CSE443 Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
Final Exam

5/15/2022, Monday
Knox 4
Start @ 8:00 AM
End @ 11:00 AM
Arrive by 7:50 AM
Entry not guaranteed after 8:30 AM
Exam format

- Expect 4 short essay questions (choose from ~6). We will use BlueBooks.

- I expect you to take about 30 minutes per question (about 2 hours total).

- This leaves you with about 1 hour to proofread/edit your responses.
Possible Exam Questions

- Anything from homeworks,

... and ...
Type checking
(Semantic processing)

- Explain how type errors are detected. Discuss how type information is gathered, stored and checked. Pick a concrete syntactic construct that can contain a type error, and explain how type checking detects the error.
Intermediate Code Generation

- Explain how short-circuit Boolean expressions are translated into intermediate code. Discuss how jump targets can be determined during backpatching. Illustrate by showing how a concrete Boolean expression involving at least two Boolean operators is translated into intermediate code.
Register Allocation and Assignment

- Describe the getReg(I) algorithm, answering the questions of what data structures it uses, when and how these structures are updated. What is meant by "spill", when does it occur, and why is it needed? Demonstrate with a concrete example.
Symbol Table Usage

Describe the structure and use of a symbol table. Explain which phases of the compiler use the table, including what data is written to or read from the table during each phase. Give a concrete code example and the corresponding ST.
Invocation Records

- Describe a typical layout for an invocation record, detailing what information is stored in the record. Explain how variable length parameters and variable length local data can be accommodated. Discuss the location and use of the stack and top pointers. Give concrete example.
Function Calls

- Explain how a function call takes place. Include in your discussion mention of the roles of the caller and callee in setting up the invocation record: discuss both calling and return sequences, and the division of labor between caller and callee. Explain how machine state is remembered at the call and restored at return. Cover how recursive calls are handled (do NOT discuss tail-call optimization). Give concrete example.
Optimizations

- Pick an optimization and explain the benefit(s) of having the compiler apply it to code, and sketch how it works for a concrete example.

- Ex:
  - tail-call optimization
  - code motion
  - dead code elimination
Thanks for a great semester!

Have a wonderful summer!

Congrats to everyone graduating!!