332:333 Computer Architecture and Assembly Language Lab

http://www.ece.rutgers.edu/~yyzhang/spring05

This lab class is intended to train the students on both assembly language programming and VHDL design. The students who are taking this class should also be taking 14:332:331 because the programming language and tools that will be used in this class are taught in 331.

Syllabus

Week	Lab Schedule	Location
1(1/17-1/21)	Lab cancelled	N/A
2(1/24-1/28)	TA hands out syllabus; TA teaches how to use SPIM; TA hands out Project 1 (on MIPS programming)	EE 203
3(1/31-2/4)		
4(2/7-2/11)		
5(2/14-2/18)	Project 1 Due; TA grades project 1;	EE 203
6(2/21-2/25)	TA teaches VHDL design; TA hands out project 2 (on VHDL design)	EE 103
7(2/28-3/4)		
8(3/7-3/11)		
9(3/14-3/18)	Spring break	N/A
10(3/21-3/25)	Project 2 Due; TA grades project 2 and hands out Project 3 (on VHDL design)	EE 103
11(3/28-4/1)		
12(4/4-4/8)		
13(4/11-4/15)		
14(4/18-4/22)		
15(4/25-4/29)	Project 3 Due; TA grades project 3	EE 103

Please note that in those weeks when the TAs do not need to teach, or grade projects, the SPIM lab (EE 203) may not be open. However, the students can work on their projects in other EE labs, such as EE105. However, the TAs will be available in their offices during those weeks, so that they can provide prompt help. During those weeks when the TAs need to teach or grade projects, the students MUST attend the lab session in the room as specified above. NO late projects will be taken.

Grading policy:

- Project 1: 20%
- Project 2: 30%
- Project 3: 50%

Preparing yourself for MIPS Assembly Language Programming:

The MIPS programs assigned in this lab course are all meant for execution on a SPIM simulator, described in detail in Appendix A of the text (*Computer Organization & Design – The Hardware/Software Interface* ***Third*** *Edition)* for the course (14:332:331) that accompanies this laboratory.

MIPS is introduced starting in section 2.2 of the text. Section 2.3 through 2.9 gradually detail many MIPS properties. In particular, 2.3 explains the alternate notations for MIPS registers and the format used for memory operands; 2.4 shows how MIPS instructions look in binary; 2.5 introduces logical operations; and 2.6 describes the use of some of the branch and jump instructions. Summaries like that on page 77 are particularly useful. 2.7 adds more register notation and discusses procedure calling and the use of the MIPS stack and stack pointer $sp. The summaries on pages 88, 89, and 90 are especially helpful. 2.8 reminds you about ASCII codes; and 2.9 describes immediate operands, PC-relative branch and jump offsets, and the special register $at. 2.10 introduces the important concept of pseudoinstructions. A pseudoinstruction can be treated by the programmer as if it were part of the MIPS repertoire. In fact, it is not. It is converted, by the assembler, into a sequence of one or more non-pseudoinstructions (i.e., those that are part of the MIPS hardware's instruction set). 3.9 also reminds you about hex notation. Finally, section 2.13 shows several examples of real MIPS assembly code. Many of the exercises at the end of Chapter 2 test your understanding of MIPS programming practices.

Sections 3.2 through 3.3 review signed numbers, negation via the two's complement, the important concept of sign extension, binary addition, and bitwise logical operations. Note again the tables on pages 169, and 175. MIPS integer multiply and divide instructions are summarized on pages 190. MIPS floating-point instructions are summarized on pages 207.

Finally, in Appendix A (which is on the accompanying CD), sections A.1 through A.4 review assemblers and assembly language, introducing some significant new notions – like assembler directives. Sections A.5 and A.6 describe how MIPS memory is subdivided and review MIPS procedure calling conventions. Skipping sections A.7 and A.8, you reach section A.9, where the SPIM simulator user manual begins. After you have studied it, through the first part of section A.10, you will be ready to attack the programming assignments given in this course.
For the additional SPIM information, please refer to 331 course web page www.ece.rutgers.edu/~yyzhang/fall03, and then click "useful links".

SPIM simulator availability

Many PCs containing already-loaded simulators PCSpim (the graphical/windows version) are available in the lab in EE building. You may also download and install PCSpim (it is free!!!) from SPIM homepage (www.cs.wisc.edu/~larus/spim.html).

To active PCSpim on a Windows machine, double-click on the PCSpim icon. To get a display that looks like that on text page A-41: First, using the Window menu, omit the Toolbar and Status Bar, and then maximize the PCSpim window; Second, if necessary, close the Messages and

Console windows. (They can be reopened at any time. During a run, the latter automatically opens whenever input from the keyboard is called for.)

SPIM is also available on workstations for both EDEN (ARC Bldg) and ECE (EE Bldg) systems. Please type ***/user/local/spim/bin/xspim*** for the graphical spim.

SPIM shows the hex value of 32-bit words with the most significant byte (bits 31-24) on the left and the least significant byte (bits 7-0) on the right. Notice that every program text segment begins, at memory location 0x00400000, with eight words of default start-up code that ends with a call (jal) to the global label main followed by an exit syscall. Your program will normally begin (with the main routine) at address 0x00400020. Further, your data area will usually begin at address 0x10010000.