Compilers Exercise: The MAlice Compiler

17 November 2010

Milestone 3

Summary

So far you have been working with a subset of the MAlice language. You will now use the lessons learned in the previous Milestones to implement a compiler for the full language.

Details

Further code examples have been released and are available now for download at `/vol/labs/secondyear/malice2010/examples/`.

You should implement a compiler which, given a valid program in the MAlice language, produces assembly code for the architecture of your choosing.

You will once again have to discern what constitutes a valid program from the samples given. You may have to reconsider the assumptions you made as to correct behaviour during the previous milestones. Think about the semantics of each new construct that you encounter in the provided examples.

As usual, the provided examples give you an idea on how the language is, i.e. which are its constructs. Your task is also to completely define the language from the examples, as in Milestone 1 you had to define a specification for a full programming language based operations on integers and characters.

Submit by: end of term

What To Do

1. Write a compiler for the MAlice programming language.

- You may find useful to extend the specification for the language of Milestone 1, or to define a brand new specification, to include the new constructs. Moreover, it would be useful also to specify an informal semantics for each new construct.
- Additionally, when designing and developing your compiler, you may find useful to go back to lecture notes and try to understand if any of the discussed optimisations could be applied to the new language.

2. Write a short report presenting (three pages maximum). The main goal of this exercise is to promote reflection on the experience. Assessment will be focussed on the professionalism and analysis that you are able to demonstrate. :

   - **The product**:Your analysis of your own critical evaluation of the quality of the compiler you built. You should consider both whether it meets the functional specification, and whether you judge that it forms a sound basis for further development. You may wish to address performance issues.
   - **The design choices**:Your analysis of the design choices that you made. What design choices *did* you make? Of particular importance is to evaluate the tools you chose to use (or to reflect on your decision not to use tools). What would you do differently?
   - **Beyond the specification**:Your evaluation of what you think are the most interesting direc- tions for extending the project. You may wish to focus on language extensions, optimisation, or perhaps other aspects. You may wish to cite evidence from preliminary implementation work.

Additional Help Getting Started

By looking at the examples provided with Milestone 3, you should recognise some very well-known constructs (even if in a nicer form), which you have encountered in previous courses and in other programming languages. Whenever you are giving semantics to the MAlice constructs, think about the semantics of similar constructs, how you use them in your programs, and how you expect them to behave.

As in Milestone 2, you are required to implement a full programming language from the examples: rely on your experience on how a programming language should be when working on Milestone 3.

Therefore:

- Consider the process you went through during Milestones 1 and 2.
- Think carefully about design - poorly thought out design will slow your development and make debugging harder.
- If you discover that some parts of the specifications from Milestone 1 or parts of the code from Milestone 2 do not fit anymore with the requirements of this Milestone, do not hesitate to apply strong modifications to your previous work, or to re-implement it.
- Implement iteratively, do not try to implement every feature in one session.

- Manage your time carefully. Do not leave everything until the final week.
- Additional and advanced help will be provided at the compilers tutorials until the end of term.

Submission

- Set up your group for Milestone 3 in CATE
- The group leader should then submit the file `malice_m3.zip` or `malice_m3.tar.gz` containing:
  - The files required to build your compiler,
  - A Makefile which builds the compiler, and
  - A command `'compile'` which runs your compiler.
  - A pdf file `report.pdf` for your report.
- You must test to ensure that your package will install and run without support from your own working environment

Assessment

This milestone will constitute 50% of the marks for the exercise.

Return of Work and Feedback

This milestone will be marked and returned by mid january. Feedback on your solution will be given on the returned copy.