A graph-coloring register allocator structure

Register allocator

Code analysis

Graph coloring

Spill

f with var spilled

f without variables and with registers

Liveness analysis

IN, OUT

Interferences analysis

Interference graph

spill(f, var, prefix)
Spilling

• Procedure used by a register allocator with the following inputs
  • A function $f$
  • A variable $v$ that needs to be allocated to the stack (to the vars section)
  • A string (see later)

• This procedure modifies $f$ to allocate $v$ on the stack
  • Make a new location on the stack
  • Replace all writes to $v$ with stores to the new stack location
  • Replace all reads from $v$ with reads from the new stack location
Spilling example

2 registers are needed

Only 1 register is now needed!

All L2 instructions can use variables, but only some L1 instructions can access a memory location!
Spilling example (2)

For every instruction that uses the spilled variable:
• Create a new variable that starts with %S and ends with a new number
• Replace the original instruction using the new variable
• Add loads/stores around the new instruction

spillForL1(@myF, %a, %S)
Spilling example (2)

What if we have only 1 register?

```plaintext
(@myF
  0
  %a <- 42
  %b <- 40
  %b += 2
  %a *= %a
  return )

spillForL1(@myF, %a, %S)

(@myF
  0 1
  %S0 <- 42
  mem rsp 0 <- %S0
  %b <- 40
  %b += 2
  %S1 <- mem rsp 0
  %S1 *= %S1
  mem rsp 0 <- %S1
  return )
```
Testing your spiller for homework #2

• Under L2/tests/spill there are the tests you have to pass

• To test:
  • To check all tests: make test_spill
  • To check one test: ./spill tests/spill/test1.L2f

• Check out each input/output for each test if you have doubts
  • tests/spill/test1.L2f
  • tests/spill/test1.L2f.out
Always have faith in your ability

Success will come your way eventually

Best of luck!