Recitation 6: Object-Oriented Code Generation*

* Low-level IR

Yotam Feldman

Based on materials by Yannis Smaragdakis and slides by Guy Golan-Gueta
• Valid programs (ASTs) compile to an LLVM program that’s
  – valid,
  – executes,
  – has the same input-output and external behavior (console output)
• Rules for valid MiniJava ASTs:
  [link](https://www.cs.tau.ac.il/research/yotam.feldman/courses/wcc20/semantic.html)
Arrays

• Allocation
• Access
• Assignment
• Dynamic checks
  – “ArrayIndexOutOfBoundsException”

• Also: array length (exercise 😊)
Objects

class A{
    int x;
    int y;
    ...
}

Runtime memory layout for object of class A
Field Selection

\[ \text{dynType}(f) = A \]

\[ q = f.x; \]

access \( f \)

access \( f.8 \)

Runtime memory layout for object of class A

Field offsets

vtable ptr

x

y

Compile time information for A class type

vtable ptr

x 8

y 12
class A{
    int x;
    int y;
    ...
}
class B extends A {
    int z;
    ...
}  

**Fields and Inheritance**

- **Runtime memory layout** for object of class B:
  - vtable ptr
  - x
  - y
  - z

- **Field offsets** for B class type:
  - vtable ptr
  - x: 8
  - y: 12
  - z: 16

- **Compile time** information for B class type:
  - Prefix of A

- **Compile time** information for B class type:
  - Prefix of A
Field selection

\[ \text{dynType}(f) \leq A \]

\[ q = f.x; \]

Runtime memory layout for object of class A

Methods (including inherited!) find \( x \) in the same place
Virtual Methods

class A {
    int x;
    int y;
    void f() {...}
    void g() {...}
}

define i32 @A.f(i8* %this)
    ...
#define i32 @A.g(i8* %this)
    ...

Runtime memory layout for object of class A

Compile time information for A class type
Methods and Inheritance

class A {
    int x;
    int y;
    void f() {...}
    void g() {...}
}
class B extends A {
    int z;
    void f() {...}
    void h() {...}
}
Object Creation

- Store vtable (e.g. global segment) with pointers to methods’ code
- Allocate memory for vtable pointer + fields
- Set vtable pointer to the correct table

**Runtime memory layout** for object of class A
Now in LLVM!

Demo
LLVM Further Notice

- `this` implicit parameter
- Allocating method formal parameters on the stack
Summary

• arrays
• fields
• virtual method calls and vtable
• Ex. 2