The final will be 8-9:50 AM on Tuesday, May 2 in Wright 5. The test will be open-book and open-notes.

About 75% of the Final will focus on the material since the last test, and about 25% will focus on the material from test 2.

Chapter 2.
MIPS Processor Architecture: registers, register conventions, addressing modes, memory layout
MIPS Instruction Set: three ML instruction formats, loads/stores, arithmetic instructions, **logical instructions, shift/rotate instructions,** branch/jump instructions
SPIM System Calls: Section A.9 on CD with text
SPIM Assembler Directives: .data, .text, .word, .asciiz, .ascii, .GLOBL, .align, .space
Arrays: element addressing both 1-d and 2-d
Walking pointer (section 2.15 called "Arrays versus Pointers")
Subprograms: MIPS Register conventions

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 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
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)

Chapter 3.
Section 3.4: Multiplication of unsigned and signed integers
Booth's Algorithm Example - see CD material "Chapter 3" | "In More Depth" | 3.23 Booth's Alg
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

Chapter 8.
Section 8.1
General I/O characteristics

Section 8.2
General concepts of hard disk: dependability, seek time, rotational delay, data transfer time, layout of surfaces, tracks, and sector
RAID: levels, striping (bitwise to large stripes) effects on the number of independent requests that can be handled and the data transfer rate of a single large request.
Operation of RAID when a disk fails

Section 8.4
Bus interconnection: shared collection of wires with lines classified as data, addr., control
Steps of a typical bus transfer
Bus design issues: 1) parallel vs. serial (bus width), 2) bus type: dedicated/(time) multiplexed, 3) bus operations
Synchronous and asynchronous: timing diagrams and protocols (I'll give you a diagram and ask questions about it)
General ideas and trends from the Pentium 4

Section 8.5
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