Class 33: A Simple Three-Address Code Representation

Held: Wednesday, 16 November 2011

Summary: We consider the intermediate code for our compiler.

Related Pages:
- EBoard.
- Lab: An Introduction to SSTAC.
- Reading: Introduction to SSTAC.

Notes:
- Yes, if you need extra time on Phase 4, you can have it.
- But Phase 5 is now available, and you should start work on it soon.
- I got a request for a class on Make and another class on debugging. I’ll probably offer that as extra sessions (at a different time than we normally meet). What do you think about a more general “Things every real C/*nix programmer should know” series, perhaps once per week next semester?

Overview:
- Three-Address Code.
- Designing our own TAC.

Three-Address Code

- Goal: An intermediate representation
  - That is not closely tied to any particular architecture.
  - That is fairly easy to generate.
  - And fairly easy to convert to real assembly code.
- Some simplifications
  - Any addressing mode is allowed for each parameter (most architectures allow only one memory reference per parameter)
  - No instruction takes more than three parameters
- Sam’s additional simplification
  - Separate instruction and data memory.

Designing Our Own TAC

- We will design the TAC in class today.
- I will implement a representation and interpreter by Friday’s class.
- The questions
- What instructions do you want?
- What addressing modes do you want?
- How will we represent these textually?

- Variable and constant declarations ("constant" memory)
- Arithmetic operations
- Control operations
- I/O operations
- Misc. operations