Lab 6: Control Flow Graph & Call Graph

Software Testing 2021

2021/04/22

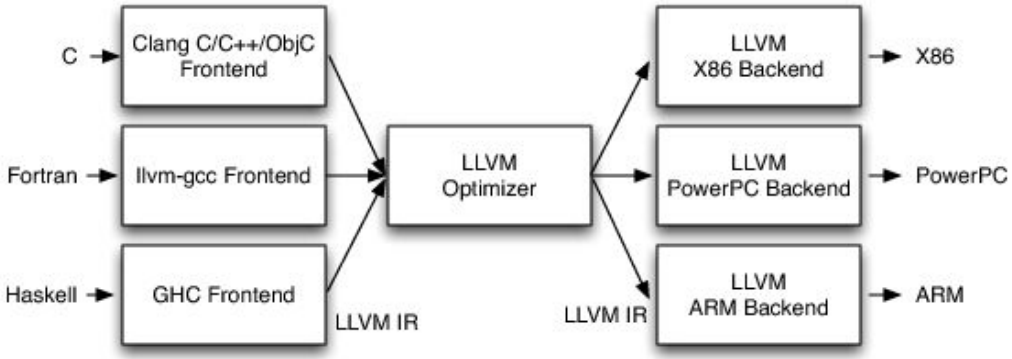
LLVM

Three-Phase Compiler

- Frontend, Optimizer, Backend
  - 解析、優化、輸出

LLVM

- [The Architecture of Open Source Applications: LLVM]



Install

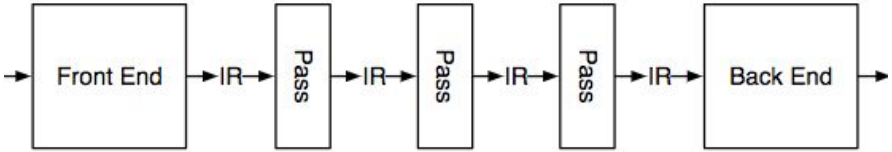
- LLVM
  - bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
  - sudo apt install llvm

- Dot
  - sudo apt install graphviz
    - dot -Tpng callgraph.dot -o callgraph.png

Demo

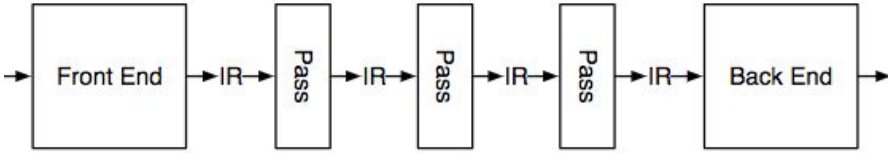
Frontend

- $ clang -S -emit-llvm ./lab_6.c
  - As LLVM bitcode format (LLVM IR)



Intermediate Representation

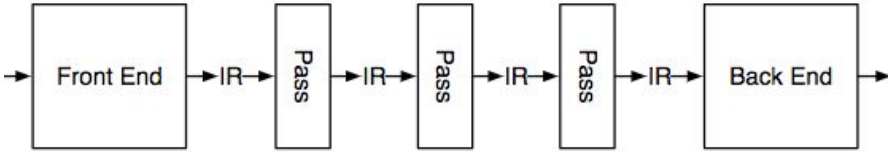
- $ lli lab_6.ll
  - Interpreter
  - Execute LLVM bitcode
- $ opt -S -mem2reg lab_6.ll
  - Optimizer
  - LLVM Pass(es)



- 從 LLVM IR 來看編譯器最佳化都在做些什麼
- 編譯器 LLVM 淺淺玩

Backend

- $ llc lab_6.ll
  - Assembly language



Others

- $ as lab_6.s -o lab_6.o
  - Assembler
- $ ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 lab_6.o /usr/lib/x86_64-linux-gnu/crt1.o /usr/lib/x86_64-linux-gnu/crti.o /usr/lib/x86_64-linux-gnu/crtn.o -lc -o lab_6
  - Linker

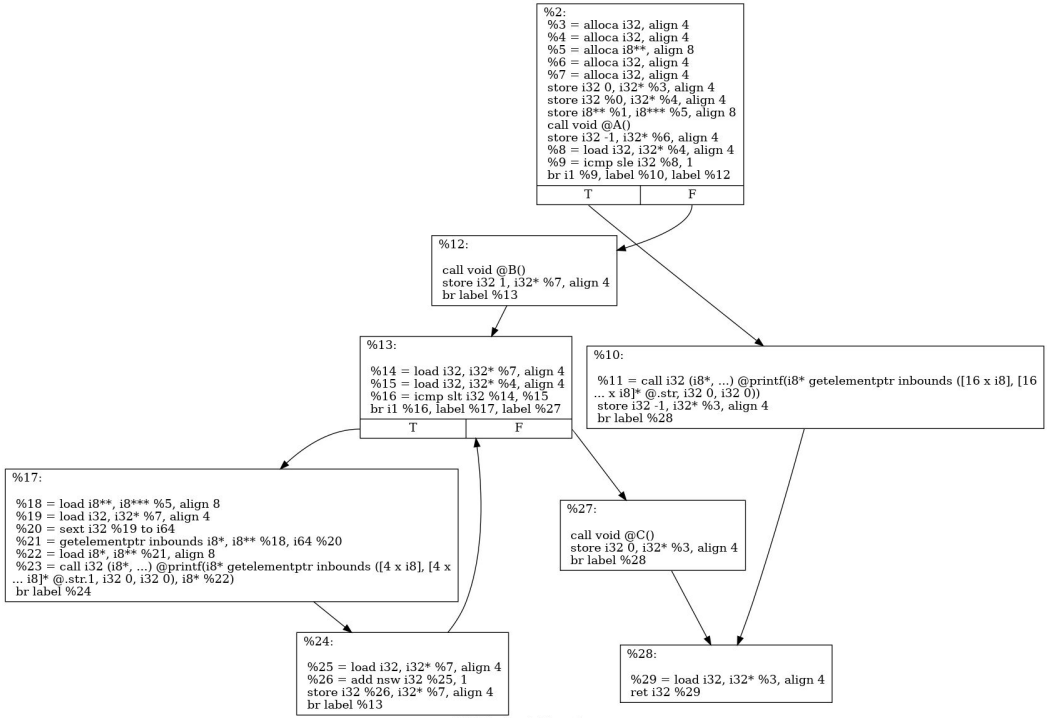
Lab

Lab

1. Download **Lab_6.c** from Github.
2. Generate call graph & control flow graph from the program above.
3. Deliverables shall include the following:
   a. Call graph:
      i. *ID_callgraph.png*
   b. Control flow graph:
      i. *ID_cfg.main.png*
      ii. *ID_cfg.A.png*
      iii. *ID_cfg.B.png*
      iv. *ID_cfg.C.png*

Do not compress the files and plagiarism.

Control Flow Graph



CFG for 'main' function

CFG for 'A' function

CFG for 'B' function

CFG for 'C' function

Call Graph

- LLVM-10
- LLVM-12