Exam in course

TDDA 37 Compiler Construction 1999-04-17  09.00 - 13.00

No books or other aids allowed.

Max = 32 points, 16 points needed to pass.

Teacher on duty: Jonas Wallgren
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 (3p) Symbol table
a) Describe how the hash-based symbol table model presented in the course handles
   1) Beginning of a new block
   2) Termination of a block
b) Which is the greatest disadvantage of representing a symbol table as a linear list?

Problem 3 (4p) Top-down parsing
The following grammar, where A is start symbol, should be used for top-down parsing:

\[
A ::= Ax | By | p \\
B ::= Ay | Bx | q
\]

If there are no problems with the grammar: Write a parser. You don’t need to declare
variables. Assume there is a procedure `scan()` which updates the global variable `token`.
If there are any problems with the grammar: Explain the problems and the solutions to
them.

Problem 4 (5p) LR parsing
a) Construct the LR-item sets for the grammar below, where N is start symbol:
   \[
   N ::= NEENx | 0 \\
   E ::= ENNE | 1
   \]
b) Decide, mainly from your constructions above, whether the grammar below is LR(0):
   \[
   N ::= NEEN | 0 \\
   E ::= ENNE | 1
   \]
c) Show, using parse tables and stack, how the string `0110x110x` is parsed (according to
the grammar in a).

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 (4p) 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:

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

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.