使用環境與編譯流程

- Linux Ubuntu 14.10
- g++
- LLVM 3.4
- 1. 編譯所有檔案

  ```
  $ make
  ```

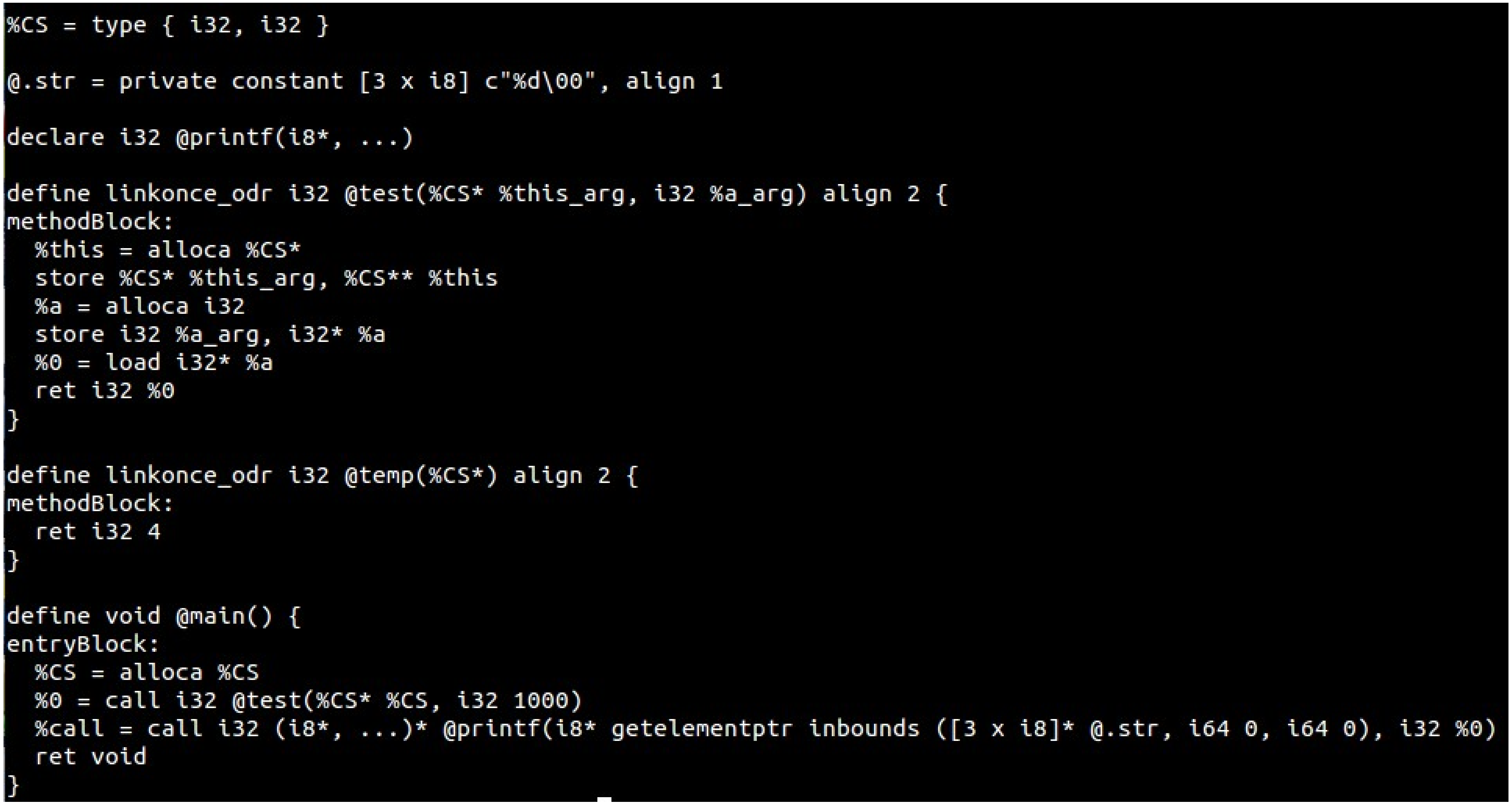
  2. 將程式碼轉成 LLVM IR

  ```
  $ ./a.out testdata.java
  ```

  3.$ lli a.ll

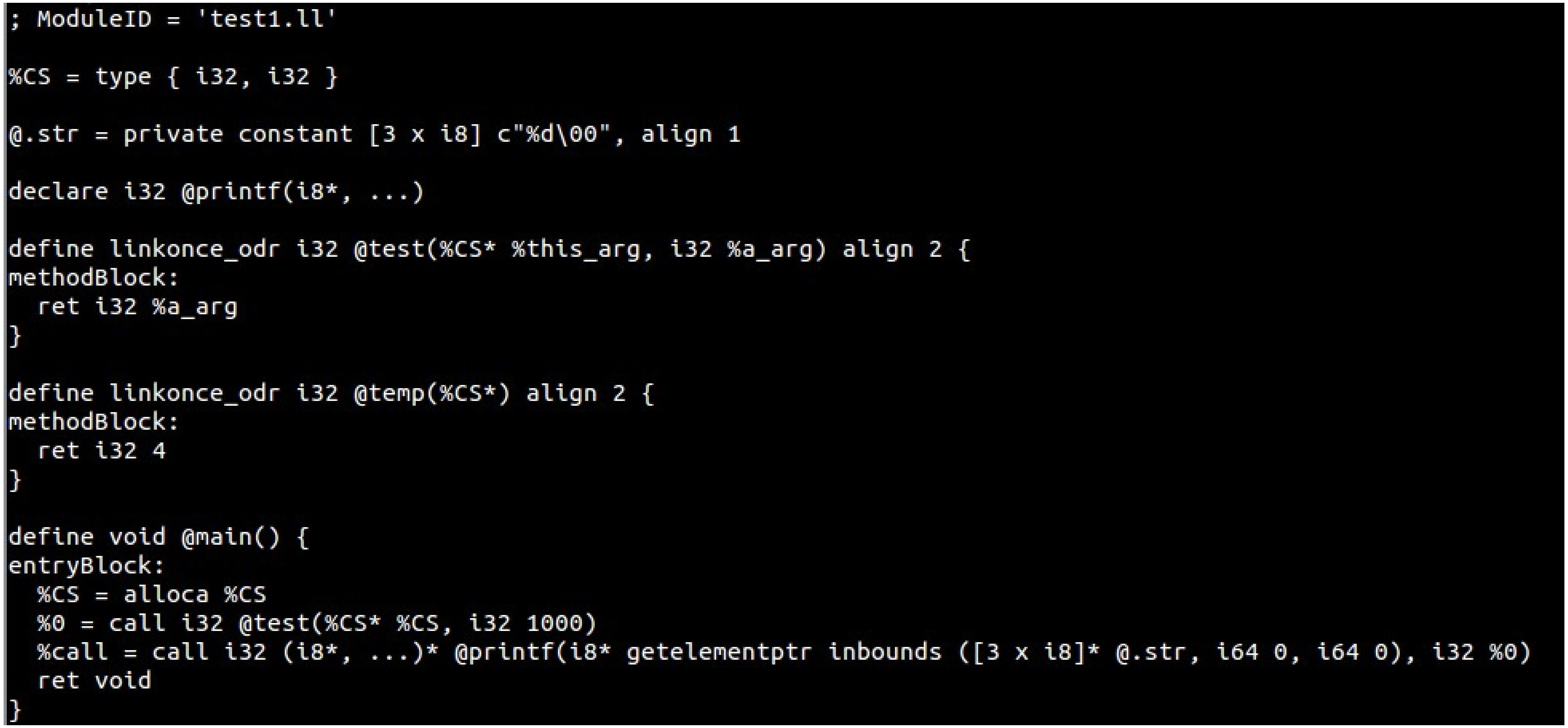
  執行 LLVM IR

Origin LLVM IR



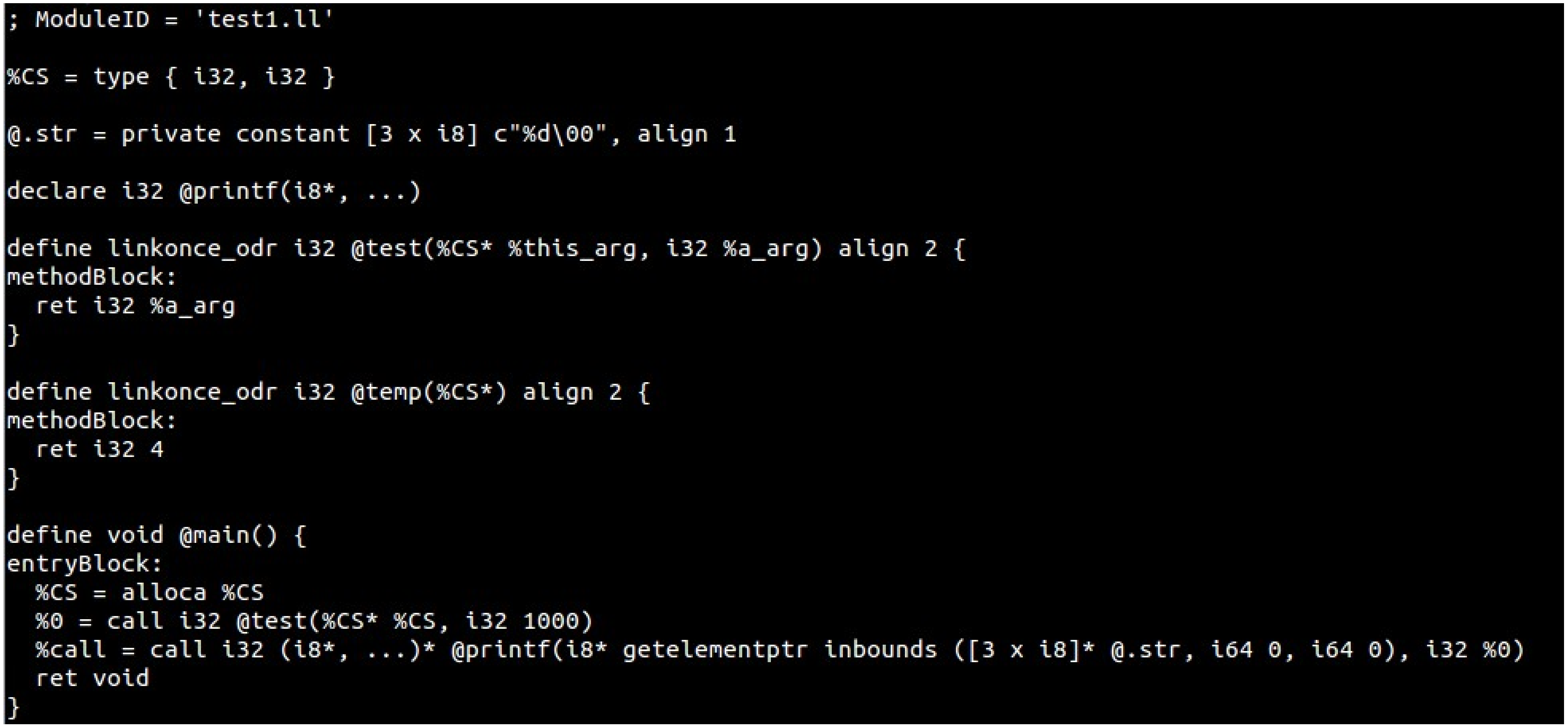
```
%CS = type { i32, i32 }

@.str = private constant [3 x i8] c"%d\00", align 1

declare i32 @printf(i8*, ...)

define linkonce_odr i32 @test(%CS* %this_arg, i32 %a_arg) align 2 {
methodBlock:
  %this = alloca %CS*
  store %CS* %this_arg, %CS** %this
  %a = alloca i32
  store i32 %a_arg, i32* %a
  %0 = load i32* %a
  ret i32 %0
}

define linkonce_odr i32 @temp(%CS*) align 2 {
methodBlock:
  ret i32 4
}

define void @main() {
entryBlock:
  %CS = alloca %CS
  %0 = call i32 @test(%CS* %CS, i32 1000)
  %call = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str, i64 0, i64 0), i32 %0)
  ret void
}
```

Opt test1.ll -mem2reg -constprop -S



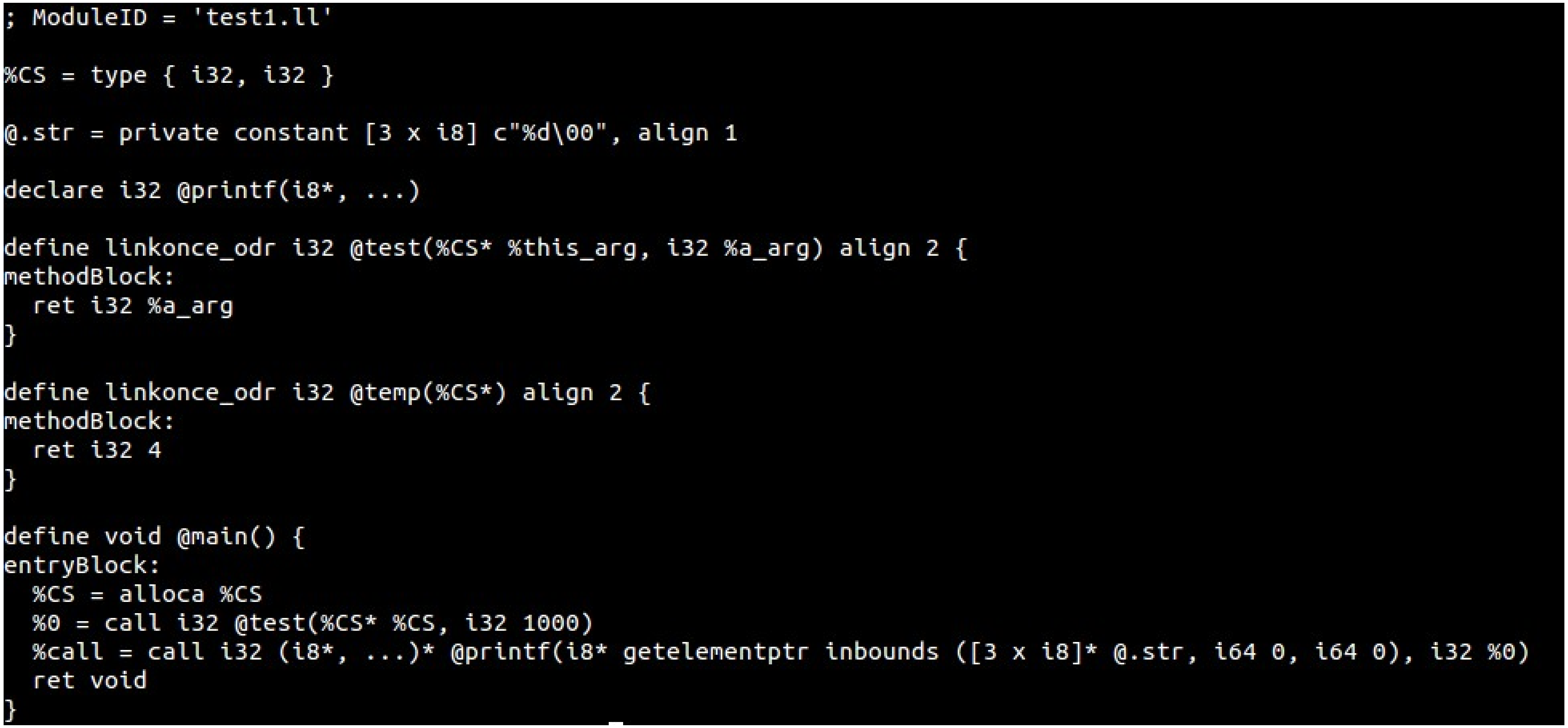
```
; ModuleID = 'test1.ll'

%CS = type { i32, i32 }

@.str = private constant [3 x i8] c"%d\00", align 1

declare i32 @printf(i8*, ...)

define linkonce_odr i32 @test(%CS* %this_arg, i32 %a_arg) align 2 {
methodBlock:
  ret i32 %a_arg
}

define linkonce_odr i32 @temp(%CS*) align 2 {
methodBlock:
  ret i32 4
}

define void @main() {
entryBlock:
  %CS = alloca %CS
  %0 = call i32 @test(%CS* %CS, i32 1000)
  %call = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str, i64 0, i64 0), i32 %0)
  ret void
}
```

opt test1.ll -mem2reg -constprop -dce -S



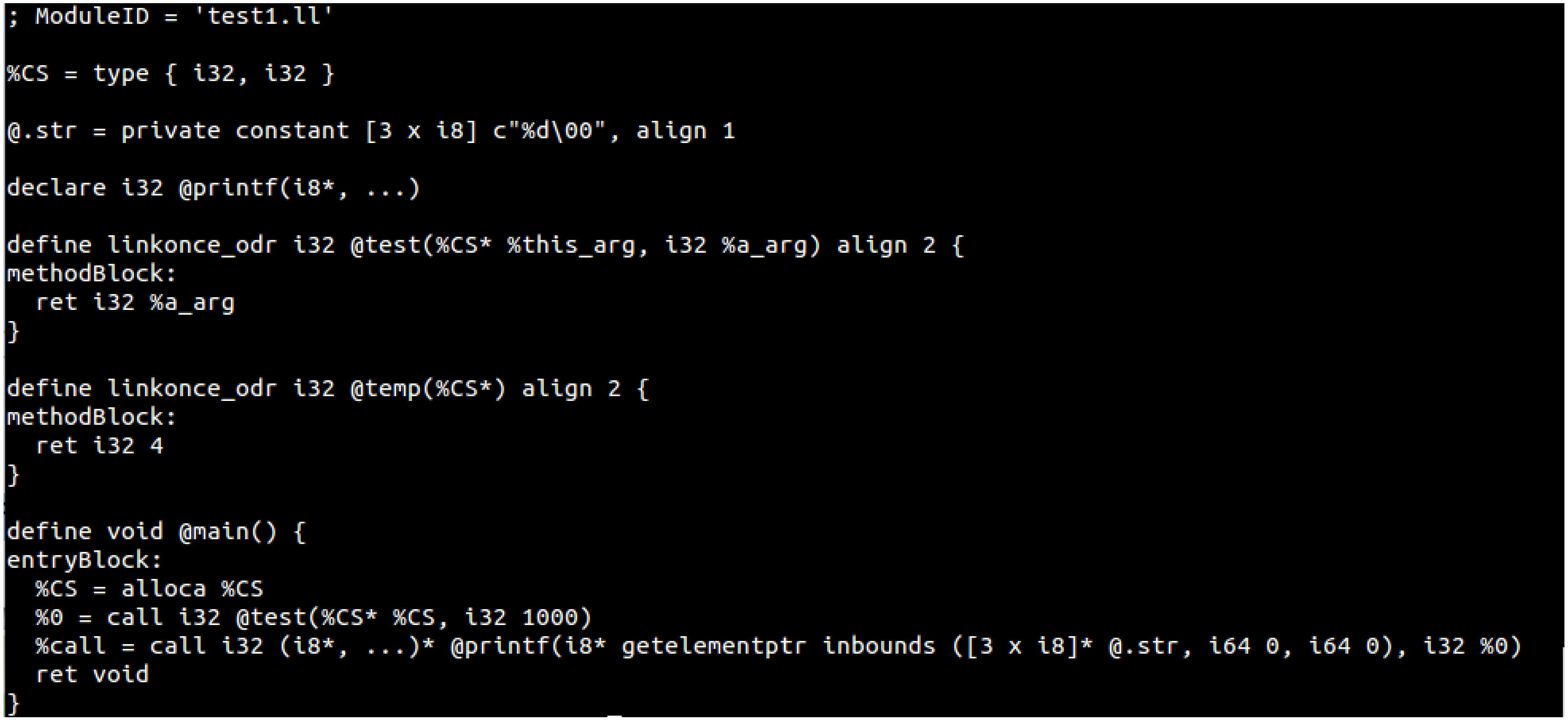
```
; ModuleID = 'test1.ll'

%CS = type { i32, i32 }

@.str = private constant [3 x i8] c"%d\00", align 1

declare i32 @printf(i8*, ...)

define linkonce_odr i32 @test(%CS* %this_arg, i32 %a_arg) align 2 {
methodBlock:
  ret i32 %a_arg
}

define linkonce_odr i32 @temp(%CS*) align 2 {
methodBlock:
  ret i32 4
}

define void @main() {
entryBlock:
  %CS = alloca %CS
  %0 = call i32 @test(%CS* %CS, i32 1000)
  %call = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str, i64 0, i64 0), i32 %0)
  ret void
}
```

opt test1.ll -mem2reg -constprop -simplifycfg -S



```
; ModuleID = 'test1.ll'

%CS = type { i32, i32 }

@.str = private constant [3 x i8] c"%d\00", align 1

declare i32 @printf(i8*, ...)

define linkonce_odr i32 @test(%CS* %this_arg, i32 %a_arg) align 2 {
methodBlock:
  ret i32 %a_arg
}

define linkonce_odr i32 @temp(%CS*) align 2 {
methodBlock:
  ret i32 4
}

define void @main() {
entryBlock:
  %CS = alloca %CS
  %0 = call i32 @test(%CS* %CS, i32 1000)
  %call = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str, i64 0, i64 0), i32 %0)
  ret void
}
```

opt test.ll -mem2reg -simplifycfg -S



```
; ModuleID = 'test1.ll'

%CS = type { i32, i32 }

@.str = private constant [3 x i8] c"%d\00", align 1

declare i32 @printf(i8*, ...)

define linkonce_odr i32 @test(%CS* %this_arg, i32 %a_arg) align 2 {
methodBlock:
  ret i32 %a_arg
}

define linkonce_odr i32 @temp(%CS*) align 2 {
methodBlock:
  ret i32 4
}

define void @main() {
entryBlock:
  %CS = alloca %CS
  %0 = call i32 @test(%CS* %CS, i32 1000)
  %call = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([3 x i8]* @.str, i64 0, i64 0), i32 %0)
  ret void
}
```