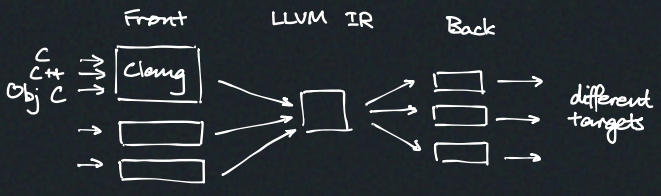
Lec 21 Project and LLVM

Lab 6 — LLVM as target
Extend language feature
co with garbage collection
choose your own adventure

LLVM

Collection of modular compiler toolchain technologies
Originally Low Level Virtual Machine for dynamic compilation

- LLVM Core — optim, LLVM IR
- Clang
- libcc +
- LLDB
- dragonegg



Ex. see slide

- Instrs have many options
- Predecessor labels in IR (the CFG)
- In SSA
- Has type info

Types : i32
[4 x i32]* ← ptr to arr of size 4
i32 (i32)* ← ptr to func taking in i32 and returning i32
{float, i32 (i32)* } ← struct containing stuff

Memory approach:

- allocate memory for vars in the source so that they can be written to (so still be SSA if other stuff in SSA)
- use mem2reg to turn some of the mem to stack

Hierarchy

- Module
  - Function — ...
    - Basic block — ...

Memory Management

So far... we only allocate
How to free ?

→ Manual C
→ Semi-auto Rust
→ Auto Java, SML

Good for programmer, but has mem & time overhead

Approaches :
1. Garbage collection
2. Reference counting
   - Each object in mem has counter of ptrs to it
   - Problems
     - have to propagate count
     - pointer scope handling
     - cycles