CS 536 Announcements for Wednesday, May 1, 2024

Course evaluation – log into HelioCampusAC.wisc.edu using your NetID

Final Exam
- Sunday, May 5, 2:45 – 4:45 pm
- B102 Van Vleck
- bring your student ID

Last Time
- wrap up optimization
- copy propagation

Today
- wrap up course / review

Where have we been?
CS 536: Introduction to Programming Languages and Compilers

What does a programming language consist of?

- tokens
- grammar
- static semantic analysis

What else? What choices are made?

- scoping rules
  how do we match var decls with var uses?
  what is allowed? nested functions, nested var decls
- types
  what types are there?
  how do the types relate to each other?
- parameter passing
  what ways are there to get info to a called procedure?
  what impacts are there on the calling procedure?
- when do we check for things?
  at compile time → static
  at run time → dynamic
  or both?

Where have we been?
CS 536: Introduction to Programming Languages and Compilers

How do we translate a PL into something a computer can run? i.e., compilers

- recognizing tokens
  - reg exprs & FSMs
  - tools for translation
- recognizing languages – context-free grammars, parsing
  - what can be parsed
  - how? top-down vs bottom-up
- enforcing scoping and typing rules
- developing data structures that assist our translation/representation/translation
  - AST, parse trees, symbol tables
- how do we organize and manage memory
  - variables – where store, how accessed
    - local vs global vs non-local
  - using register and stack
- handling control flow within a program
  - interprocedural – how fctn calls & returns are implemented
  - intraprocedural – how loops & selection stmts are implemented

How can we make our translation better?

- intermediate representations
- IR optimizations
  - copy propagation, LICM & other loop optimizations
- MC optimizations
  - peephole optimizations

Course wrap-up

Covered a broad range of topics

- some formal concepts
- some practical concepts

What we skipped

- object-oriented language features – classes with methods, inheritance
- dynamically-allocated memory management
- linking and loading
- interpreters
- register allocation – how can we use registers to make our program faster and still correct & safe
- dataflow analysis – reaching defs, unreachable code, live variables, aliasing info
- performance analysis – determining bad memory accesses, which parts of code use a lot of execution time (or memory)
- proofs – correctness
  termination
  complexity

Final Exam, Sunday, May 5, 2:45 pm
B102 Van Vleck

Bring your UW Student ID

Reference material provided along with exam:
- copy of the base grammar
- compiler class reference with selected class, methods, fields

Topic overview

Basic ideas of scanning & parsing

Symbol-table management / name analysis
- static scoping
- dynamic scoping

Type checking

Runtime storage management
- general storage layout
- activation records
- access to variables at runtime (parameters, locals, globals, non-locals)

Hw 6

Parameter-passing modes ← Hw 5, sample exam

Code generation

Optimization
- goals
- optimization techniques (e.g., peephole optimization, copy propagation)

Extending
- grammar
- AST
- name analysis
- type checking
- code generation

to handle new language constructs

see Hw 4, sample exam