

Writing and Executing MIPS Programs

- Today we study the mechanics of writing a MIPS program.
- **As mentioned earlier, assembly language programming is a “no frills” activity -- sophisticated compiler functions do not exist.**
- Even functions such as calculating the numbers in an array, changing or swapping array variables, etc. must be specifically programmed in SPIM.
- **“Loops,” “function calls,” and activation of subroutines (all of which we have yet to study), in general, must be done manually.**
- Using the stack is also a manual operation; there are no “push” or “pop” assembly language instructions.
- **In general, any operation which requires controlling or modifying the program flow is much more laborious than in higher-level languages, but the visibility into computer operation is better.**

Program File Linking and Loading

- All link and load functions occur within the SPIM environment as a part of the assembly process.
- The file is simply (1) opened by the SPIM assembler, (2) **automatically assembled**, (3) **linked**, and (4) **readied for execution**.
- The user simply has to initiate execution, and the program will go forward.
- Execution can be single-step if desired (using F10).
- A system console is available for seeing appropriate program “output.”
- Debug is simple; for program errors (lines that will not properly assemble), SPIM merely stops with a note indicating the line that failed (more on this below).

Writing the MIPS Assembly Language Program

- **SPIM does not supply an “environment” for writing and debugging MIPS assembly language programs.**
- **You may write a MIPS program using any simple text editor.**
- **Comments below refer only to writing and executing programs on the PC (using PCSPIM).**
 - **Do not use a word processor, which sticks in extraneous formatting characters that can confuse the SPIM assembler. It is possible to use Word TM and store the result as a “txt” rather than “doc,” but there are possible problems opening and using the data.**
 - **Any simple text editor will do -- I prefer Windows Notepad, which uses simple textual representation with only the actual characters that are typed (WYSIWYG – “What you see is what you get”).**
 - **Make sure to use the exact form of the instruction shown in your text (either P&H or Pervin) -- the assembler is sensitive to upper case.**

Review: SPIM Program Format

- **Comments – May make as many as necessary:**
 - Always preceded by **#** symbol (signals assembler that “this is a comment, and not a MIPS instruction”).
 - **Comments do NOT “wrap.”** If a comment takes more than a line, start a new line with **#** and continue your comment.
- **Text section:**
 - Headed by **.text** declaration (required).
 - First text line always labeled “**main:**” (note colon!).
 - **This section contains the program (instructions).**
 - **Only one instruction to a line** (once past the instruction, you may add a comment, simply preceding it with a **#** sign).
 - **Normally, text section ends with “syscall 10.”**

Format of PC SPIM Program (2)

- **Data segment:**
 - Always headed by “.data” declaration.
 - Every data argument should be identified with a label. *
 - Location is optional -- may be placed in front of text section if preferred.
 - **Final comments: Optional**
- * A label for every piece of data is not absolutely required, but it is generally unwise not to label each data entry.

A Simple PC SPIM Example from Lecture 11

```
#      Lecture 11 Exercise 2: Prints out "hello world."  
  
      .text  
  
main:  nop  
      la $a0,str      # put string address into $a0  
      li $v0,4        # system call to print  
      syscall        # outputting the ASCII string  
      li $v0,10  
      syscall        # exit (pass control back to operating system)  
  
      .data  
  
str:   .asciiz "hello world\n"
```

Running the Example Program

- **Sequence of demonstration:**
 - Start PC SPIM
 - Load Lecture 11 Exercise 2
 - Run Lecture 11 Exercise 2
 - Examine console and various readout windows
 - Single-step-run program
 - Introduce error into program listing
 - Re-load program
 - Note debug errors; repair, reload, and run

PC SPIM Simulation “Windows”

- In our earlier PCSPIM demonstrations, you will have noticed that in addition to the text window, PC SPIM has three additional views, or windows:
 - **The registers window:**
 - Gives real-time register contents during step execution.
 - Includes program counter and exception register view.
 - **The data segment window:**
 - Shows all relevant memory locations containing data.
 - Also by-instruction status as program is single-stepped.
 - **Message window:**
 - Shows relevant status messages, program progress.
 - May be worthwhile occasionally, but I do not find it very useful.

Another Example

- **Example sequence:**
 - **Restart PC SPIM**
 - **Load Lecture 12 Demo 1**
 - **Start Lecture 12 Demo 1**
 - **Do console input**
 - **Complete program**
 - **View answer to input on Console**
 - **Repeat in single step mode**
- **First a note about syscall 5, which we have not used before.**

Syscall 5

- **System call 5 allows input of numerical data from the keyboard while a program is running.**
- **Syscall 5 is a little unusual in that it requires the use of register \$v0 twice. In syscall 5 (as for all system calls), the number 5 is loaded into \$v0 before executing the “syscall” instruction.**
- **However, after a number is input from the keyboard (input completion is signified by pressing the “Enter” key), this number is stored in \$v0.**

Syscall 5 (2)

- Since $\$v0$ is used for all system calls, **the programmer needs to immediately provide for the input data stored in $\$v0$ to be moved to memory or at least shifted to another register.**
- Another characteristic of syscall 5 is that it will only input enough information to fill $\$v0$ (i.e., $0x\text{ ffff ffff}$). The rest is lost (most significant digit first!). Also, MSB is still the sign bit!
- Thus the biggest positive number that can be input on syscall 5 is $\sim 2,145,000,000$. Keep that in mind when programming numerical inputs from the keyboard.
- The syscall 5 form would be:
 - `li $v0,5`
 - `syscall`

At the end of this system call, the number input from the keyboard is in $\$v0$.
- **Now to the demo (next slide).**

Program Lecture 12 Demo 1

This program converts temperature in Celsius to Fahrenheit, printing the results.

v0 - reads in Celsius temperature from the keyboard

t0 - holds Fahrenheit result

a0 - points to output strings

.text

```
main: la $a0,prompt      # print prompt on terminal
      li $v0,4
      syscall
      li $v0,5           # syscall 5 reads an integer
      syscall
      mul $t0,$v0,9      # to convert, multiply by 9
      div $t0,$t0,5      # then divide by 5, and
      add $t0,$t0,32     # add 32 to complete the conversion
      la $a0,ans1        # print string before result
      li $v0,4
      syscall
      move $a0,$t0       # print result
      li $v0,1
      syscall
      la $a0,endl        # system call to print
      li $v0,4           # out a new line
      syscall
      li $v0,10
      syscall            # end program
```

.data

```
prompt: .asciiz "Enter Temperature (Celsius): "
ans1:   .asciiz "The temperature in Fahrenheit is "
endl:   .asciiz "\n"
##
```

Exercise 1

- **Write a program to input a number from the console, square it, and put the result in registers \$t0 and \$s0. Then output the result using the correct system call and stop the program using the proper syscall as well.**

Obtaining SPIM from the University of Wisconsin

- The most up-to-date version of the SPIM simulator currently available (to the instructor's knowledge) may be downloaded from the home page of James Larus, formerly of the University of Wisconsin at Madison (it is "freeware").
- Go to his website: <http://www.cs.wisc.edu/~larus/spim.html>
- Page down until you get to the area denoted "Downloading SPIM."
- Click on the link labeled for your computer – for most of us, it is the one labeled "Microsoft Windows (Windows NT, 2000, XP) – **make sure you download the executable version.** That web address is: <http://www.cs.wisc.edu/~larus/SPIM/pcspim.zip> (but go to the upper address first so that you can read the information before downloading). Note that there are Mac, Linux, and Unix versions.
- Follow instructions to load onto your computer (see next page).

James Larus Version of SPIM (2)

- The download software will ask if you want to open or save. Click “save.”
- The “save as” window will pop up – note that the file is “pcspim.zip.”
- **A good approach is to save this program in the Windows folder called “Program Files.” Use the “save in” selection at the top of this pop-up window to select “Program Files.” Click “OK.”**
- The file will be saved to the selected folder. It is about 1.1 Mbytes, so it will take a while unless you have DSL or broadband.
- Now open Windows Explorer, enter the Program Files folder, and find the zipped PCSPIM. Double-click on it to unzip using “Winzip” (ignore all the sales info and simply unzip).

James Larus Version of SPIM (3)

- Click “unzip and install” for PCSPIM to be installed.
- The program will unzip and start to install. Follow directions and exit other programs on your computer.
- **The program will give you the option of storing PCSPIM under “C:/Program Files/PCSPIM.” Click “OK” and the file will be created.**
- When you look in program files using Windows Explorer, you will find a PCSPIM.exe icon that you can drag to your desktop so that you can easily start up this version of SPIM.
- **Once PCSPIM is created, you may open it to see if you get the text, data, memory, and execution windows. If you do, it is correctly installed.**

Alternate SPIM Simulator

- **A version of SPIM is also available locally from Professor Cantrell's website. It may or may not be as up to date as the Wisconsin version.**
- **To download this version of SPIM, first go to the EE 2310 section of Professor Cantrell's web site. The web address is:**

<http://www.utdallas.edu/~cantrell/ee2310/>

- **Download instructions are on the following two slides.**

Alternate SPIM Simulator (2)

- On Professor Cantrell’s website, go to the very bottom of the page and click on “Other courses.”
- On the next page, under EE 2310, click on “Course home page.”
- On the 2310 home page, click on “required software.”
- On the following page, click on “simulators.”
- On this page, click on “The Windows 95/98 Version of SPIM (without SAL).” **Though labeled “95/98,” this is also XP software.**
- You will get a popup window about downloading the software. Make sure the “save” option is selected and click on “OK.”
- Another window will appear asking for a file name under which to save the program. **A good approach is to save in the Windows “Program Files” folder, as for the other simulator.**

Alternate SPIM Simulator (3)

- **The file will download to your computer.**
- **Once it is downloaded, unzip it to create the SPIM simulator. You will get a screen that says “Welcome to PCSPIM for Windows setup.” Simply click on “setup” to get started.**
- **The program will give you the option of storing PCSPIM under “C:/Program Files/PCSPIM.” Click “OK.”**
- **Once PCSPIM is created, you may open it to see if you get the text, data, memory, and message windows.**
- **You may also create a desktop icon as with the other SPIM version.**

Note on MIPSter

- You will note in Dr. Pervin's text that he recommends MIPSter, which is an integrated MIPS development environment.
- **I do not use it, because I prefer the flexibility of Dr. Larus' SPIM assembler, even though I have to use Notepad to write my programs.**
- You may use it if you wish. The website url is:
<http://www.downcastsystems.com/mipster/>
- **You have to pay for a license to use MIPSter, while SPIM is free. The cost of MIPSter, however, is minimal (~\$10).**
- MIPSter does have some nice features, including color-coded syntax types, error checking, and so forth. Its diagnostics are not as good, in my opinion, as SPIM.

Summary: Composing MIPS Programs (1)

- **SPIM programs should be written in a standard text editor (unless using MIPSter).**
- **You can save the programs as `name.s`, per Pervin. My experience is that the “.s” makes only one difference: PCSPIM will show no programs in the “open” window until you click “all files” in the “Files of type” space. The SPIM assembler always recognizes my SPIM listings, even if not labeled as “.s” files.**
- **Program run procedure:**
 - 1. Launch PCSPIM.**
 - 2. To debug your program, simply open it in PCSPIM. It will automatically be assembled as far as it will successfully go.**
 - It will stop on an error and give a brief description.**
 - 3. Open your text editor, change offending text line, and re-save.**

Summary: Composing MIPS Programs (2)

4. Re-open the text document in PCSPIM.
5. Once again, the program will try to assemble.
 - Treat each new error just as was done previously.
 - This is the primary debug “loop” until the program is correct.
6. If the program is syntax error-free, it will assemble, and the display will show the program in the text segment window.
7. You may now run the program:
 - Click on “Go” (under the “Simulator” tab) to see if the program runs. A pop-up window will appear labeled “Run Parameters.” The starting address in the window should show “0x 00400000.” If not, enter that number. Click “OK.” The program should run.
 - If your program writes to the console, check it for the desired result.
 - Next, single step the program to watch instructions execute.
 - Pseudo instructions will be displayed as their component “real” instructions in column three of the text window.

Exercise 2

- Analyze the following program and state in words what it does.
- What is the dimension of the answer?
- What instruction is wrong?

```
main:    .text
         li $v0,4
         la $a0,com
         syscall
         li $v0,5
         syscall
         move $t0,$v0
         mul $t1,$t0,36
         mul $t2,$t1,$t1
         li $v0,4
         la $a0,ans
         syscall
         move $a0,$t2
         li $v0,8
         syscall

         .data
com:     .asciiz "Input dimension in yards:"
ans:    .asciiz "Area is "
```

Homework

- **As usual:**
 - Write down two or three important things you learned today; add to your list.
 - Write down two or three things you did not clearly understand. After finishing the assigned reading, if you still have questions, see me during office hours.
- **Read Pervin chapter 3 (all)**
- **Download PCSPIM to your computer, following instructions in lecture. Use MIPSter if you wish.**
- **Complete Homework #8.**
- **Also do the following:**
 1. List “hello world” in your text editor of choice, assemble with SPIM, run the program, and make sure that it prints out the greeting correctly on the console.
 2. Write a program that requests on the console that a decimal number be input from the keyboard using syscall 5. The program should input the number, then print it out to the console.