LLVM Introduction

Compiler Design WS 17/18

Lab Assignment 3

- Implement "potentially uninitialized variable" pass using LLVM

```
void test(bool b) {
    int a;




    printf("%d\n",a);
}
```

Variable a is always uninitialized

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }



    printf("%d\n",a);
}
```

Variable a may be uninitialized

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    } else {
        a = 2;
    }

    printf("%d\n",a);
}
```

Variable a is never uninitialized

```
void test(bool b) {
    int x, y;
    if (b) x = 1;
    else   y = 1;

    if (b) printf("%d\n",x);
    else   printf("%d\n",y);
}
```

x, y are always initialized if they are used. However we detect this as "potentially uninitialized".

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

Allocate b, a on stack

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

Write function argument (%0)
into stack slot %2

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

Specify debugging information about %2 and %3

```
call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15

...

!9 = !DIBasicType(name: "_Bool", size: 8, encoding: DW_ATE_boolean)
!10 = !DILocalVariable(name: "b", arg: 1, scope: !6, file: !1, line: 1, type: !9)
!11 = !DIExpression()
!12 = !DILocation(line: 1, column: 16, scope: !6)
!13 = !DILocalVariable(name: "a", scope: !6, file: !1, line: 2, type: !14)
!14 = !DIBasicType(name: "int", size: 32, encoding: DW_ATE_signed)
!15 = !DILocation(line: 2, column: 9, scope: !6)

...
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
```

... %2 is variable "b" declared on line 1 having type _Bool

```
!9 = !DIBasicType(name: "_Bool", size: 8, encoding: DW_ATE_boolean)
!10 = !DILocalVariable(name: "b", arg: 1, scope: !6, file: !1, line: 1, type: !9)
!11 = !DIExpression()
!12 = !DILocation(line: 1, column: 16, scope: !6)
!13 = !DILocalVariable(name: "a", scope: !6, file: !1, line: 2, type: !14)
!14 = !DIBasicType(name: "int", size: 32, encoding: DW_ATE_signed)
!15 = !DILocation(line: 2, column: 9, scope: !6)
```

...

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
```

... %3 is variable "a" declared on line 2 having type int

```
!9 = !DIBasicType(name: "_Bool", size: 8, encoding: DW_ATE_boolean)
!10 = !DILocalVariable(name: "b", arg: 1, scope: !6, file: !1, line: 1, type: !9)
!11 = !DIExpression()
!12 = !DILocation(line: 1, column: 16, scope: !6)
!13 = !DILocalVariable(name: "a", scope: !6, file: !1, line: 2, type: !14)
!14 = !DIBasicType(name: "int", size: 32, encoding: DW_ATE_signed)
!15 = !DILocation(line: 2, column: 9, scope: !6)
```

...

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15

...

!9 = !DIBasicType(name: "_Bool", size: 8, encoding: DW_ATE_boolean)
!10 = !DILocalVariable(name: "b", arg: 1, scope: !6, file: !1, line: 1, type: !9)
!11 = !DIExpression()
!12 = !DILocation(line: 1, column: 16, scope: !6)
!13 = !DILocalVariable(name: "a", scope: !6, file: !1, line: 2, type: !14)
!14 = !DIBasicType(name: "int", size: 32, encoding: DW_ATE_signed)
!15 = !DILocation(line: 2, column: 9, scope: !6)

...
```

!dbg provides precise location information.

Also used for other instructions, not just var declarations.

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

Branch on b (%2)

Go to %7 if true, go to %8 if false

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

Store integer 1 in a (%3)

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

Call printf() with argument a
(Some parts omitted)

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

b uninitialized

b initialized

b still initialized

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

a uninitialized

a initialized **(in this branch)**

a may be initialized

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

; <label>:8:                                      ; preds = %7, %1
  %9 = load i32, i32* %3, align 4, !dbg !22
  %10 = call i32 (i8*, ...) @printf(...), !dbg !23
  ret void, !dbg !24
}
```

a uninitialized

a initialized **(in this branch)**

a may be initialized, depending on whether it comes from block %7 or %1

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
  %2 = alloca i8, align 1
  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8
  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16
  %6 = trunc i8 %5 to i1, !dbg !16
  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

  ...
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
                  %2 = alloca i8, align 1
                                  %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8

  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16

  %6 = trunc i8 %5 to i1, !dbg !16

  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

  ...
}
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
                  %2 = alloca i8, align 1
                                        %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8

  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16

  %6 = trunc i8 %5 to i1, !dbg !16

  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

  ...
}
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

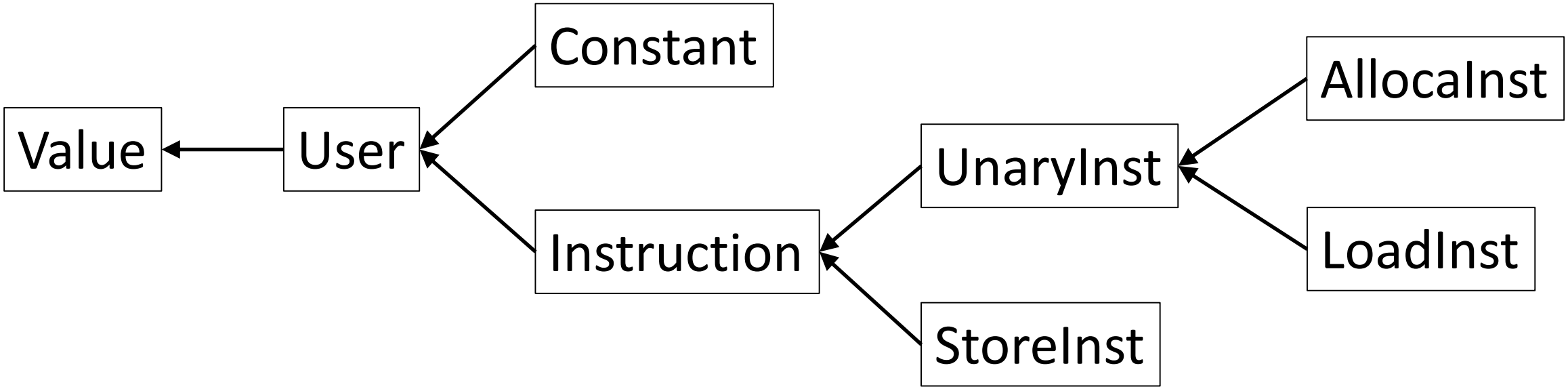
```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
                  %2 = alloca i8, align 1
                                        %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8

  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16

  %6 = trunc i8 %5 to i1, !dbg !16

  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

  ...
}
```

```
void test(bool b) {
    int a;
    if (b) {
        a = 1;
    }
    printf("%d\n",a);
}
```

```
define void @test(i1 zeroext) #0 !dbg !6 {
                  %2 = alloca i8, align 1
                                        %3 = alloca i32, align 4
  %4 = zext i1 %0 to i8

  store i8 %4, i8* %2, align 1
  call void @llvm.dbg.declare(metadata i8* %2, metadata !10, metadata !11), !dbg !12
  call void @llvm.dbg.declare(metadata i32* %3, metadata !13, metadata !11), !dbg !15
  %5 = load i8, i8* %2, align 1, !dbg !16

  %6 = trunc i8 %5 to i1, !dbg !16

  br i1 %6, label %7, label %8, !dbg !18

; <label>:7:                                      ; preds = %1
  store i32 1, i32* %3, align 4, !dbg !19
  br label %8, !dbg !21

  ...
}
```

Constant(i32 1)


Value
User
Constant
Instruction
UnaryInst
StoreInst
AllocaInst
LoadInst

Lab Assignment 3

- Implement "potentially uninitialized variable" pass using LLVM
- Suggestion: Iterative data-flow analysis

Lab Assignment 3

- Implement "potentially uninitialized variable" pass using LLVM
- Suggestion: Iterative data-flow analysis
  - Direction?
  - GEN, KILL?
  - IN, OUT equations? (Union or intersection?)
  - Initialization?

Lab Assignment 3

- Implement "potentially uninitialized variable" pass using LLVM
- Suggestion: Iterative data-flow analysis
- Implement "fixing" pass, which initialized all potentially uninitialized variables

Lab Assignment 3

- Implement "potentially uninitialized variable" pass using LLVM
- Suggestion: Iterative data-flow analysis
- Implement "fixing" pass, which initialized all potentially uninitialized variables

```
testN.c    :  Test file
testN.def  :  Expected "uninitialized variable" output
testN.fix  :  Expected output of code after fixing pass applied
```

Lab Assignment 3

- Implement "potentially uninitialized variable" pass using LLVM
- Suggestion: Iterative data-flow analysis
- Implement "fixing" pass, which initialized all potentially uninitialized variables

```
testN.c    :  Test file
testN.def  :  Expected "uninitialized variable" output
testN.fix  :  Expected output of code after fixing pass applied
---
pass.cpp   :  Pass template
./test.sh  :  Tests your pass against expected outputs
```