Comprehensive Examination
Compiler Construction
Spring 1996

This is a closed-notes, closed-book exam administered under the Rice Honor Code. You have two hours to complete the exam.

We have tried to give you all the information required to complete each question. If you reach a point where you are missing a critical piece of information, make a reasonable assumption, record it in your answer book, and continue.

---

1. *Syntax analysis* — A classic example of an ambiguous grammar is the `if-then-else` problem.

$$
\begin{array}{lcl}
\langle \text{stmt} \rangle & ::= & \texttt{if } \langle \texttt{expr} \rangle \texttt{ then } \langle \texttt{stmt} \rangle \texttt{ else } \langle \texttt{stmt} \rangle \\
 & | & \texttt{if } \langle \texttt{expr} \rangle \texttt{ then } \langle \texttt{stmt} \rangle \\
 & | & \texttt{other statements}
\end{array}
$$

   (a) Show a code fragment that has two possible parse trees with this grammar. Draw the parse trees.

   (b) Modify the grammar to resolve this ambiguity in the obvious way—that is, to match the dangling `else` with the closest unmatched `then`.

2. *Linkage Conventions* — Your employer has asked you to design the linkage convention for their next generation system. Your convention will be used to support a number of languages, including Pascal, Fortran, and C.

   There are several key issues that you must decide today.

   (a) How are parameters passed from caller to callee? How are arrays passed?

   (b) Who saves registers at the point of call? Explain the rationale behind your choice.

   (c) How is addressability established for non-local variables? Assume that they come in three flavors: C's static variables, C's global variables, and Pascal's lexically scoped local variables.

   (d) How are return values handled? In particular, C has the ability to return a structure.

   (e) Sketch the layout of an activation record. Include all the major fields required to support your implementation of the procedure abstraction.

3. *Code generation* — Aho, Sethi, and Ullman describe two methods for "optimal" code generation: the tree-labelling algorithm due to Sethi and Ullman and the dynamic programming algorithm due to Aho and Johnson.

   (a) For each algorithm, describe the machine model on which it operates and the criteria by which it is "optimal".

   (b) For each algorithm, what is its asymptotic complexity?

   (c) On a modern machine, which algorithm would you expect to generate better code? Justify your opinion.

4. *Analysis and Optimization* — Aho, Sethi, and Ullman describe several different "data-flow analysis problems." Choose one.

   (a) Write down the equations that describe the data-flow problem and define each set mentioned in the equations.

   (b) Show an algorithm for solving these data-flow equations.

   (c) What is the complexity of your algorithm? How many set operations will it perform before halting?

   (d) Describe an optimization that might use the results of this data-flow analysis.