Exam in courses

TDDB 44 Compiler construction
TDDB 29 Compilers and interpreters
2005-03-30, 08.00 - 12.00

Aids: None.

Max = 26 points, 13 points to pass.

Teacher on duty: Jonas Wallgren - by phone

Problem 1 (4p) Formal languages and automata theory

NB! Only TDDB 29 (Compilers and Interpreters) students should solve this problem!

L= the language of strings w over {0,1} such that w contains 00 iff it contains 11. (Eg. 110100 and 01010 belongs to L. 110 and 100 don't.)

a) Construct a DFA for L.

b) Construct a regular expression for L.

Problem 2 (2p) Top-down parsing

The following grammar should be used for recursive descent parsing. What is/are the problem(s)? Rewrite the grammar to make it useable.

```
X -> XaX | bYb | c
Y -> YdY | eXe | f
```

Problem 3 (2p) LR parsing

Invent a language that is LR. Construct a grammar for the language that isn't SLR(1).

Explain why the language is LR and why the grammar isn't SLR(1).

Problem 4 (4p): Intermediate code generation

Translate the following code segment to quadruples, postfix code, and abstract syntax tree (see problem 8 for the `randomorder` construction):

```
randomorder
  for i in 1..5 loop
      y:=y-i;
  endloop;
  y:=y+1;
end;
```

Problem 5 (2p) Symbol tables

Describe what happens in a tree based symbol table at

a) variable declaration

b) variable use

c) block entry

d) block exit

Problem 6 (2p) Boot strapping

On the M machine there is an compiler for the language X - both exucutable and as source code written in X itself. On the N machine there is no X compiler. Y is a new programming language - an extension of X. How to make, using bootstrapping, a Y compiler usable on N?

Problem 7 (4p) Optimization

Give an example of one piece of code before and after optimization where the loop optimization methods presented in the course have been used one by one. Name the methods and indicate how/where they are used, for each method.

Problem 8 (6p) Syntax directed translation

An Algol-like language is augmented with an `random` statement in the following way:

```
<random-stmt> ::= randomorder
                      <random-stmts>
                  end;
```

Where `<random-stmts>` is a sequence of up to 3 statements.

A `random` statement executes the (max 3) statements in `<random-stmts>` in arbitrary order - different between executions. E.g. the program segment

```
print(1);
randomorder
  print(2);
  print(3);
end;
print(4);
```

could result in the printouts 1234 and 1324.

Write the semantic rules - a syntax directed translation scheme - for translating the `random` statement, including `<random-stmts>`, to quadruples. Assume that the translation scheme is to be used in a bottom-up parsing environment using a semantic stack. Use the grammar rule above as a starting point, but maybe it has to be changed. Assume that there is a predefined function `rand()` that returns true or false. Also assume that there is a `<stmt>` rule.

You are not allowed to define and use symbolic labels, i.e. all jumps should have absolute quadruple addresses as their destinations. Explain all the attributes, functions, and instructions that you introduce. State all your assumptions.

Problem 9 (4p) Code generation for RISC

NB! Only TDDB 44 (Compiler Construction) students should solve this problem!

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.