MAT 250 — Introduction to Computer Organization and Architecture, Fall 2004

Review Sheet #2

- Addition
  - Two's complement, overflow p61
  - ripple carry adder (8 delays) p65
  - carry look ahead (5 delays) - more logic but computes carry more quickly p76
- Multiplication
  - Multiplicand (M) times multipier (Q) is a product (P) p69
  - Serial Multiplication using computers p70
  - Booth encoding p78
  - Booth Recoding p80
- Steps of a Program
  - compile high level languages into assembly p100
  - assemble assembly into machine code p100
  - machine code is loaded into ALU from disk p101
  - load one instruction at a time along with necessary data p101
  - output is placed on I/O devices p101
- Assembly Considerations
  - machine language vs assembly language p99
  - most machines can address bytes and not bits p102
  - addresses and address spaces p102
  - big endian vs. little endian p102
  - memory map (OS, user , system stack, I/O) p103
  - data section vs. control section p104
  - fetch-execution cycle (fetch, decode, read, execute, repeat) p105
  - pipelined architectures p123,p385,p389
  - instruction set p106
  - recompiled - high-level languages on different machines p107
  - RISC vs CISC p108

- ARC
  - 32 bit address space, byte addressable p109
  - 32 bit word and data types p109
  - big endian p109
  - access a word by the byte at the lowest address p109
  - 32 32-bit registers, a program counter (PC) an instruction register (IR) p110
  - PSR - processor status register - includes conditions codes, z,n,c, and v p110
  - instructions are one word in size p110
  - load-store machine p110
  - %r0 is always 0 p113
  - %r14 is a stack ptr. p113
  - %r15 is a link register p113
  - instructions p117-p120
  - pseudo-ops p121
  - Three address, two address and one address instructions p126-127
  - Size of a program, travel time for data in a program p126-127
  - memory addressing modes (immediate, direct, indirect and register indirect) p129
  - using functions in assembly (registers, data link, stack) p130-136
- Assemblers and Compilers
  - Steps of Compilation p152
    * lexical analysis
    * syntatic analysis (parser)
    * semantic analysis
    * code generation
    * more (optimize, track register usage, allocate variables to registers)
  - symbol table p162-164
  - two-pass assemblers (first pass does what? second pass?) p162
  - forward referencing p162
  - relocatable programs p167,p170
  - linker (object modules, load modules) p168
  - loader p171

- dynamic link libraries p172
- Linker Steps p168
  * resolve external address references
  * relocate each module by combining them end-to-end. Change module addresses.
  * specify starting symbol of the load module
  * specify identifiers and contents of various memory segments if present