University of Oxford

Department of Computer Science

Compilers — Practical Assignment

December 2022

> *Your report on this assignment should be submitted online via Inspera by 12 noon on Friday, Hilary Week 2 (27th January) 2023. The assignment is worth 35 marks out of a total of 100 marks for the course. Late submission will incur a penalty as set out in the Course Handbook and the Notice to Candidates.*

This assignment is based on the Lab 4 compiler. It asks you to implement an extension to the picoPascal language, as specified later in this document.

To carry out the assignment, you will need access to a UNIX environment containing version control tools, an OCaml compiler, software tools that enable cross-compiling for the ARM, an ARM emulator, and other support software. Hints and resources for setting up such an environment are provided on the course website. Lab machines are available for remote access.

You should ensure that you are working with the latest versions of the practical materials for the course, for example by cloning the Mercurial repository to a directory named `task` with the command

```
$ hg clone https://spivey.oriel.ox.ac.uk/hg/compilers task
```

or by cloning the identical Git repository with

```
$ git clone https://spivey.oriel.ox.ac.uk/git/compilers.git task
```

If you download the materials directly from the website you should ensure that you have the revision called "Improved infrastructure for 2022".

Candidates are required to submit a written report, describing the changes they made to the compiler in order to support the extensions, and the provisions they have made for testing the modified compiler.

A typical report will consist of five to ten pages of text with embedded code fragments, plus listings of the compiler changes and additional test cases. A sample report for a different exercise is provided on the course website at

```
https://spivey.oriel.ox.ac.uk/compilers/Christmas_assignment
```

and shows the suggested structure, with a narrative account of the compiler changes stage by stage, followed by difference listings generated with a version control system, and test cases in the format of the existing tests for Lab 4, showing the compiler input, the expected output, and the assembly language code. Candidates are not required or permitted to submit the code of their implementation in machine-readable form.

> *Your attention is drawn to the University's policy on Plagiarism set out in Appendix A of each Course Handbook and on the University's website. The work submitted for this assignment should be entirely your own and not done in collusion with others. You may make use of written and online sources, but these should be acknowledged.*

The remainder of this document sets out the task that constitutes the Practical Assignment.

Simultaneous assignment

It is often convenient to specify simultaneous assignments, for example when swapping values of two variables:

```
x, y := y, x
```

The intended meaning of this is that the values of `x` and `y` are read before either is written. More generally one can specify any number of simultaneous assignments, where the effect should be the same as performing the assignments individually in a way that keeps them from interfering with one another. For variables $x_1, x_2, \ldots, x_n$ and expressions $e_1, e_2, \ldots, e_n$,

$$x_1, x_2, \ldots, x_n \texttt{ := } e_1, e_2, \ldots, e_n$$

could be implemented by evaluating all of $e_1, e_2, \ldots, e_n$ before assigning each resulting value to the corresponding $x_1, x_2, \ldots, x_n$. (*Variable* here means any expression that can represent a memory address.)

Your task will be to extend picoPascal to allow the user to write simultaneous assignments in this form. Individual assignments of the form x`:=`e should still work as before.

Notes

- Ensure that you have the latest versions of lab materials, as instructed on the previous page. You should extend the Lab 4 compiler with this new assignment construct. (You may incorporate your previous Lab 4 solutions for better use of addressing modes, but you don't need to explain them in your report for this task.)

- You may simplify your implementation by allowing only values with scalar types (as defined in `dict.ml`) to be assigned simultaneously. You will not be penalised for imposing this constraint, but if you do so you must say what complication it might avoid.

- As registers are limited in number and are also used in CSE (common subexpression elimination), your implementation is allowed to fail with an appropriate compile-time message if a simultaneous assignment causes the compiler to run out of registers. When there is little scope for CSE, your implementation should be able to handle simultaneous assignment of up to four variables.

- The intermediate code already provides DEFTEMP and TEMP instructions (as used in CSE) for introducing new temporary values.

- Individual assignments, as well as potentially changing other values on the right-hand side of a simultaneous assignment, might change some addresses on the left-hand side — for example

  ```
  i, a[i] := i+1, a[i]+1
  ```

  Therefore as well as all right-hand side expressions, some left-hand side expressions might need to be evaluated (as addresses) before any of the individual assignments are performed.

Assessment

Partial marks will be given for implementing only some aspects of the task, or generating code that is significantly inefficient or over-long. To gain full marks, your submission should include test cases that illustrate the quality of your implementation and the generated code. The following mark scheme will be used, giving a maximum total of 35 marks.

- A basic implementation of the task that allows simultaneous assignment to named variables, with simple test cases showing the code. *(10 marks)*
- An implementation that also allows simultaneous assignment to other expressions such as array elements and record fields, with simple test cases. *(10 marks)*
- Tidy object code of reasonable efficiency, demonstrated by test cases. *(5 marks)*
- Further test cases that demonstrate subtle aspects of the solution. *(5 marks)*
- A written report with good clarity and presentation. *(5 marks)*