Scalpa to MIPS

This is a Scalpa compiler that can translate code to MIPS assembly language.

It is a project part of the "Compilation" module made during our Master of computer science.
Link to the subject

Link to the project

Project Members
This project is made by:

- Hugo Brua
- Louis Politanski
- Maxime Princelle

Table of contents

Technologies used

The compiler is made with the following languages and/or technologies:

- Lex
- Yacc
- C

Environnement

In order to make sure that the environment is healthy and that everything runs smoothly we have created a Docker image with everything you need to compile and run this project.

To start the image with the project, you need to first install Docker on your machine, if it's not already there, otherwise, please install it.

> ⚠ If you want to compile and run the project directly on your machine, please note that we do not recommend using this method and therefore we cannot be held responsible.

To enter the environnemnt, simply run this script :
`./start_env.sh`

> ⏰ Wait a little bit (depending on your internet connection) while the image is downloading and the environment is started and ready. You will be hosted by a bash console on the project directory.

> 👉 Every modification made inside the environnement is automatically applied to the project on your host machine through a Docker volume mounted to the project directory.

Compile and run

To compile the project, a makefile file is provided.

There are two commands available:

- `make` : to build the project.
- `make clean` : to clean the directory

To run the compiler, the command is:

`./scalpa [-version] [-help] [-tos] [-tov] [-toa] [-o <out_file>] <in_file>`

Arguments (optional by default):

- `-version` / `-help` : display this help and exit.
- `-tos` : displays the table of symbols before exiting.
- `-tov` : displays the table of variables before exiting.
- `-toa` : displays the table of arrays before exiting.
- `-o <out_file>` : specify the output file for MIPS code, if not specified, defaults to <in_file>.s in the same directory as input file
- `<in_file>` (required) : path to the file containing the SCALPA code to be compiled into MIPS.

To run the generated MIPS code we provide an emulator.

Simply run the following command to start the emulator: `./exec.sh <mips_file>`

Argument (required):

- `<mips_file>` : path to the file containing the MIPS code to be executed by the emulator.

The emulator is made in java. Again, everything needed to run it is provided with the Docker environment.

Tests

We made tests for the compiler in order to check if every capabilities works and doesn't crash the emulator with the generated MIPS code and/or the compiler itself.

If you want to run those tests, simply run the provided script: `./test.sh`

This script will go through every SCALPA code files and first run the file in the compiler and then execute the generated MIPS code.

If either one of the two tests fail, the process will stop and mark an error.

There is also manual tests provided, for example when MIPS prompts for a value for the "read" function. Those are available in the folder: "tests-manual".

Capabilities

Here are the capabilities of the compiler:

- Handle the traduction of almost all the grammar of SCALPA.
- Includes automated and manuals tests.
- CI system that runs the tests and returns for each commit the result of those.
- Setup a specific output file.
- Display the table of symbols, variables and arrays.

Missing features

Those features are not handled by the compiler:

- functions (calls, declaration).
- referenced variables

Limitations

- We can only use arrays of maximum 7 dimensions because we only have access to 9 temporary variables at a time.
- We do not support "else" for condionnal expressions.
- Check that the values of the array are initalized before reading them.

  > For example, this code:
  >
  > ```
  > program mainOnly
  >    var val : array[1..3] of int;
  >    begin
  >        write val[1];
  >    end
  > ```
  >
  > should return a syntax error because it's missing: `val[1] := ?`.

Difficulties

Here we are going to share with you the difficulties that we encountered throughout the project.

Compiler:

- Handling of arrays:
  - Calculation of space required,
  - Calculation of the position i[1,1] with MIPS
  - Verify with MIPS if a given index is valid within a given ensemble (i.e. 1 is not in [-4..10]).

Project organisation:

- Learning MIPS took a lot of time.
- Working on the project with a small number of people who do not came from the UFR faculty did not facilitate the distribution of tasks.