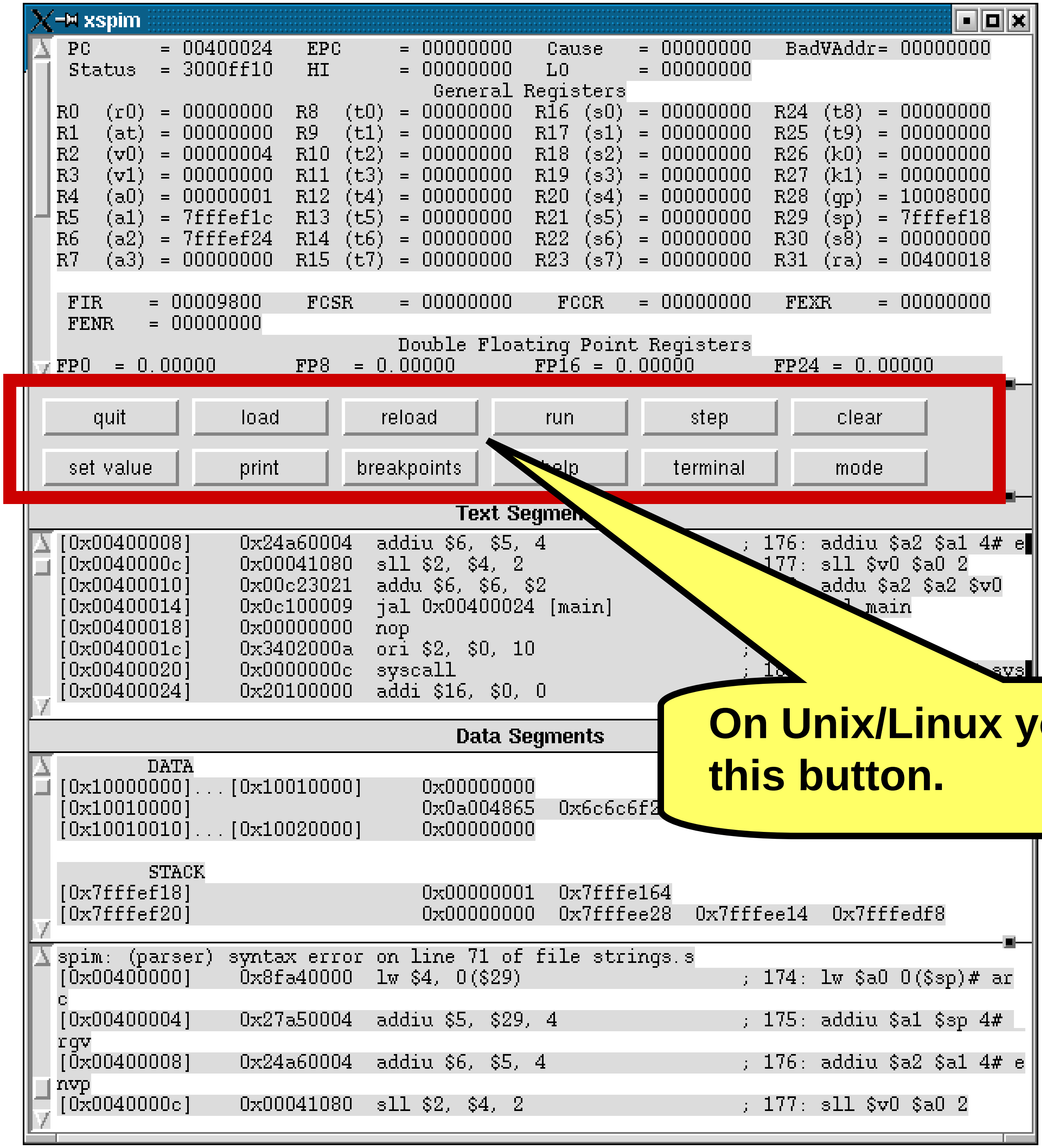
```
[0x00400010]    0x00c23021  addu $6, $6, $2
[0x00400014]    0x0c100009  jal 0x00400024 [main]          ; 180: jal main
[0x00400018]    0x00000000  nop                            ; 181: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                 ; 183: li $v0 10

        DATA
[0x10000000]...[0x10040000]    0x00000000

        STACK
[0x7fffef78]                   0x00000000  0x00000000
[0x7fffef80]                   0x7fffefc9  0x7fffef88  0x7fffef47  0x7fffef16
[0x7fffef90]                   0x7fffef03  0x7fffeedf  0x7fffeecb  0x7fffeebe
[0x7fffefa0]                   0x7fffee8f  0x7fffee7b  0x7fffee64  0x7fffee56

Copyright 1990-2004 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:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first
```



```
       = 00000000   EPC     = 00000000   Cause    = 00000000   BadVAddr= 00000000
atus   = 3000ff10   HI      = 00000000   LO       = 00000000
                          General Registers
 (r0) = 00000000  R8  (t0) = 00000003  R16 (s0) = 00000000  R24 (t8) = 00000000
 (at) = 00000000  R9  (t1) = 00000002  R17 (s1) = 00000000  R25 (t9) = 00000000
 (v0) = 0000000a  R10 (t2) = 00000005  R18 (s2) = 00000000  R26 (k0) = 00000000
 (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
 (a0) = 00000000  R12 (t4) = 00000001  R20 (s4) = 00000000  R28 (gp) = 10008000

00400000]    0x8fa40000  lw $4, 0($29)             ; 175: lw $a0 0($sp)        # argc
00400004]    0x27a50004  addiu $5, $29, 4          ; 176: addiu $a1 $sp 4      # argv
00400008]    0x24a60004  addiu $6, $5, 4           ; 177: addiu $a2 $a1 4      # envp
0040000c]    0x00041080  sll $2, $4, 2             ; 178: sll $v0 $a0 2
00400010]    0x00c23021  addu $6, $                          $a2 $v0
00400014]    0x0c100009  jal 0x0040
00400018]    0x00000000  nop
0040001c]    0x3402000a  ori $2, $0                          0

    DATA
10000000]...[0x10040000]    0x0000

    STACK
7fffef78]                   0x00000000  0x     00000
7fffef80]                   0x7fffefc9  0     fef88  0x7fffef47  0x7fffef16
7fffef90]                   0x7fffef03  0     eedf   0x7fffeecb  0x7fffeebe
7fffefa0]                   0x7fffee8f  0     ee7b   0x7fffee64  0x7fffee56

yright 1990-2004 by James                   edu).
 Rights Reserved.
 and Windows ports by                       ).
yright 1997 by Morgan
 the file README for a full copyright notice.
ded: C:\Program Files\PCSpim\exceptions.s
Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_tr

lp, press F1                 PC=0x00000000  EPC=0x00000000  Cause=0x00000000
```

Clear program and reinitialize simulator before loading?

Click yes.



```
 PC       = 00400024   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                               General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000004  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000001  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 7fffef1c  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffef18
R6  (a2) = 7fffef24  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400018

 FIR     = 00009800    FCSR     = 00000000    FCCR     = 00000000   FEXR     = 00000000
 FENR    = 00000000
                            Double Floating Point Registers
FP0   = 0.00000        FP8   = 0.00000       FP16  = 0.00000       FP24  = 0.00000
```

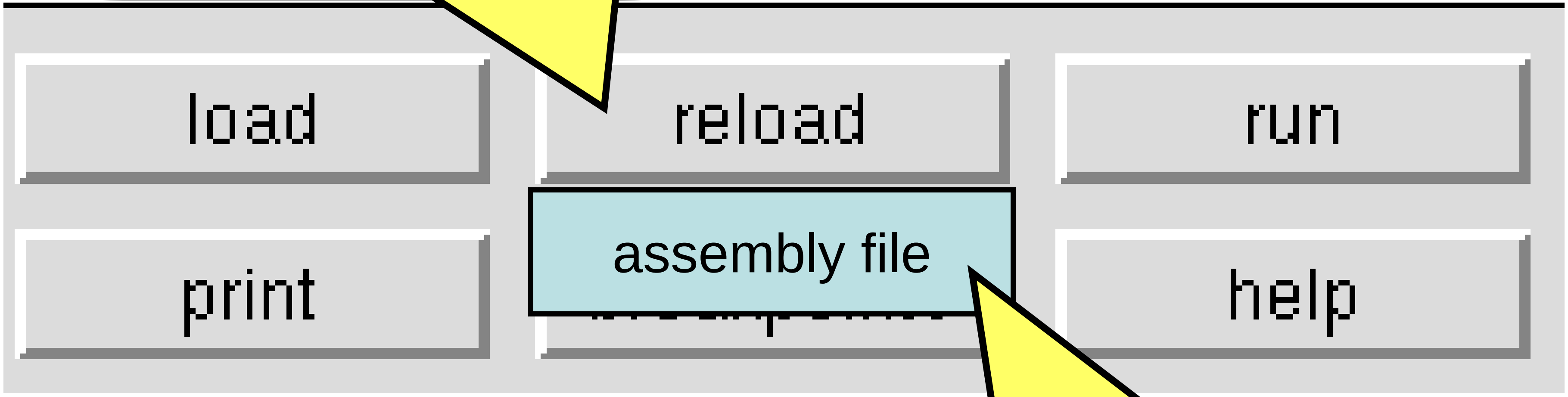
```
[0x00400008]    0x24a60004  addiu $6, $5, 4                      ; 176: addiu $a2 $a1 4# e
[0x0040000c]    0x00041080  sll $2, $4, 2                          177: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2                             addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]                        main
[0x00400018]    0x00000000  nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                       ;
[0x00400020]    0x0000000c  syscall                              ; 18                  sys
[0x00400024]    0x20100000  addi $16, $0, 0
```

```
        DATA
[0x10000000]...[0x10010000]      0x00000000
[0x10010000]                     0x0a004865  0x6c6c6f2
[0x10010010]...[0x10020000]      0x00000000

        STACK
[0x7fffef18]                     0x00000001  0x7fffe164
[0x7fffef20]                     0x00000000  0x7fffee28  0x7fffee14  0x7fffedf8
```

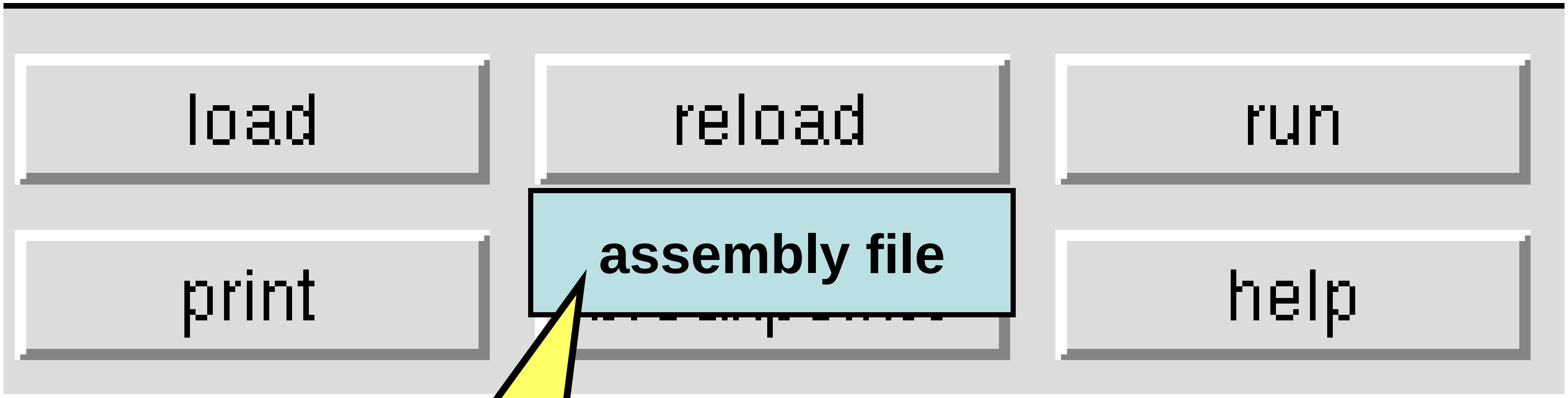
```
spim: (parser) syntax error on line 71 of file strings.s
[0x00400000]    0x8fa40000  lw $4, 0($29)                        ; 174: lw $a0 0($sp)# ar
c
[0x00400004]    0x27a50004  addiu $5, $29, 4                     ; 175: addiu $a1 $sp 4#
rgv
[0x00400008]    0x24a60004  addiu $6, $5, 4                      ; 176: addiu $a2 $a1 4# e
nvp
[0x0040000c]    0x00041080  sll $2, $4, 2                        ; 177: sll $v0 $a0 2
```

On Unix/Linux you reload using this button.

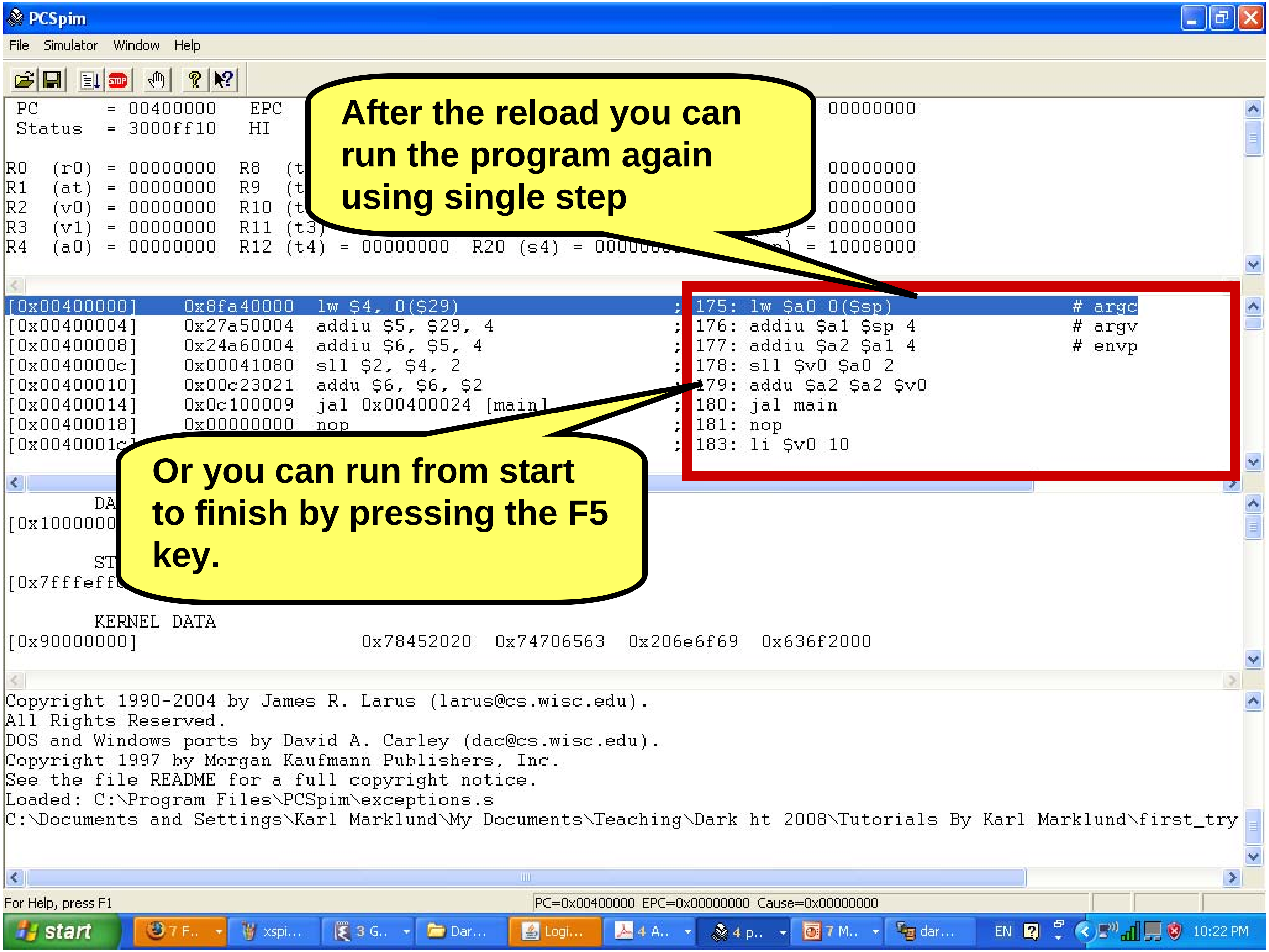
Clicking the reload button...



... makes the following option vissible.



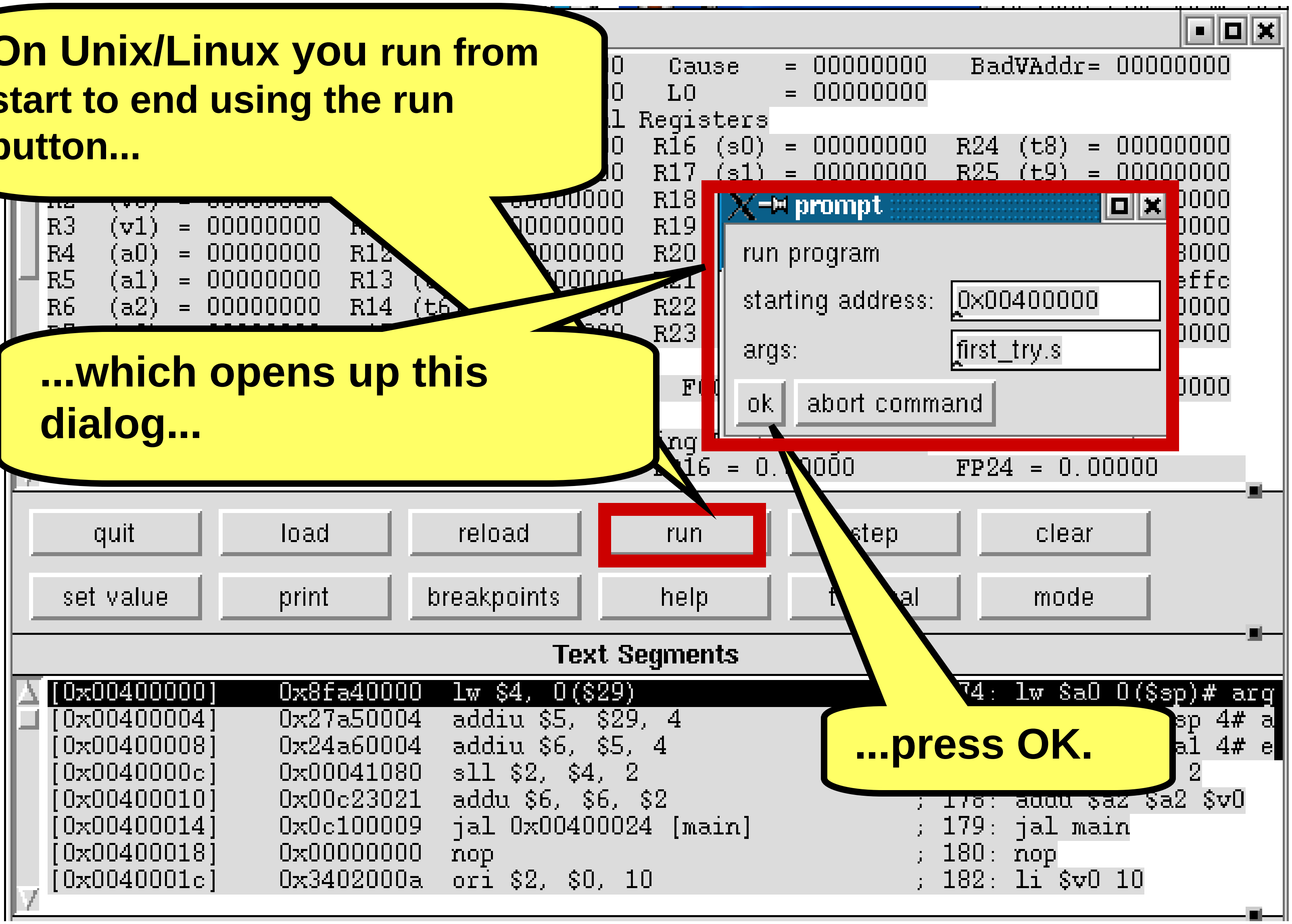
To complete the reload you must point the mouse here.



After the reload you can run the program again using single step

Or you can run from start to finish by pressing the F5 key.

```
 PC      = 00400000   EPC                                          00000000
 Status  = 3000ff10   HI

R0  (r0) = 00000000  R8  (t                                        00000000
R1  (at) = 00000000  R9  (t                                        00000000
R2  (v0) = 00000000  R10 (t                                        00000000
R3  (v1) = 00000000  R11 (t3)                                    = 00000000
R4  (a0) = 00000000  R12 (t4) = 00000000  R20 (s4) = 0000000     = 10008000

[0x00400000]    0x8fa40000  lw $4, 0($29)                ; 175: lw $a0 0($sp)            # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4             ; 176: addiu $a1 $sp 4          # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4              ; 177: addiu $a2 $a1 4          # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 178: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 179: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]        ; 180: jal main
[0x00400018]    0x00000000  nop                          ; 181: nop
[0x0040001c]                                             ; 183: li $v0 10

        DA
[0x1000000

        ST
[0x7fffeff

        KERNEL DATA
[0x90000000]                     0x78452020  0x74706563  0x206e6f69  0x636f2000

Copyright 1990-2004 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:\Program Files\PCSpim\exceptions.s
C:\Documents and Settings\Karl Marklund\My Documents\Teaching\Dark ht 2008\Tutorials By Karl Marklund\first_try
```



On Unix/Linux you run from start to end using the run button...

...which opens up this dialog...

...press OK.

```
Cause    = 00000000   BadVAddr= 00000000
LO       = 00000000
l Registers
R16 (s0) = 00000000   R24 (t8) = 00000000
R17 (s1) = 00000000   R25 (t9) = 00000000
R3  (v1) = 00000000
R4  (a0) = 00000000   R12
R5  (a1) = 00000000   R13
R6  (a2) = 00000000   R14 (t6
```

prompt

run program

starting address: 0x00400000

args: first_try.s

ok | abort command

```
[0x00400000]    0x8fa40000  lw $4, 0($29)                  74: lw $a0 0($sp)# arg
[0x00400004]    0x27a50004  addiu $5, $29, 4                  sp 4# a
[0x00400008]    0x24a60004  addiu $6, $5, 4                   a1 4# e
[0x0040000c]    0x00041080  sll $2, $4, 2                        2
[0x00400010]    0x00c23021  addu $6, $6, $2                ; 178: addu $a2 $a2 $v0
[0x00400014]    0x0c100009  jal 0x00400024 [main]          ; 179: jal main
[0x00400018]    0x00000000  nop                            ; 180: nop
[0x0040001c]    0x3402000a  ori $2, $0, 10                 ; 182: li $v0 10
```

You are now done with part one of your MIPS assembly programming training.