Candidate Number

G5035

THE UNIVERSITY OF SUSSEX

BSc and MComp SECOND YEAR EXAMINATION
January 2017 (A1)

Compilers and Computer Architecture

Assessment Period: January 2017 (A1)

DO NOT TURN OVER UNTIL INSTRUCTED TO BY THE LEAD INVIGILATOR

Candidates should answer TWO questions out of THREE. If all three questions are attempted only the first two answers will be marked.

The time allowed is TWO hours.

Each question is worth 50 marks.

At the end of the examination the question paper and any answer books/answer sheets, used or unused, will be collected from you before you leave the examination room.

1. This question is about the front-end of a compiler.

   (a) Explain the purpose of the lexing, parsing and type-checking phases in a compiler. What does each phase take as input and return as output? [10 marks]

   (b) The lexer and parser of a compiler can be written “by hand”. What is an alternative to hand-writing lexer and parser? List advantages and disadvantages of either approach. [10 marks]

   (c) Consider a Java-like language with keywords including `if`, `then`, `else`, `for`, `while`, `do`, `repeat` and `until`. Identifiers in the language are given by the regular expression $[a-z][a-zA-Z0-9]^*$.

   We assume that the token corresponding to "`if`" is `T_Keyword_If`, the token corresponding to "`then`" is `T_Keyword_Then`, and likewise for the remaining keywords. The token for identifiers is `T_Keyword_Ident(...)` where dots stand for the identifier, e.g. "`numberOfConnections`" becomes the token `T_Keyword_Ident("numberOfConnections")`.

   i. Note that the string "`if`" is the description of identifiers in the language. Is "`if`" lexed into the token `T_Keyword_If` or into `T_Keyword_Identifier("if")`? Justify your decision, and explain what mechanism the lexer uses to achieve this effect.

   ii. Assume your L compiler is lexing a string that starts as follows:

   ```
   int ifthen = 1; ifthen += 1; ...
   ```

   Is the string "`ifthen`" lexed into two consecutive tokens `T_Keyword_If T_Keyword_Then` or into a single token `T_Keyword_Identifier("ifthen")`? Justify your decision, and explain what mechanism the lexer uses to achieve this effect.

   iii. Tokens like `T_Keyword_Identifier("ifthen")` carry two pieces of information, namely (a) the nature of the token (in this case `Identifier`) and (b) the name of the identifier (in this case `ifthen`). Which of those two pieces of information is used by the parser for checking whether the input is syntactically well-formed? Justify your answer.

   [20 marks]

   (d) The lexical definitions of programming languages typically define what counts as “whitespace”, for example containing

   ```
   whitespace = (' ' | '\n' | '\t')+
   ```

   Why is it important to specify what counts as “whitespace” when defining the lexical level of a language? [10 marks]

2. This question is about code generation.

   (a) What data-structure is the input of a code-generator? What feature of this data-structure is designed to make the definition of code-generators simple, and to make the execution of code-generation fast? [10 marks]

   (b) Assume in the translation of a program P the compiler has a choice between allocating the variable x on the heap or on the stack. (Choice here means that both options generate the same observable program behaviour.) Do you recommend stack-allocation or heap-allocation in this case? Justify your recommendation. [10 marks]

   (c) In the lectures and tutorials we used (a variant of) the following simple programming language to explain code-generation.

$$\begin{array}{lcl} E, E' & ::= & n \mid x \mid E + E' \\ P, Q & ::= & \text{while } E > 0 \text{ do } P \mid \text{for } i = E \text{ to } E \text{ do } P \\ & \mid & \text{if } E > 0 \text{ then } P \text{ else } Q \mid x = E \end{array}$$

Here x ranges over variables, and n, i over integers. Design a code generator for this language. The target machine architecture for your code generator is the **register machine** with an unlimited number of registers (as introduced in the lectures). Make sure to explain your design appropriately. As a reminder, the commands of the machine language for the target architecture are listed below. We assume that the registers are named $R0, R1, R2, ...$, and r, r' range over registers. [30 marks]

Command	Meaning
Nop	Does nothing
Pop r	Removes the top of the stack and stores it in register r
Push r	Pushes the content of the register r on stack
Load r x	Loads the content of memory location x into register r
LoadImm r n	Loads integer n into register r
Store r x	Stores the content of register r in memory location x
CompGreaterThan r r'	Compares the content of register r with the content of register r'. Stores 1 in r if former is bigger than latter, otherwise stores 0
Jump l	Jumps to l
JumpTrue r l	Jumps to address/label l if the content of register r is not 0

Command	Meaning
Plus r r'	Adds the content of r and r', stores result in r
Minus r r'	Subtracts the content of r' from r, stores result in r
Times r r'	Multiplies the content of r and r', stores result in r
Divide r r'	Divides the content of r by r', stores result in r
Negate r	Negates the content of r and stores result in r

3. This question is about computer architecture and its effect on performance.

   (a) A modern CPU uses many different forms of memory such as registers, static RAM, dynamic RAM and flash memory. Explain why. Outline the advantages and disadvantages of each. [10 marks]

   (b) What is the meaning of data locality in the execution of a program? Give an example of a program fragment that exhibits a high degree of data locality. Explain why your example exhibits data locality. [18 marks]

   (c) What is the purpose of caches in a CPU? How can caches affect the performance of a program? [10 marks]

   (d) Consider the following program that first executes a subprogram `P` and then two identical assignments.

```
P;
x = x + 1;
x = x + 1
```

   Assume the translation of this program compiles both assignments to the same machine code. Explain under what circumstances the execution of this program on a modern CPU takes much longer to execute the first assignment than the second? When do both commands take roughly identical time to execute? [12 marks]

End of Paper