An example `X86Target` `emit` method

```
 Location emitIntConstant(int value) {
    Location result_location =
       allocateReg(ILType.intILType());
    emitOp("movl",
           intOperand(value),
           regOperand(result_location));
    return result_location;
 }
```

`Location allocateReg(ILType):`
allocate a new register to hold a value of the given type

`void emitOp(String opname, String arg1, ...):`
emit assembly code

`String intOperand(int):`
return the asm syntax for an int constant operand

`String regOperand(Location):`
return the asm syntax for a reference to a register

An example `X86Target` `emit` method

What x86 code to generate for *`arg1`* `+.int` *`arg2`*?

x86 int add instruction: `addl %arg, %dest`

- semantics: `%dest = %dest + %arg;`

emit *`arg1`* into register `%arg1`
emit *`arg2`* into register `%arg2`
then?

An example `X86Target` `emit` method

```
 Location emitIntAdd(ILExpr arg1, ILExpr arg2) {
    Location arg1_location = arg1.codegen(this);
    Location arg2_location = arg2.codegen(this);

    emitOp("addl",
           regOperand(arg2_location),
           regOperand(arg1_location));

    deallocateReg(arg2_location);
    return arg1_location;
 }
```

`void deallocateReg(Location):`
deallocate register,
make available for use by later instructions

An example `X86Target` `emit` method

What x86 code to generate for *`var`* read or assignment?

Need to access var’s home stack location

x86 stack reference operand: `%ebp(offset)`

- semantics: `*(%ebp + offset);`
- `%ebp` = frame pointer

An example `X86Target` `emit` method

```
Location emitVarRead(ILVarDecl var) {
   int var_offset = var.getByteOffset(this);
   ILType var_type = var.getType();
   Location result_location =
      allocateReg(var_type);
   emitOp("movl",
          ptrOffsetOperand(FP, var_offset),
          regOperand(result_location));
   return result_location;
}
void emitVarAssign(ILVarDecl var,
                   Location rhs_location) {
   int var_offset = var.getByteOffset(this);
   emitOp("movl",
          regOperand(rhs_location),
          ptrOffsetOperand(FP, var_offset));
}
```

`String ptrOffsetOperand(Location, int):`
return the asm syntax for a reference to a "ptr + offset" memory location

An example `X86Target` `emit` method

```
void emitAssign(ILAssignableExpr lhs,
                ILExpr rhs) {
   Location rhs_location = rhs.codegen(this);
   lhs.codegenAssign(rhs_location, this);
   deallocateReg(rhs_location);
}
```

Each `ILAssignableExpr` implements `codegenAssign`

- invokes appropriate `emitAssign` operation, e.g. `emitVarAssign`