Simone Campanoni
simonec@eeecs.northwestern.edu
A graph-coloring register allocator structure

Register allocator
- Code analysis
- Graph coloring
- Spill

Liveness analysis
- IN, OUT

Interferences analysis
- Interference graph

Spill without variables and with registers

Spill(f, var, prefix)

f with var spilled
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!

Only some L1 instructions can access a memory location!
Spilling example (2)

\[
\begin{align*}
(:\text{myF} & \ 0 \\
%a & \leftarrow 42 \\
%a & +\ = %a \\
\text{return} & \)
\end{align*}
\]

\[
\begin{align*}
\text{spillForL1}(:\text{myF}, %a, %S) \\
\begin{align*}
(:\text{myF} & \ 0 \ 1 \\
%S0 & \leftarrow 42 \\
\text{mem rsp 0} & \leftarrow %S0 \\
%S1 & \leftarrow \text{mem rsp 0} \\
%S1 & +\ = %S1 \\
\text{mem rsp 0} & \leftarrow %S1 \\
\text{return} & \)
\end{align*}
\end{align*}
\]
Spilling example (2)

```plaintext
(:myF
  0
  %a <- 42
  %a += %a
  %a *= %a
 return )
```

```plaintext
(:myF
  0 1
  %S0 <- 42
  mem rsp 0 <- %S0
  %S1 <- mem rsp 0
  %S1 += %S1
  mem rsp 0 <- %S1
  %S2 <- mem rsp 0
  %S2 *= %S2
  mem rsp 0 <- %S2
 return )
```

spillForL1(:myF, %a, %S)
Testing your 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