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

- types

- parameter passing

- when do we check for things?
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
- recognizing languages
- enforcing scoping and typing rules
- developing data structures that assist our translation/representation/translation
- how do we organize and manage memory
- handling control flow within a program
  - interprocedural
  - intraprocedural

How can we make our translation better?
- intermediate representations
- IR optimizations
- MC optimizations
Course wrap-up

Covered a broad range of topics

• some formal concepts
• some practical concepts

What we skipped

• object-oriented language features
• dynamically-allocated memory management
• linking and loading
• interpreters
• register allocation
• dataflow analysis
• performance analysis
• proofs
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)

Parameter-passing modes

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