Test 1 for Computer Architecture will be Thursday, Oct. 3 in class. The test will be open book and notes. Test 1 review topics are:

**Comp. Org. Review:**
High-level programming view: Run-time stack, Compiler, Linker
Read RISC Assembly-language programs
General CPU organization: general purpose (user-visible) registers, control & status registers (PC, IR, MAR, MBR, PSW), ALU, control unit, internal CPU bus
Fetch-Decode-Execute Instruction cycle

**Hardware Support for Operating System:**
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 and non-privileged instructions
3. ways to restrict a user program to its allocated address space

General I/O characteristics
I/ O Controller role and function
I/O address mapping: I/O-instructions 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

Process control blocks (PCB) and OS queues for I/O and process scheduling

**Memory Hierarchy: Cache Memory and Virtual Memory**
Memory hierarchy: levels and goals (speed, capacity, and cost)
Terminology: hit, miss, hit rate, miss rate, hit time, miss penalty; effective (avg.) memory access time
Locality of reference (temporal and spatial)
Cache types: direct mapped, fully associative, set associative; replacement algorithms; effective-access time calculations; write policies (write-through, write-back); cache-coherency solutions
Cache design issues: block size; number/level of caches; unified vs. split cache

Virtual memory: pages, page frames, memory management unit (MMU), page faults, demand paging
Paging: page table, virtual to physical address translation, time and memory efficiency considerations
Page Table Organization and Page table entries (physical page #, disk page address, valid bit, dirty bit, reference bit, protection bits)
TLB (translation lookaside buffer)
Hierarchical (two or more levels) page tables
Segmentation
Combining paging and segmentation
Instruction Pipelining and Superscalar:
RISC vs. CISC characteristics: # of and complexity of instructions, instruction format and length (fixed, variable), number of operands (2, 3, etc) and addressing modes (direct, immediate, register, base-register, PC-relative, indexing), RISC machine language formats

Instruction pipelining, pipeline latches purpose
Pipeline stalls/delay causes:
1) structural hazards (i.e., piece of hardware needed by several stages at the same time)
2) data hazards (i.e., need a value before it is calculated) and bypass signal paths/forwarding to minimize
3) control hazards/branch delays (i.e., fetch wrong instructions before you either know it is a branch instruction or the outcome of the branch is known)
Ways to reduce the branch penalty: Branch Target Buffer

General idea of superpipelining
Superscalar processor stages:
• Instruction Fetch - obtain “next” instruction(s) from memory (I cache)
• Instruction Decode - decode instr(s) and rename user-visible registers to avoid data hazards (WAW & WAR) introduced by out-of-order execution
• Instruction issue - sent instruction to reservation station associated with an appropriate execution unit (integer ALU, fl. pt. ALU, LOAD/STORE memory unit, etc.) to await execution
• Reservation station - dispatch instruction to execution unit when execution unit (e.g., ALU) becomes free and all of the instruction’s operand values are known (so out-of-order execution possible)
• Instruction retire - writes results of potentially out-of-order instructions back to registers to ensure correct in-order completion. Also, communicates with the reservation stages when instruction completion frees resources (e.g., “virtual” registers used in register renaming)

Pentium 4 superscalar example: (CISC x86 program with inner RISC “core”)