The final will be 3-4:50 PM on Tuesday, December 12 in ITT 328. The test will be closed book and notes, except for one 8.5" x 11" sheet of paper (front and back) with notes, green MARIE Assembly Language handout, and the ***front two*** pages of your MIPS Assembly Language Guide (available at: http://www.cs.uni.edu/~fienup/cs041f06/lectures).

About 65% of the Final will focus on the material since the last test, and about 35% will focus on the material from tests 1 and 2.

MIPS Assembly Language (same as Test 2)
MIPS Processor Architecture: registers, register conventions, addressing modes, memory layout
Basic MIPS Instruction Set: loads/stores, arithmetic instructions, logical instructions, shift/rotate instructions, branch/jump instructions
SPIM Assembler Directives: .data, .text, .word, .globl

In addition to knowledge about the above concepts, the following assembly-language programming skills are to be tested too:
1) translate high-level language control statements (while, for, if, etc.) into MARIE and MIPS assembly language (be able to handle complex Boolean expressions involving ANDs, ORs, etc.)
2) translate high-level language code containing array accesses into MIPS assembly language

NEW MATERIAL THAT WAS NOT ON TEST 2, BUT IS COVERED ON THE FINAL:
MIPS Instruction Set: three ML instruction formats
Subprograms: MIPS Register conventions
MIPS Logical, Shift/Rotate Instructions
SPIM I/O and other System Calls:
SPIM Assembler Directives: .asciiz, .ascii, .align, .space
Arrays: element addressing 1-d, 2-d, 3-d, and higher
Walking pointer through an array

NEW ASSEMBLY-LANGUAGE PROGRAMMING SKILLS THAT WAS NOT ON TEST 2, BUT IS COVERED ON THE FINAL:
3) use MIPS register conventions to decide which arguments/parameters and local variables should be stored in caller-saved ($a and $t-registers) or callee-saved ($s-registers)
4) translate high-level language subprograms into MIPS assembly language (passing parameters into the subprogram using the $a registers, building the call frame on the run-time stack if necessary, save $s and $ra registers if necessary, passing the value returned by a function in the $v0 register, restoring $s and $ra registers if necessary, jr back to the caller)

Hardware Support for the Operating System sections 8.1-8.2

You should understand the general concept of how the operating system with hardware support provide protection from user programs that:

1. go into infinite loops
2. try to access memory of other programs or the OS
3. try to access files of other programs

This involves understanding the concepts of

1. CPU timer
2. dual-mode operation of the CPU, and idea of privileged instructions and non-privileged instructions
3. ways to restrict a user program to its allocated address space

I/O sections 7.1-7.4

General I/O characteristics
I/ O Controller role and function
I/O address mapping: Isolated-I/O vs. memory-mapped I/O
I/O Data Transfer: programmed I/O, interrupt-driven I/O, and direct-memory access (DMA)
General interrupt mechanism
Usage of interrupts by the hardware/operating system to restrict a user program's activities

Misc. Sections:

General Idea of Cache (section 6.1-6.4)
General Idea of Virtual Memory (section 6.5)
General Idea of Pipelining Processors (section 5.5)