LLVM: Libraries & Tools

Max Thrun - Spring 2014

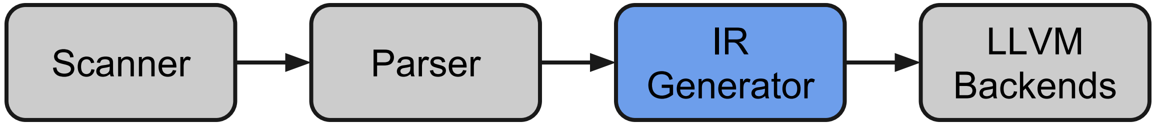
LLVM

- A collection of well-integrated libraries
  - Analyses, optimizations, code generators, JIT compiler, garbage collection support, profiling…
- A collection of tools built from the libraries
  - Assemblers, automatic debugger, linker, code generator, compiler driver, modular optimizer...

http://llvm.org/pubs/2004-09-22-LCPCLLVMTutorial.pdf

Custom IR Code Generator

- Can be a standalone C++/Python program
- Uses LLVM API to describe IR at high level
- Takes care of casting and other details



http://llvm.org/docs/ProgrammersManual.html

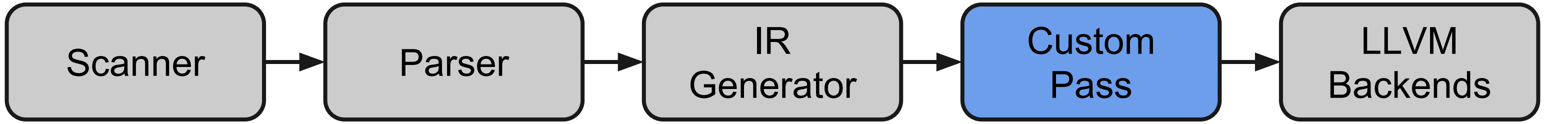
http://www.llvmpy.org/

Custom IR Code Generator

Demo

Custom Optimization Pass

- Compiles to a shared library (.so)
- Use `opt` tool to run it against .ll or .bc file
- Profile pass with `-time-passes` flag



http://llvm.org/docs/WritingAnLLVMPass.html

Custom Optimization Pass

- 7 Different types of pass classes:
    - BasicBlock
    - Region
    - Loop
    - Function
    - CallGraphSCC
    - Module
- http://llvm.org/docs/WritingAnLLVMPass.html

Optimization Pass Classes

- BasicBlockPass
  - Cannot modify or inspect other basic blocks
  - Cannot maintain state between basic blocks
  - Cannot modify control flow graph
- RegionPass
  - Executes on each single entry single exit regions in all functions
  - Inner-most region processed first

What is a region?

```
>>  CFG:   1
>>        / |
>>       2   |
>>      / \   3
>>     4   5  |
>>     |   |  |
>>     6   7  8
>>      \  | /
>>       \ |/       region A: 1 -> 9 {1,2,3,4,5,6,7,8}
>>         9        region B: 2 -> 9 {2,4,5,6,7}
```

http://lists.cs.uiuc.edu/pipermail/llvmdev/2010-March/029952.html

Optimization Pass Classes

- LoopPass
  - Executes on each loop in the function
  - Independent from other loops
  - Inner-most loop processed first
  - Example: llvm/lib/Transforms/Utils/LoopUnroll.cpp
- FunctionPass
  - Cannot inspect or modify other functions
  - Cannot add or remove functions
  - Cannot add or remove global variables

Optimization Pass Classes

- FunctionPass
  - Cannot inspect or modify other functions
  - Cannot add or remove functions
  - Cannot add or remove global variables
  - Cannot maintain state across functions
  - Example: `llvm/lib/Transforms/Scalar/ConstantProp.cpp`

Optimization Pass Classes

- CallGraphSCCPass
  - Used by passes needing to traverse call graph bottom up
  - Operate on the call-graph in SCC (strongly connected components) order
    - http://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm
  - Example: llvm/lib/Transforms/IPO/Inliner.cpp

Optimization Pass Classes

- ModulePass
    - Most general of all the classes, can operate on everything
    - LLVM cannot optimize execution since it is totally unknown what this pass might do
    - Example: llvm/lib/Transforms/Utils/MetaRenamer.cpp

Most common type?

```
$ n=("BasicBlock" "Region" "Loop" "Function"
"CallGraphSCC" "Module"); for i in "${n[@]}"; do
num=$(grep -lR "public ${i}Pass" --include=\*.cpp
/tmp/llvm/lib/Transforms | wc -l); echo "$num $i"; done |
sort -nr
```

```
43 Function

19 Module

11 Loop

3 CallGraphSCC

3 BasicBlock

1 Region
```

Custom Optimization Pass

Demo

Demo Code

https://github.com/bear24rw/EECE6083_Presentation

Google Doc

https://docs.google.com/presentation/d/1HrbLaFtovhrgthdHx7I1aZqasUAxoa-m1n0L2NNy21k/edit?usp=sharing