Topics for the Second Midterm

CS 315
Spring, 2016

• Chapter 2
  – R-format instructions
  – I-format instructions
  – Storing large immediates in a register with lui and ori.
  – Zero-extended vs Sign-extended immediates.
  – ori vs addi
  – J-format instructions
  – Immediates in branch and jump instructions count words rather than bytes.
  – Calculation of addresses in branch instructions (PC-relative addressing) and jump instructions (pseudodirect addressing).
  – Dealing with branches and jumps to “distant” destinations.
  – Conversion of C source to running executable: preprocessor, compiler, assembler, linker, loader.
  – Object file format.
  – Issues with static linking.
  – DLL’s and how they’re used.
  – Data races, atomic operations, critical sections, mutual exclusion, busy waiting.
  – Implementation of spinlocks with atomic exchange.
  – Implementation of atomic exchange with load-linked and store-conditional.
  – Potential issues with LL-SC.
  – Hardware implementation of LL-SC using snooping.

• Chapter 3
  – Algorithm for decimal addition.
  – Handling overflow.
  – Algorithm for binary addition.
– Implementation of adder using logic gates.
– Algorithm for decimal subtraction.
– Algorithm for binary subtraction.
– Using two’s complement addition for subtraction.
– When can overflow/underflow occur in signed/unsigned addition and subtraction?
– C, MIPS addition and subtraction.
– MIPS exceptions: coprocessor 0, vaddr, status, cause, epc registers
– Decimal multiplication algorithm
– Binary multiplication algorithms
– Number of digits/bits in product
– Performance of text’s basic and improved binary multiplication
– Improving the performance of binary multiplication
– Signed multiplication
– MIPS multiplication: lo, hi registers

• Chapter 4

– One-cycle processor implementing lw, sw, add, sub, and, or, slt, beq, j.
– Stages in executing MIPS instructions: IF, ID, EX, MEM, WB.
– Pipelining: ideal execution when all instructions use the same stages.
– Pipelining: ideal execution when all stages use the same time but different instructions can use different numbers of stages.
– Pipelining: ideal speedup over unpipelined system