Test 1 for Computer Architecture will be Thursday, Oct. 4 in class. The test will be open book and notes. Test 1 review topics are:

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

**Chapter 5: Instruction Set Architectures**
Instruction-set Design Issues: which instructions to include (#, complexity), which built-in data types, instruction format [length (fixed, variable), number of address (2, 3, etc), field sizes], Instruction Types: data movement, arithmetic, Boolean logic, bit manipulation (shifts, bit-wise logical instructions), I/O, Branch instruction
Addressing Modes: direct, immediate, register, register indirect, indirect, base-register, PC-relative, indexing
Instruction pipelining:
pipeline latches/registers - 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) (Section 11.5) 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 prediction, delayed branch, Branch Target Buffer

**Chapter 6. Memory Hierarchy (Only sections 6.1-6.4 on Test 1)**
Types of RAM: static (SRAM) and dynamic (DRAM)
Magnetic disk- organization/format; characteristics; access time (seek time, rotational delay/latency)
Memory hierarchy: levels and goals (speed, capacity, and cost)
Terminology: hit, miss, hit rate, miss rate, hit time, miss penalty
Locality of reference (temporal and spatial)
Idea of cache (and virtual memory)
cache: type (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