Plan for today

Translating program features to the Tree IR

Last time
- arithmetic and some binary operations
- assignments
- accesses to array variables
- array assignment

This time:
- functions in the MiniJava run-time library
- allocating class instances
- allocating arrays
- length expression
- this expression
- call expressions
- less than operator
- if and while statements

Functions in MiniJava runtime library

Assume the following will be available
- void printInt( int x )
  - Prints given integer followed by a newline
- void * halloc( int n )
  - Allocates n consecutive bytes in the heap and returns the address of the first byte allocated.
- void initArray( int [], int n)
  - Assumes that a points at an array allocation of size (n+1) * wordsize
  - Puts the length n into a[0]
  - Initializes a[1] through a[n] to zero

Calling the functions within Tree IR
- Exp initExp = frame.externalCall("initArray", arglist);
- Frame.externalCall places an "_" in front of all function names

Allocating class instances

class Foo { ... }
... new Foo ...

Allocating an array

int [] y;
y = new int [20];
**Length expression**

```java
double y[] = new double[y.length];
```

**Call expression**

```java
... this . otherFunc(y,f)...
```

**StmCJUMP**

```java
public class StmCJUMP extends Stm {
    public int relop;
    public Exp left, right;
    public ExpNAME(left, Exp r, Label t, Label f) {
    ...
    }

eval(relop, left, right);
    evalExp(left, right);
    evalExpMEM(left, right);
    evalExpCALL(left, right);
    evalStmMOVE(left, right);
    evalStmEXP(left, right);
    evalStmJUMP(left, right);
    evalStmLABEL(left, right);
}
```

**Tree intermediate representation**

- `ExpCONST(int i)` - The integer constant i
- `ExpNAME(Label n)` - Constant address. Corresponds to label in assembly.
- `ExpTEMP(Temp t)` - Temporary t. “Infinite” Temps in Tree IR.
- `ExpMEM(Exp exp)` - If left child of move, then store into address calculated by exp. Otherwise, fetch value at address calculated by exp.
- `ExpCALL(Exp func, List<Exp> args)` - evaluate func to find func address, then evaluate args left to right.
- `ExpEXP(Exp e)` - Eval e and ignore result
- `ExpMOVE(Temp t, e)` - Eval e and put result in t.
- `ExpMEM(ExpMEM(e1), e2)` - Eval e2 and store into address e1.
- `ExpLABEL(Label l)` - Label in assembly.