Computer Science Department
Stanford University
Comprehensive Examination in Compilers
Autumn 1996

October 17, 1996

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. Be sure you have all the pages of this exam. There is 1 page.

   The exam takes 30 minutes.

3. This exam is OPEN BOOK. You may use notes, articles, or books—but no help from other sentient agents such as other humans or robots.

4. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

CS Comprehensive Exam: Compilers (30 points)

Question 1 (9 points)

Consider the following block of 3-address code:

```
1)  a := b + c
2)  d := b
3)  e := c + d
4)  f := a - e
5)  d := a
```

Assume that the value of each of a through f is needed following the block. Give six transformations that may be performed on this block to improve the running time on a typical machine. Indicate the change made at each of your six steps (it is sufficient to give the rewritten line only, rather than the whole block) and *give the name of the transformation or brief justification.* You may apply more than one transformation to a single statement, but use separate steps to do so.

Question 2 (11 points)

For the grammar

$$\begin{array}{lrcl} 1) & S & \rightarrow & aS \\ 2) & S & \rightarrow & b \end{array}$$

a) [8 points] Construct the LR(0) sets of items (sometimes called SLR items; they involve no lookahead) and show their transition diagram.

b) [3 points] Give the LR(0) parsing table and indicate any conflicts that arise.

Question 3 (10 points)

Suppose we wish to build a lexical analyzer that recognizes the following three tokens:

1) **01**
2) **010**
3) **0*10***

Assuming the LEX priority rules (longest token recognized, and in case of ties, pick the token listed first), draw a deterministic finite automaton that recognizes the above three tokens. Indicate, for each state, which, if any, token is recognized. Also, since a token may not be recognized until further symbols are read by the automaton, indicate how far back the recognized token ends. For example, attaching $(2, 3)$ to an accepting state means that the second token, **010**, was recognized, and the last 3 symbols read by the automaton are not part of the token.

(19)