Lab 5 Part 1: Escape Analysis and Translation

Description

In this lab, you're going to do the escape analysis and translation of tiger compiler, you're supposed to ensure your escape analysis right and your translation could at least print out a complete IR tree.

Related files for this lab are:

- **src/tiger/frame/.*** Files related to function stack frame(chapter 6)
- **src/tiger/translate/.*** Files related to IR tree translation(chapter 7)
- **src/tiger/runtime/runtime.c** Tiger program runtime file(will be linked with code files generated by your compiler)
- **src/tiger/env/env.*** The EnvEntry classes which help compiler store the information of variables&functions
- **src/tiger/escape/escape.*** Files related to escape analysis(chapter 6)

To finish this lab, you will only need to finish the following modules: { x64 stack frame } { IR tree translation} { escape analysis } you can modify any file to finish your design.

Notice: Before you start this lab, you should carefully read the chapter 6,7,12 of the textbook. And if you have any question about this lab, feel free to contact Jinze Si, who is the teaching assistant responsible for lab5 part1.

Important Notes

1. Before you start to implement IR tree translation, carefully read the chapter 6&7 of the textbook.

2. The file runtime.c is a C-language file containing several external functions useful to your Tiger program. These are generally reached by externalCall from code generated by your compiler.

3. Carefully read the chapter 12, which gives some useful interface declarations you may use in module *frame*.

4. You can only use one register(%rsp) as stack pointer, **%rbp is not allowed to be used as stack pointer.**

5.This lab is very difficult, **please leave yourself enough time to finish this lab.** Although we don't check if your IR tree is correct in this lab, but it's very important in part 2, we strongly suggest you implement it as robust and correct as possible to save your work in part 2, which is also difficult and compicated.

Environment

You will use the same code framework that you had set up when you worked on lab4 and use the code you written in previous labs. What you need to do now is to fetch the latest update and you may have to do some code merging jobs. If you have any difficulties in merging codes, you can ask TAs in our Wechat group.

```
shell% git checkout -b lab5-part1 upstream/lab5-part1
shell% git push -u origin
shell% git merge lab4

You may have to do some code merging jobs here


After merging, you are supposed to push your update to your remote repo
shell% git add files
shell% git commit -m "[lab5-1] merge lab4"
shell% git push origin lab5-part1

Note that in this lab you must carefully merge and make sure you get full score for pervious lab, unless you may not get full score for this lab.
```

If you haven't set it up before, you should follow the instructions here to set up your lab environment.

Grade Test

The lab environment contains a grading script named as **grade.sh**, you can use it to evaluate your code, and that's how we grade your code, too. If you pass all the tests, the script will print a successful hint, otherwise, it will output some error messages. You can execute the script with the following commands.

```
Remember grading your lab under docker or unix shell! Never run these commands under windows cmd.
shell% make gradelab5-1
shell% ...
shell% [^_^]: Pass #If you pass all the tests, you will see these messages.
shell% TOTAL SCORE: 100
```

Handin

The deadline of this lab is on Tuesday 12:00 AT NOON, Nov 21, 2023, and if you miss the deadline, points will be deducted based on the number of days you are late!

```
shell% git push origin lab5-part1
```

Go to Top // *Compilers* Home Page
Questions or comments regarding *Compilers* course? Send e-mail to the course Staffs or TAs.
Last updated: Tue Nov. 9 2021