IA-32 Instructions II

- Logic Instructions
  - AND, OR, XOR
    - All use 2 operands and put result into destination
    - Example: AND EBX,EAX
  - NOT uses a single operand
    - Takes 1's complement (i.e. flips each bit)
    - Example: NOT EAX
  - NEG negates it's single operand (takes 2's complement)
- Test Instruction Format: TEST dst,src
  - Sets ZF=0 in any matching bits are set in src and dst
  - Operands are ANDed but neither operand is modified
  - destination may be a register or in memory
  - source may be a register, in memory or immediate

IA-32 Instructions II

- Bit Test Instructions
  - BT op1,n Sets CF = bit n within op1
  - BTC op1,n Sets CF = bit n within op1 & flip bit n within op1
  - BTR op1,n Sets CF = bit n within op1 & clear bit n within op1
  - BTS op1,n Sets CF = bit n within op1 & set bit n within op1
  - The operand op1 may be a register or in memory

  - Other examples of available CISC type instructions include:
  - BSF & BSR bit scan forward and bit scan reverse
  - SCANSB, SCANSW, SCANSD for scanning strings in search of a particular value (direction of scan is controlled by DF flag in status register).

IA-32 Instructions II

- Shift Instructions
  - SHL (shift left logical)
  - SHR (shift right logical)
  - SAL (shift left arithmetic; same as SHL)
  - SAR (shift right arithmetic)
- Rotate Instructions
  - ROL (Rotate left without the carry flag CF)
  - ROR (Rotate right without the carry flag CF)
  - RCL (Rotate left including the carry flag CF)
  - RCR (Rotate right including the carry flag CF)
- Shift & Rotate instruction format: OP dst,count
- dst is shifted (Any addressing mode can be used)
- Count must be an 8-bit immediate or in the 8-bit CL register (low byte of ECX)

IA-32 Instructions II

- Subroutine Linkage
  - The CALL instruction is used to call subroutines
  - Example: CALL ROUTINE
    - The address of the instruction following the CALL is the return address
    - The return address is pushed onto the stack before the transfer
    - ESP register points to the top of stack (TOS)
    - The stack grows downward toward lower addresses
    - PUSHAD can be used to save all 8 general purpose registers before call
    - The subroutine uses POPAD to restore the registers before returning
    - PUSH & POP may be used to handle individual registers or data items
    - RET returns control to caller by poping TOS into EIP
    - EBP register points to call frame on stack
    - LIFO stack facilitates nested subroutine calls