Writing LLVM Optimization

SWPP

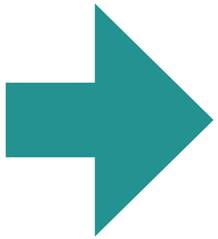
Apr. 1th

Juneyoung Lee

Simple Optimization: Constant Folding

- I'll explain my-opt.cpp that does constant folding.

```
define i32 @constant_fold() {
  %a = add i32 1, 2
  %b = sub i32 %a, 1
  ret i32 %b
}
```



```
define i32 @constant_fold() {
 ret i32 2
}
```

Writing FileCheck Tests

- Syntactic check!
- `opt -passes="my-opt" test.ll -S -o result.ll`
- `my-llvm-releaseassert/bin/FileCheck test.ll < result.ll`

test.ll

```
define i32 @negated_operand(i32 %x) {
; CHECK-LABEL: @negated_operand(
; CHECK-NEXT:    ret i32 0
  %negx = sub i32 0, %x
  %r = add i32 %negx, %x
  ret i32 %r
}
```

Writing FileCheck Tests

- Assigning variables:

```
define i32 @f(i32 %x, i32 %y) {
; CHECK-LABEL: @f(
; CHECK-NEXT:    [[Z:%.*]] = add i32 %x, %y
; CHECK-NEXT:    ret i32 [[Z]]
  %z =  add i32 %x, %y
  %z2 = add i32 %z, 0
  ret i32 %z2
}
```

- Other commands: CHECK, CHECK-NOT, ...
- Link: https://llvm.org/docs/CommandGuide/FileCheck.html

Testing Correctness of Opt. Using Alive2

- Online: https://alive2.llvm.org
- Offline: https://github.com/aliveToolkit/alive2
  - alive-tv src.ll tgt.ll
  - Building it requires Alive2-tailored LLVM build
  - You can use llvmscript/examples/llvm-alive2.json

```
1
2   ----------------------------------------
3   define i32 @src(i32 %a) {
4   %0:
5     %x = add i32 %a, 1
6     %y = add i32 %x, 2
7     ret i32 %y
8   }
9   =>
10  define i32 @tgt(i32 %a) {
11  %0:
12    %x = add i32 %a, 3
13    ret i32 %x
14  }
15  Transformation seems to be correct!
```