Chapter 6 Math

6.1 Negation

- The *neg* instruction performs the twos complement of the operand.
- The operand can be a general purpose register or a memory reference.
- Sets the (SF) flag if the result is negative and the zero flag (ZF) if the result is 0.

6.2 Addition

- Addition is performed using the *add* instruction.
- Syntax:
  - add dest, sourc ; dest = src + dest
  - source can be an immediate, memory, or register.
  - destination can be memory, or register.
  - only one operand may be a memory reference.
- Clears several flags in the *rflags* registers based on results.
  - The flags can be used in conditional statements following the add.
    - Overflow Flag (OF): set if addition overflows
      - The overflow flag is set when the add is carried out in a particular binary representation produced a result that no longer makes sense in that binary representation.
      - The result overflowed the bounds of the binary representation.
    - Sign Flag (SF): set to the sign bit of the result.
    - Zero Flag (ZF): Set if the result is O.

- We can increment or decriment numbers by 1 using *inc/dec*.

6.3 Subtraction

- Subtraction is done using the *sub* instruction
- Syntax:
  - sub dest, src ; dest = dest - src
  - Follows the same patterns as add.
  - Sets the same flags as *add*

6.4 Multiplication

- Multiplication of unsigned integers is done with *mul*
- Multiplication of signed integers is done with *imul*
  - *imul*
    - has three different forms.
      - 1st: has 1 source operand only.
      - 2nd: has source and destination operand.
      - 3rd: one destination and two source operands.
  - One operand *imul*
    - Multiplies the value in rax by the source operand.
    - Stores in rdx:rax
      - Multiplying two 64-bit integers is 128-bit.
      - The lower bits are in rax higher bits in rdx.
    - Source could be memory or a register.
  - Two operand *imul*
    - Allows specifying source operand.
      - Can be a register, memory reference, or immediate.
    - imul dest, srce ; dest = dest * srce
  - Three operand *imul*
    - Destination register is not one of the factors.

            - imul dest, src1, src2 ; dest = src2 * src1
                - src2 must be an immediate.

6.5 Division

- Division returns a quotient and a remainder.
- *idiv* instsruction uses a single source operand.
    - can be a register or memory reference.

6.6 Conditional Move Instructions

- Conditional move instructions can be used instead of branching.
    - Branching causes the CPU to perform branch prediction which slow down the CPU with missed predictions.
    - These include tests and a mov instruction.
        - Example:
            - cmovz : move if result is zero.

6.7 Why Move to a Register?

- Both add and sub can operate on values stored in memory.
- If the value from memory is used in more than one operation it might be faster to move it into a register first.
    - More useful if the code is going to be executed a high number of times.
    - If the uses are more than a few instructions apart then may not be worth doing.