**Accessing and Storing Variables**

**Local Variables**

Review instructions iload and istore from [JVM Spec](http://lara.epfl.ch/w/compilation:jvm_spec).

Assigning indices (called *slots*) to local variables:

* in which [compiler phase](http://lara.epfl.ch/w/cc09:phases_of_a_compiler). What information about the symbol do we need to know?
* how to compute the indices?

Function $slot : Var \to Int$

* maps variable *ocurrence* into an index (need the symbol table)

**Translation Rules for Local Variables**

If $x$is local variable and we translate expression, then   
\begin{displaymath}
   [\![ x ]\!] = \textbf{iload}(slot(x))
\end{displaymath}  
For assignment to local variable $x=e$, we have   
\begin{displaymath}
  [\![x=e]\!] = [\![e]\!] ::: \textbf{istore}(slot(x))
\end{displaymath}

**Static Variables**

Consult [JVM Spec](http://lara.epfl.ch/w/compilation:jvm_spec) and see

* getstatic
* putstatic

as well as the section on Runtime Constant Pool.

**Instance Variables (Fields)**

To access these we use

* getfield
* putfield

Note: in a lower-level model x.f is often evaluated as mem[x + offset(f)]

* with inheritance, offset(f) must be consistent despite the fact that we do not know exactly what the run-time class of x will be
  + this can be very tricky to implement efficiently with multiple inheritance
  + general approach: do a run-time test on the dynamic type of an object to decide how to interpret field or method access