322 Compilers: Assignment 4
Linearization

Your Job

Design a series of transformations that linearize `tree-stm` and `tree-exp` expressions, build basic blocks, and then schedule the basic blocks. Break up your transformations into (at least) these functions:

linearize-stm : tree-stm ⟶ lin-stm list
linearize-exp : tree-exp ⟶ lin-stm list and a pure-exp

basic-blocks : lin-stm list ⟶ bb list and a label

trace-sch : bb list and a label ⟶ lin-stm list
clean-sch : lin-stm list ⟶ lin-stm list

Linearized Statments & Expresions

Linearized statements are a special form of the *tree-stm* that do not contain nested statements and basic blocks are a special form of a list of linearized statements that begin with a label and end with a jump (and have no jumps or labels in between).

lin-stm	←	*mc-stm* / *jump-stm* / *label*
mc-stm	←	(`move` *temp pure-exp*)
	/	(`move` *temp* (`call fn` *pure-exp*))
	/	(`move` (`mem` *pure-exp*) *pure-exp*)
	/	(`texp` (`call fn` *pure-exp* ⋯))
jump-stm	←	(`jump` *pure-exp label* ⋯)
	/	(`cjump relop` *pure-exp pure-exp label label*)
pure-exp	←	*num*
	/	*label*
	/	*temp*
	/	(`biop` *pure-exp pure-exp*)
	/	(`mem` *pure-exp*)
bb	←	(*label mc-stm* ⋯ *jump-stm*)

Advice

Even thought the test fests will run with Tiger input programs, it is best to test each of these functions directly on tree expressions. Build test suites for each one of them, using `evalil` to make sure that the output of each function behaves like the input does.

Submission Instructions

Submit a single `zip` file containing your test cases in a directory called `4a` and your code in a directory called `4b`. The `4b` directory should contain a script called `lin` that accepts a filename on the command-line and then prints out a sequence of `lin-stm`s (wrapped with parentheses) corresponding to the Tiger program in the file.

For example, if the input file contains

```
if 1<2 then 3 else 4
```

the output might look like this:[1]

```
((cjump < 1 2 l:if-false0 l:if-true0)
 l:if-false0
 (move r:if-result0 4)
 (jump l:if-afterwards0 l:if-afterwards0)
 l:if-true0
 (move r:if-result0 3)
 l:if-afterwards0
 (move r:linearize-call0
       (call "printint" r:if-result0))
 l:end-of-basic-blocks0)
```

The filenames must follow the same conventions as in assignment 3.

Your `zip` file should also contain subdirectories `1a`, `1b`, `2a`, `2b`, `3a`, and `3b` containing either your submissions from last time, or fixed versions of them. The revised implementations (but not the revised test cases) will be used when we re-run the parsing and typechecking test fests.

[1] It does not have to look like this exactly and, in fact, my compiler only produces something vaguely similar. It does, however, have to behave like this one does.