PA3 实验报告

工作内容

实验部分

PA3 实验要求实现抽象类、匿名函数的 TAC 生成，具体来说需要做这些事情：

1. 处理抽象类和抽象方法，跳过抽象方法的生成
2. 修改函数调用的方式，使得它可以处理各种情况
3. 实现匿名函数的闭包生成

抽象类

抽象类的支持很简单，只需要把抽象方法的生成全部跳过即可。

局部类型推断

啥也不用做。

First-class functions

通用的调用方式

原框架代码中直接按照函数的信息生成 Virtual/Static 的 Call ，但现在调用的可能是一个任意的函数，所以我采用了函数指针的方法：

```
struct FuncPtr {
  fn: fn(),
  self: *mut (),
}
```

其中 `fn` 是需要调用的函数的“地址”， `self` 是要传给函数的第一个参数。我也让每个 `static` 函数多接收了一个参数，只不过不会用到，这样就统一了调用的方式。

调用的时候，会生成这样的代码：

```
1       _T15 = *(_T13 + 0)
2       _T16 = *(_T13 + 4)
3       parm _T16
4       _T17 = call _T15
```

这种方法会 Alloc 很多次，但反正内存不要钱，能 work 就行，而且是很好优化的（PA4 #flag）。

在过程中遇到需要升级 tacvm 依赖的版本和需要使用 LoadFunc 获取静态函数的地址的问题，在阅读 tacvm 文档后解决。

闭包

每个 lambda 都要生成一段函数，名为 `_lambda_row_col` ，调用时，先把所有本地的 variable 和 this 复制一份到一块空间，作为 lambda 调用的 this 参数。然后覆盖掉这些 variable 的信息，使得在 lambda 中都通过 this 去找对应的 variable 。这样就实现了闭包，和文档里要求的是一致的，只不过并不是按需 capture ，而是把所有可能涉及的都 capture 了。需要特别处理 lambda 嵌套的一些情况。

遇到的困难

不难。多了 LoadFunc 的 tac 指令就更简单了。