SASM Instructions

August 28

CSC201 Section 002
Fall, 2000
Watch Out for Reserved Words!

• (see Appendix A of our textbook)
**Instruction Formats and Operands**

• Instructions have at most two operands
  – called "two address" instruction format

• The first operand is also the "destination", where results are stored

• Operands can be of two types: in memory, or "immediate values"
  – operands in memory are referred to by a label (for now)

• Immediate values
  – are constants specified in the instruction
  – have the same format as DD and DB "initial values"
  – cannot be the destination of an instruction!

• (we will not use registers in SASM instructions)
The Move Instruction

- move A,B -- copy the contents of doubleword B into doubleword A
  - example: move myval1, myval3
  - example: move myval2, -2735

- moveb A,B -- copy the contents of byte B into byte A
  - example: moveb mychar1, mychar2
  - example: moveb mychar3, 'a'

- movezx -- copy character to doubleword-sized integer
  - example: movezx mydouble1, mychar3

- movesx -- copy character-sized integer to doubleword-sized integer
  - example: movesx mydouble2, mychar3
Integer Arithmetic Instructions

- `iadd A, B` -- add contents of B to contents of A, store result in A
- `isub A, B` -- subtract contents of B from contents of A, store result in A
- `ineg A` -- change the sign of A, store result in A
- `imult A, B` -- multiply A by B, store result in A
- `idivi A, B` -- integer divide A by B, store result (quotient) in A
- `irem A, B` -- compute A modulo B, store result in A
  - watch out for negative numbers!
**Condition Codes**

- These are "flags" that are set by arithmetic operations
- **SF** is the "sign flag"
  - Equals 1 if the result of the operation is negative
- **ZF** is the "zero flag"
  - Equals 1 if the result of the operation is zero
Overflow

• Results of arithmetic operation may not "fit" in a destination operand of fixed size
• We'll ignore for now (just for a week or two)
Divide Examples
Expression Examples

- \( z = w + x - y \)
  - move z, w
  - iadd x
  - isub y

- \( j = k + (m \times (n - 1)) \)
  - move j, n
  - isub j, 1
  - imul j, m
  - iadd j, k

- \( q = (r + s) / (t - u) \)
Floating Point Arithmetic Instructions

- J and K are floating point numbers; neither may be an immediate value
- `fpadd J, K` -- add J to K, store floating point result in J
- `fpsub J, K` -- subtract K from J, store floating point result in J
- `fpmul J, K` -- multiply J times K, store floating point result in J
- `fpdiv J, K` -- divide J by K, store floating point result (quotient) in J
- Note: we have no conversion routines between integer and floating point numbers
  - all we can use is "put_fp" to see the results