St Petersburg University

Department of Software Engineering

Code generation for floatting-point arithmetic in architecture MIPS

Author: Ivan Arkhipov

28 May 2020

Introduction

- High variability of code generation in RISC and CISC architectures (as opposed to HLL)
- In the MIPS32 architecture floatting-point arithmetic differs from integer one
- In architecture MIPS32 Linux operating system interface for implementing printf is complex and often unpredictable

Work purpose

- Implementation of floatting-point arithmetic in the RuC translator using the request and response technique for the MIPS32 architecture based on the Russian Baikal-T1 processor
- Increased reliability of the RuC translator
  - The rejection of pointer arithmetic
  - Strict control of indexes
  - Thorough and detailed output of syntactic and dynamic errors

Tasks

- Implement code generation for operations with floatting-point numbers in the MIPS32 architecture
  - Unary operations
  - Binary operations
  - Logic operations
- Implement printing floatting-point numbers to the console
- Prepare tests and test the implemented code generation

Relevance

- The MIPS32 architecture is the basic architecture of one of the russian computers Baikal-T1
- Possibility of further development: optimizing viewing
- Working on industrial implementation

Existing analogues

- Code generator of a translator from the ALGOL 68 language for ES EVM
- Code generator gcc
- Code generator Clang

The technique of request and response

- Types of requests:
  - BREG – to a defined register
  - BREGF – to a defined register or constant
  - BF – free request
- Types of responses to a free request:
  - AREG – in register
  - AMEM – in memory
  - CONST – constant
- Example: res = (a + b) * c - d / e

Implementation

- Work with the module mipsgen
- Lexeme processing operation with floatting-point numbers
- Processing the printf call lexeme and the string lexeme

Unary operations

These operations include: =, +=, -=, *=, /=, if the left operand is a static variable, and also increment, postincrement, decrement, and postdecrement.

Binary operations

These operations include arithmetic operations with floatting-point numbers:+, -,*, / and =, +=, -=, *=, /=, if the left operand is an expression.

Logic operations

- These operations include logic operations with floatting-point numbers: ==, !=, <, >, >=, <=
- Unlike in similar operations with integers floating point operations change flag FP

Printing floatting-point numbers

- Loading of strings
- Loading of floatting-point arguments
- Call printf
- We used standard Linux functions, but we had to understand the subtleties

Testing

- RuC does not have automatic testing yet
- A separate test was prepared for each operation
- Tests with complex arithmetic expressions
- Pass the generated code files using the assembler and creating an executable file
- Running programs on the Baikal-T1

Results

- Code generation of operations with floatting-point numbers is implemented
- Printing of floatting-point numbers is implemented
- The generated code was successfully run on Baikal-T1