CS323 Project Phrase 4 Report

12112421 王德涵 12112620 徐霄阳 12112822 孔繁初

**Assumptions**

As described in the requirement document, our implementation is based on the following assumptions:

**Assumption 1** The intermediate code are logically correct, meaning that you can run

them on the IR simulator and obtain the correct output.

**Assumption 2** There are no structure or array variables, so you don’t need to translate

the DEC instruction.

**Assumption 3** All integer constants are in the range so that they can be safely represented by MIPS32 immediates.

**Translation Schemes**

**Register Allocation**

We employ a method similar to the way cache operates for register allocation. Specifically, the following steps are taken whenever a variable needs to be used:

1. If the variable is in a register, get its value directly from the register.
2. If the variable is not in a register, and there is an available space in the registers, allocate a register to the variable.
3. If the variable is not in a register, and there are no available spaces in the registers, find the least recently used variable in the registers, store it back to memory, and allocate the register to the requested variable.

However, due to the difference between the MIPS code generation order and the actual execution order, extra handling is required for statements like if statements that introduce branching. When encountering an if statement, we store back into memory all variables that may be used in the registers shortly after, to mitigate potential issues.

**TAC Translation**

We didn’t use the provided starter code. Instead, we implemented an end-to-end compiler by directly adapting the compiler front-end to the target code generator. We translate

**Extra Test Cases**

In out extra test cases, **test case a** includes calling a function that requires the caller to pass up to 32 parameters, showing that the generated code can correctly pass more than 4 parameters. **Test case b** includes a relatively complicated calling stack, showing that the generated code can handle local variables in different running environments properly. **Test case c** includes calculating using exponentiation by squaring algorithm, and the result can be calculated correctly.