CSE 410
Computer Systems

Hal Perkins

Spring 2010

Lecture 4 – SPIM: A MIPS Simulator

Reading and References

- Primary reference: textbook Appendix B
  - (Appendix A in older editions of the textbook)

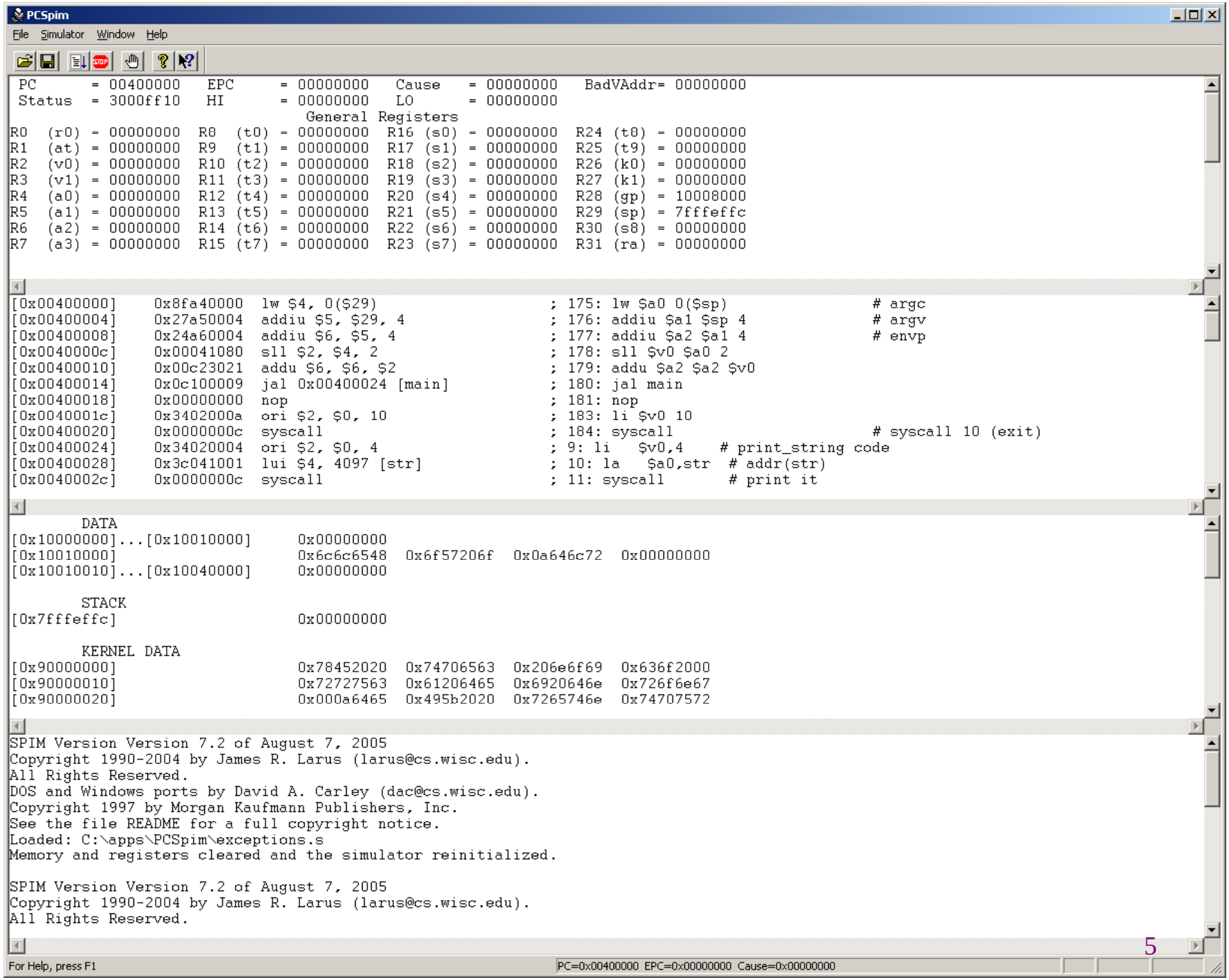
- Also, see the resources section on the SPIM web page for additional documentation on SPIM
  - Look for Getting Started guides
  - http://www.cs.wisc.edu/~larus/spim.html
  - (shortcut: google “spim” – it should be the first hit)
  - Download SPIM for your machine from here

SPIM simulator

- SPIM lets you write MIPS assembly language code and run it on a PC, linux, or mac
- PCSpim should be installed on the machines in the A&S computer lab
- You can download versions for Windows and all varieties of *nix (including MacOS X) from the web site
  - Trade hints on the discussion list if you have troubles building/installing it
  - Known bug: won't build on OS X 10.6; binaries built on 10.5 intel should install ok on 10.6.

Spim display

- Register panel
  - register names and numbers
- Text segment (code) panel
  - note jump and link to “main” at [0x00400014]
  - your code defines the label “main”
- Data and Stack segment panel
- Message panel


PCSpim
File Simulator Window Help
PC = 00400000 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
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
[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 ; 183: li $v0 10
[0x00400020] 0x0000000c syscall ; 184: syscall # syscall 10 (exit)
[0x00400024] 0x34020004 ori $2, $0, 4 ; 9: li $v0,4 # print_string code
[0x00400028] 0x3c041001 lui $4, 4097 [str] ; 10: la $a0,str # addr(str)
[0x0040002c] 0x0000000c syscall ; 11: syscall # print it
DATA
[0x10000000]...[0x10010000] 0x00000000
[0x10010000] 0x6c6c6548 0x6f57206f 0x0a646c72 0x00000000
[0x10010010]...[0x10040000] 0x00000000
STACK
[0x7fffeffc] 0x00000000
KERNEL DATA
[0x90000000] 0x78452020 0x74706563 0x206e6f69 0x636f2000
[0x90000010] 0x72727563 0x61206465 0x6920646e 0x726f6e67
[0x90000020] 0x000a6465 0x495b2020 0x7265746e 0x74707572
SPIM Version Version 7.2 of August 7, 2005
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:\apps\PCSpim\exceptions.s
Memory and registers cleared and the simulator reinitialized.
SPIM Version Version 7.2 of August 7, 2005
Copyright 1990-2004 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
For Help, press F1
PC=0x00400000 EPC=0x00000000 Cause=0x00000000

Editing SPIM programs

- You can use any (plain) text editor you like to write the source code
  - *Not* Microsoft Word
- Textpad, notepad++, etc., on PC's
- jEdit also provides a MIPS assembly highlighter
- emacs can do anything including asm – but has a pretty steep learning curve

- A few examples follow on the next slides

hello.s

```
# This MIPS program uses a system call to print a string
        .data
str:
        .asciiz "Hello World\n"
        .text
main:
        li $v0,4        # print_string code
        la $a0,str      # addr(str)
        syscall         # print it
        jr $ra          # return
```

Assembly language basics

- SPIM reads a program written in MIPS assembly language, translates it to machine code (001011001100…), then executes it
- Programs have two sections
  - .data – storage for constants and variables
  - .text – program code

  Sections can be repeated (alternated) as often as needed
- Code must contain a label `main:`
  - Execution begins here; SPIM "calls" main
  - main should return when done (`jr $ra`)
- Much, much more in the book and upcoming lectures on function calling conventions

add.s

```
# load two numbers from memory into registers, add them,
# and store their sum
        .data
one:    .word   1
two:    .word   2
sum:    .word   -1
        .text
main:   lw      $t0,one
        lw      $t1,two
        add     $t2,$t0,$t1
        sw      $t2,sum
        jr      $ra # return
```

addi.s

```
# load two numbers into registers and add them.
# this time the numbers are loaded directly
# from the instructions, not from memory
        .text
main:   li      $t0,1
        li      $t1,2
        add     $t2,$t0,$t1
        jr      $ra # return
```