Exam in course

TDDA 37 Compiler Construction 2000-05-03  08.00 - 12.00

No books or other aids allowed.

Max = 32 points, 16 points needed to pass.

Teacher on duty: Jonas Wallgren (only by phone)
Problem 1 (2p) Phases and passes
Why could a compiler need several passes?
Pascal was designed for one-pass compilation. Why could that be desirable?

Problem 2 (2p) Symbol table
Describe how the hash-based symbol table model presented in the course handles
a) declaration of a variable.
b) termination of a block.

Problem 3 (4p) Top-down parsing
Explain and remedy the problems in the grammar

A ::= Aa | bA | BC | dB | e
B ::= Af | Ag | hB | iB | j

which will be used for recursive descent parsing.

Problem 4 (5p) LR parsing
Show, using automaton and tables, how the string

a · a + a · (a + a)

is parsed according to the grammar

E ::= T | E + T
T ::= F | T * F
F ::= a | (E)

where E is the start symbol.

Problem 5 (5p): Intermediate code generation
Transform the code below to quadruples, postfix code, and abstract syntax tree:

while y<20 do
    if x>15
        then x:=x+1
    else y:=y-1;

Problem 6 (3p) Code optimization
What is a loop?
Explain, using clear examples, the loop optimization methods presented in the course.
Problem 7 (5p) Syntax directed translation

A simple version of a FOR statement could be described using this rule:

\[ <\text{for-stat}> ::= \text{FOR } i := <\text{expr}>_1 \ \text{TO} \ <\text{expr}>_2 \ \text{DO} \ <\text{S}> \]

Semantically the statement is equivalent to:

\[
\begin{align*}
\text{BEGIN} \\
\quad i := <\text{expr}>_1; \\
\quad \text{temp} :=<\text{expr}>_2; \\
\quad \text{WHILE } i <= \text{temp} \ \text{DO} \\
\quad \quad \text{BEGIN} \\
\quad \quad \quad <\text{S}>; \\
\quad \quad \quad i := i + 1; \\
\quad \text{END}; \\
\text{END}; \\
\end{align*}
\]

Write a syntax directed translation scheme, with attributes and semantic rules, for translation of the FOR statement to quadruples. Assume that the translation scheme is to be implemented in a bottom-up parsing environment. Explain all introduced attributes and functions. Let \(<\text{expr}>_1, <\text{expr}>_2\) and \(<\text{S}>\) be non-terminals for which you don’t need to generate quadruples, and assume that the result of e.g. \(<\text{expr}>\) is available in the attribute \(<\text{expr}>.\text{ADDR}\).

Problem 8 (2p) Bootstrapping

Explain the concepts of rehosting and retargeting. Describe how they are done. Use T diagrams.

Problem 9 (4p) Code generation for RISC

a) What is branch prediction and when is it used? Give an example! Why is it important for pipelined processors?

b) Shortly explain software pipelining. Give a simple example.