Department of Computer and Information Science
Database and Information Systems
Prof. Dr. Michael Grossniklaus

Universität Konstanz

Winter Semester 2018/2019

INF-21440

Compiler Construction

Sample Exam

Your Student Number

0	1	/						

Read the following instructions *before* you start to write your answers.

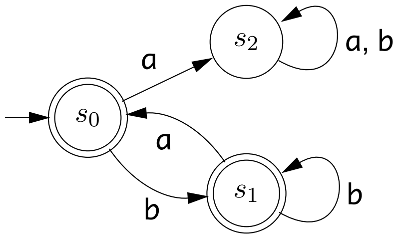
- This exam consists of **6 questions**, each of which is worth a total of **15 points**.
- Answers can be given in **German** or **English**.
- Write your answer on the **designated lines** below each question.
- If you use the back of a page to write your answer, **clearly indicate** to which question it belongs.
- Do **not** write in the margins.
- Your answers should be written in **blue** or **black ink** and not in pencil.
- Any parts of your answer that you do not want to be included as part of the final answer should be **clearly scored through** with your pen.
- With the exception of a calculator, **no electronic devices** (laptops, phones, music players, smart watches, dishwashers, *etc.*) can be used during the exam.

Do not write below this line

1	2	3	4	5	6	Total	Exam	Project	Final

1 Scanners

(a) Describe *informally* the languages accepted by the following finite automaton.



. . . .

. . . .

. . . .

. . . .

. . . .

(b) Consider the regular expressions $(ab|ac)^*$.

(i) Use Thompson's construction to construct an nondeterministic finite automaton for the regular expression.

(ii) Convert the nondeterministic finite automaton to a deterministic finite automaton.

(iii) Minimize the deterministic finite automaton

Answer ____

Answer

2 Parsers

(a) Write a context-free grammar for the Backus-Naur form (BNF) notation for context-free grammars.

(b) Consider the following grammar.

$$
\begin{array}{r|lcl}
1 & A & \rightarrow & B\ \mathsf{a} \\
2 & B & \rightarrow & \mathsf{dab} \\
3 & & | & C\ \mathsf{b} \\
4 & C & \rightarrow & \mathsf{c}\ B \\
5 & & | & A\ \mathsf{c}
\end{array}
$$

Does this grammar satisfy the LL(1) condition? Justify your answer. If it does not, rewrite it as an ll(1) grammar for the same language.

3 Intermediate Representations

Consider the following code fragment.

```
IF c[i] # 0 THEN
   a[i] := b[i] DIV c[i]
ELSE
   a[i] := b[i]
END
```

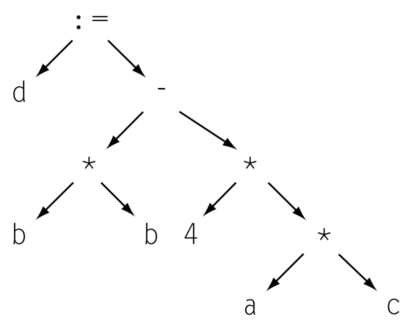
(a) Show how the following code fragment might be represented in an *abstract syntax tree* and in a *control-flow graph*.

(b) For what applications would one representation be preferable to the others?

Answer ____

4 Code Shape

(a) Use the treewalk code-generation algorithm to generate naive code for the following expression tree. Assume an unlimited set of registers and that the variables are stored at offsets @*a*, @*b*, @*c*, and @*d*, respectively.



Answer ____

(b) Sketch how you would structure the following control flow in terms of basic blocks. You only need to write the code for tests jumps.

```
int i = 0;
do {
   if (i > 20) {
      break;
   } else {
      i++;
   }
} while (true);
```

Answer

5 Instruction Selection and Scheduling

Consider the following code fragment. Assume that the processor has a single functional unit, loads and stores take three cycles, a multiply takes two cycles, and all other operations complete in a single cycle.

loadAI	r_{arp}, @a	$\Rightarrow$	r_1
loadAI	r_{arp}, @b	$\Rightarrow$	r_2
add	r_1, r_2	$\Rightarrow$	r_4
loadAI	r_{arp}, @c	$\Rightarrow$	r_3
sub	r_3, r_1	$\Rightarrow$	r_5
mult	r_4, r_5	$\Rightarrow$	r_6
multI	r_3, 2	$\Rightarrow$	r_7
add	r_6, r_7	$\Rightarrow$	r_8
storeAI	r_8	$\Rightarrow$	r_{arp}, @d

(a) Draw the *dependence graph* $\mathcal{D}$ of this code fragment and annotate every node with its *cumulative latency*.

(b) Use the *local list scheduling* algorithm from the lecture with cumulative latency as a priority criterion to schedule this code fragment.

Answer

6 Register Allocation

Consider the following code fragment. Assume you have a total of $k = 4$ physical registers, r_1, r_2, and r_3, plus the reserved register r_{arp}. Note on the ILOC instruction set architecture the number of registers needed to generate code for values that live in memory is $\mathcal{F} = 2$.

```
loadAI   r_arp, @v1  =>  vr_1
loadAI   r_arp, @v2  =>  vr_2
loadAI   r_arp, @v3  =>  vr_3
add      vr_1, vr_1  =>  vr_4
loadAI   r_arp, @v4  =>  vr_5
mult     vr_2, vr_3  =>  vr_6
mult     vr_4, vr_6  =>  vr_7
mult     vr_5, vr_7  =>  vr_8
storeAI  vr_8        =>  r_arp, @a
```

(a) Write down the *live ranges* as a set of intervals.

(b) Show the result of using the *top-down* local algorithm on it to allocate registers.

(c) Show the result of using the *bottom-up* local algorithm on it to allocate registers.

Answer

Answer ______________________________