

## 322 Compilers Assignment: spill-test

### Register allocation: spill testing

**Your job:** Design (at least) 15 test cases for the spill functions as a pair of files: input and expected output. Note that the following describes the spill function (so you can write tests for it), but the spill function itself is not due until next week. This is the signature for the spill function:

```
spill : (label nat nat i ...) var var -> (label nat nat i ...)
```

The `spill` function accepts an L2 function, the name of a variable to be spilled, and a prefix for the names of the spill variables. It returns a new program that spills the variable to the next location in the stack (based on the current number of spills listed in the function). You can assume that no variable in the program begins with the spill variable prefix. You must use that prefix for any variable you spill. The first instruction with an occurrence of variable to be spilled should use a new variable that begins with the prefix and is followed by the number 0, the second instruction with an occurrence of the variable to be spilled should have the same prefix followed by the number 1, etc.

The `spill` function should be wrapped up into a script that accepts a filename naming a file that contains the arguments in the file. The script should write its answers to stdout.

For example, if the file `f.L2f` contains:

```
(:f 0 0 (x <- 1) (rax += x)) x s_
```

Then this transcript would show how your script should behave:

```
% spill f.L2f
(:f 0 1
 (mem rsp 0) <- 1)
(s_0 <- (mem rsp 0))
(rax += s_0))
```

In general, use exactly one new temporary per instruction that uses a spill variable (no more and no less). For example, if `g.L2f` contains:

```
(:g 0 0 (x <- x < x)) x s
```

then this is the correct behavior:

```
% spill g.L2f
(:g 0 1
 (s0 <- (mem rsp 0))
 (s0 <- s0 < s0)
 (mem rsp 0) <- s0))
```

One special-case: when your spilling function is asked to spill `x` in the instruction `(x <- x)`, just drop the instruction completely instead of adding in memory accesses. One warning: when you see an instruction `(rax <- x)` and you're being asked to spill `x` to `(mem rsp 8)`, just generate `(rax <- (mem rbp 8))`, don't generate something like `(s0 <- (mem rsp 8)) (rax <- s0)`. Any instructions that do not mention the spill variable should be left alone. Note that there is no checking that the function is a part of a valid L2 program. It only has to conform to the grammar, not be free of errors at runtime (so there might be use of variables before initialization, for example).

The test case input files should use the suffix `.L2f` and the files with the correct answers should use the suffix `.sres`.

Hand in your assignment by uploading it to the server at <http://penghu.eecs.northwestern.edu:8123/>. The uploaded file should be a gzipped tar file named `name.spill-test.tar.gz`. The `name` should be your family name in all lowercase letters (except for the names He, Liu, or Wang, see below) unless you are pair programming, in which case it should be both family names in alphabetical order, separated by `+`. If your name has any non-alphabetic characters, remove them first. For example, if Conan O'Brien and Shawn Knowles-Carter were pair programming and handing in this assignment, they'd

send in a tarfile named `obrien+kowlescarter.spill-test.tar.gz`. If your family name is He, Liu, or Wang, then include your first name as well, but also without any spaces. For example, if your name is Liu Bolin, then use `bolinliu` as your name. And if Bolin and and Shawn team up, they'd submit `bolinliu+kowlescarter.spill-test.tar.gz`.

The file must contain a single directory named `spill-test` containing the test cases.