Compiler construction (MRJP) lab project

Laboratory grade is based on a project involving the implementation of a compiler for a simple imperative language Latte.

Stages

The project consits of three stages:

1. Front-end: syntactic and semantic analysis - should be finished until Dec 9, but there is no separate moodle assignment for it.
2. LLVM or x86 back-end. Deadline Jan 7.
3. (optional) extensions. Deadline Jan 20.

Your lab supervisor can set slightly different deadline, in particular require project presentation in person. Nonetheless, all projects must be submitted to moodle.

Scoring

At most 24 points in total, counting towards final course mark. To pass the lab and qualify to sit the exam, it is required to submit satisfactory solutions for all stages and get a total of at least 15 points (including points for the Instant compiler).

Points are awarded for:

1. front-end (4);
2. back-end for LLVM (8) or **(exclusive)** x86 (10)
3. LLVM code in SSA form - additional 1p
4. register allocation for x86 - additional (up to) 4p
5. optimisations - up to 3p
6. Extension (x/y denotes points for LLVM/x86 respectively)
    1. arrays (2)
    2. structures (2)
    3. objects (attributes, methods, inheritance without method override) - (3/4) extra
    4. virtual methods (3/4) extra (in total 8/10 can be awarded for objects with virtual methods)
    5. garbage collection (2)

A more precise specification of extensions is contained in the language description.

Late submissions

Late submissions are subject to penalty of 2p for each week started; however no submissions will be accepted after February 7.

Rules

The project must be original work, developed independently by the student submitting it. In particular:

- you must not look at other students’ code, show or share your code in any manner.
- any foreign code (e.g. from books, tutorials, or other sources) must be clearly marked and attributed.

Technical requirements

1. The project must be submitted as a packed TAR archive (.tar.gz, .tgz, tar.bz2 or .tbz)
2. The project root must contain at least:
    - Text file **README** describing how to compile and run the project, used tools and libraries, implemented extensions, structure of the project, pointers to more detailed documentation.
    - **Makefile** allowing to build the program
    - A directory **src** containing only source files of the project (possibly including Latte.cf, makefiles etc.); auxiliary files such as libraries, etc should be placed in other directories.
3. The project must be buildable on lab computers by running **make** in the project root.
4. All necessary libraries (except standard library of the programming language used) must be described in README.
5. After the build, project root must contain an executable file **latc** (may be a shell script calling other programs)
6. The compiler must accept all test programs from the directory **good** and reject (with appropriate error messages) all programs from the directory **bad**. For the extensions, the compiler must accept all programs from the respective subdirectories in **extension**. Execution of a correct test program must give output exactly the same as in the corresponding **.output** file (given input in the corresponding **.input** file, if any).

7. For an accepted program, the compiler must output OK (`"OK\n"`) and exit with code 0.
8. For a rejected program, the first line of stderr must be ERROR (`"ERROR\n"`). Further lines should contain appropriate error messages. The compiler must then exit with a non-zero exit code.

LLVM backend

1. After running **make** the project root should contain executable **latc_llvm**
2. Running **latc_llvm foo/bar/baz.lat** for a correct program **baz.lat** should create files **baz.ll** (readable LLVM code) and executable **a.out** in the directory **foo/bar**.
3. Helper functions (**printInt** etc.) should be placed in the file **runtime.bc** in the **lib** directory (with sources in **runtime.ll**).

x86 backend

1. After running **make** the project root should contain executable **latc_ARCH** where ARCH is x86 or x86_64
2. Running **latc_ARCH foo/bar/baz.lat** for a correct program **baz.lat** should create files **baz.s** (assembly) and executtable **a.out** in the directory **foo/bar**.
3. Helper functions (**printInt** etc.) should be placed in the file **runtime.o** in the **lib** directory (with sources in **runtime.s**).

Test programs archive:

lattests121017.tgz, MD5:49ca5702ca9795fb8f33d52b0b3c6fc3