Stage 4 Codegen Example 2

Monday, May 6, 2024 6:44 PM

```
foo: &(int) -> int

fn main() -> int {
  let x:int, fp:&(int) -> int

  entry:
    fp = $copy foo
    x = $call_idr fp(42) then exit

  exit:
    $ret x
}

fn foo(p: int) -> int {
  entry:
    $ret p
}
```

```
.data

.globl foo_              // name mangling
foo_: .quad "foo"        // note initialization

.text

.globl foo
foo:
  pushq %rbp
  movq %rsp, %rbp
  subq $0, %rsp
  jmp foo_entry

foo_entry:
  movq 16(%rbp), %rax
  jmp foo_epilogue

foo_epilogue:
  movq %rbp, %rsp
  popq %rbp
  ret

.globl main
main:
  pushq %rbp
  movq %rsp, %rbp
  subq $16, %rsp
  movq $0, -8(%rbp)
  movq $0, -16(%rbp)
  jmp main_entry

main_entry:
  movq foo_(%rip), %r8   // name mangling
  movq %r8, -8(%rbp)     // notice ordered `fp` before `x`
  pushq $42
  subq $8, %rsp          // fixing alignment
  call *-8(%rbp)         // indirect call
  movq %rax, -16(%rbp)
  addq $16, %rsp         // restoring stack (inc. alignment)
  jmp main_exit

main_exit:
  movq -16(%rbp), %rax
  jmp main_epilogue

main_epilogue:
  movq %rbp, %rsp
  popq %rbp
  ret
```