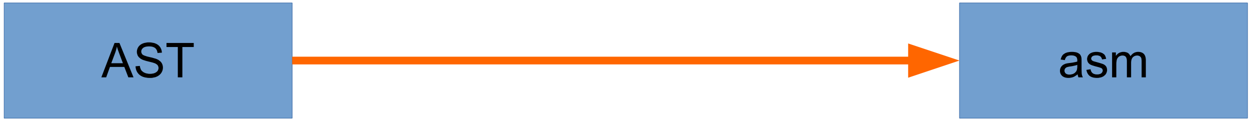
Compilers 2018-2019

IR analyses and transformations

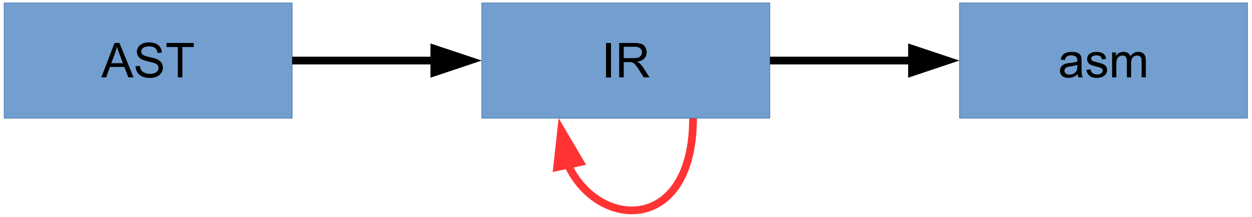
pract3



AST

asm

pract4



Goal

- IR transformation: bounds checking

- Performance analysis + optimization

Set-up

- LLVM 7.1 (debug build)
  - `LLVM_BUILD_LLVM_DYLIB=ON`
  - `CMAKE_BUILD_TYPE=Debug`

- GNU C Compiler

Compilation flow

```
int answer = 42;
echo(answer);
```

```
define void @echo(i32)

define i32 @main() {
entry:
  call void @echo(i32 42) #0
  ret i32 0
}
```

```
cheetah helloworld.c > helloworld.ll
```

Compilation flow

```
        .globl  main
        .type   main,@function
main:
.Lfunc_begin2:
        .file   1 "helloworld.c"
        .loc    1 1 0
        pushq   %rax
.Ltmp0:
        .loc    1 2 6 prologue_end
        movl    $42, %edi
        callq   echo
        xorl    %eax, %eax
        popq    %rcx
        retq
.Ltmp1:
.Lfunc_end2:
        .size   main, .Lfunc_end2-main
```

```
define void @echo(i32)

define i32 @main() {
entry:
  call void @echo(i32 42) #0
  ret i32 0
}
```

```
llc helloworld.ll > helloworld.s
```

Compilation flow

```
        .globl  main
        .type   main,@function
main:
.Lfunc_begin2:
        .file   1 "helloworld.c"
        .loc    1 1 0
        pushq   %rax
.Ltmp0:
        .loc    1 2 6 prologue_end
        movl    $42, %edi
        callq   echo
        xorl    %eax, %eax
        popq    %rcx
        retq
.Ltmp1:
.Lfunc_end2:
        .size   main, .Lfunc_end2-main
```

```
$ file helloworld
helloworld: ELF 64-bit LSB pie executable,
            x86-64, version 1 (SYSV),
            dynamically linked
```

```
gcc helloworld.s > helloworld
```

Compiler transformation

Part 1

```
define void @echo(i32)

define i32 @main() {
entry:
  call void @echo(i32 42) #0
  ret i32 0
}
```

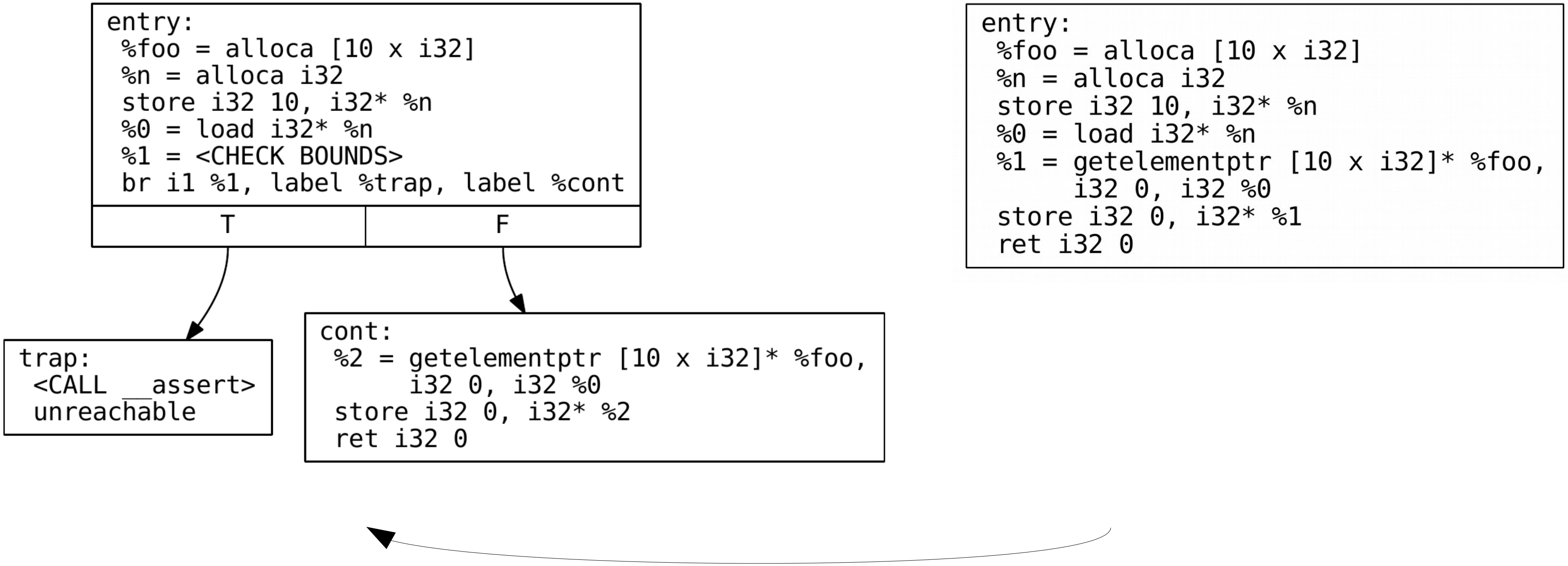
```
define void @echo(i32)

define i32 @main() {
entry:
  call void @echo(i32 42) #0
  ret i32 0
}
```

```
opt -cheetah-bc helloworld.ll
     > helloworld.checked.ll
```

Compiler transformation

Part 1



```
entry:
 %foo = alloca [10 x i32]
 %n = alloca i32
 store i32 10, i32* %n
 %0 = load i32* %n
 %1 = <CHECK BOUNDS>
 br i1 %1, label %trap, label %cont
```

```
trap:
 <CALL __assert>
 unreachable
```

```
cont:
 %2 = getelementptr [10 x i32]* %foo,
      i32 0, i32 %0
 store i32 0, i32* %2
 ret i32 0
```

```
entry:
 %foo = alloca [10 x i32]
 %n = alloca i32
 store i32 10, i32* %n
 %0 = load i32* %n
 %1 = getelementptr [10 x i32]* %foo,
      i32 0, i32 %0
 store i32 0, i32* %1
 ret i32 0
```

```
opt -cheetah-bc helloworld.ll
    > helloworld.checked.ll
```

Performance analysis

Part 2

- Measure and analyze
- Explain overhead
- Optimize

Organization

- Only modify:
  - `BoundsCheck/pass.cpp`
  - `REPORT.pdf`
  - `test/*.c`
- 2-week lab: deadline 21/5