Project Compilers 2020

Mano Marichal & Joren Van Borm

Overview

We hebben alle mandatory dingen afgewerkt. Onderaan de readme kan je een overzicht zien van wat we allemaal gemaakt hebben. Voor elk feature hebben we een testfile, in test_IO/working_examples Je kan voor al deze de llvm ir en ast dot / png files genereren met run.py. clean.py Verwijdert alle gegenereerde files uit test_IO/working_examples.

Installing and running:

(assuming a linux-based system)

Clone git repository

First, open a terminal where you'd like to clone the repository, then run:

`git clone https://github.com/shano19/compilers-2020.git`

Then navigate to the repository with `cd ./compilers-2020`.

Install LLVM

```
sudo apt-get install llvm
```

Install pip

```
sudo apt-get install python3-pip
```

Install virtualenv using pip3

```
sudo pip3 install virtualenv
```

Create virtual environment

```
virtualenv venv
```

Active virtual environment:

`source venv/bin/activate`

When you're done using the compiler, deactivate it again using the `deactivate` command. (or just quit the terminal)

Install prerequisites:

`pip3 install -r requirements.txt`

Run the test files

`python3 run.py`

Some of these test files will print to stderr when warnings (or errors) are encountered. Some of them won't compile at all because they're testing error detection.

Compiling a file

`python3 ./src/main.py <filename>`

The `-cf` flag can be added after `<filename>` to enable constant folding.

Status:

Project 1)

- 2 Expression Parser
  - 2.1 Grammar:
    - [x] (mandatory) Binary operations + , - , * , and /
    - [x] (mandatory) Binary operations > , < , and ==
    - [x] (mandatory) Unary operators + and -
    - [x] (mandatory) Brackets to overwrite the order of operations
    - [x] (optional) Binary operator %
    - [x] (optional) Comparison operators >= , <= , and !=
    - [x] (optional) Logical operators && , || , and !
  - [x] 2.2 (mandatory) AST
  - [x] 2.3 (mandatory) Visualization
  - [x] 2.4 (optional) Constant folding

Project 2)

- 1 Variables:
  - 1.1 Grammar:
    - (mandatory) Types

            - [x] char
            - [x] int
            - [x] float
            - [x] pointer (no pointer arithmetic)
        - (mandatory) Reserved words
            - [x] const
            - [x] int
            - [x] float
            - [x] char
        - [x] (mandatory) Variables
        - [x] (mandatory) Pointer Operations * and &
        - [x] (optional) Identifier Operations ++ and –
        - [x] (optional) Implicit Conversions (+ warnings for non-promotions)
- [x] 1.2 (mandatory) AST
- [x] 1.3 (mandatory) Visualization
- [ ] 1.4 (optional) Constant Propagation
- 2 Error Analysis
    - [x] 2.1 Syntax Errors
    - [x] 2.2 Semantic Errors
        - [x] undefined & uninitialised variables
        - [x] redeclared & redefined variables
        - [x] operations on incompatible types (dereferencing a non-ptr type)
        - [x] Assignment to an rvalue
        - [x] Assignment to a const variable
        - [x] Symbol table (scoped)

Project 3)

- 1 Variables
    - 1.1 Grammar
        - [x] (mandatory) Comments
        - [x] (mandatory) printf() for char, int & float (without metastring)
    - [x] 1.2 (mandatory) AST
    - [x] 1.3 (mandatory) Visualization
    - 2 (mandatory) LLVM
        - [x] (mandatory) Binary operations + , - , * , and /
        - [x] (mandatory) Binary operations > , < , and ==
        - [x] (mandatory) Unary operators + and -
        - [x] (mandatory) Printf
        - [x] (mandatory) Pointers + pointer operators
        - [x] (optional) Identifier Operations ++ and –
        - [x] (optional) Comments for each machine instruction
        - [x] (optional) Comparison operators >= , <= , and !=

* [x] (optional) Logical operators && , || , and !
* [x] (optional) Conversions (bool <> char <> int <> float)
* [x] (optional) Binary operator %
* [ ] (optional) Include comments in compiled LLVM

Note: soms als je met floats werkt crashed het in assembly met een:

```
error: floating point constant invalid for type
```

dit komt wanneer je een floating point constant wil inladen dat een repeating decimal is in binary, bevoorbeeld

```
float a = 1.3;
```

Meer hierover op de LLVM documentatie: https://llvm.org/docs/LangRef.html#simple-constants

We hebben een mail gestuurt naar Brent of we hier rekening mee moesten houden, en hij zij van niet.

Remarks + extras

- We hebben een assignment operator
- We supporten operators * en & voor pointers, en pointers naar pointers naar pointers etc..

huge_test.c combineert zo een beetje alles, dus ik raad aan om deze zeker te bekijken.