# Finding (and Outlining) Similarity at the IR Level

**Andrew Litteken**

Apple

## Copy-and-Pasted Code

```
int c = 4;
int d = 5;
c = c + 1;
d = c + 2;
d = c * 3;
d = 4 + d;

   . . .

int c = 4;
int d = 5;
c = c + 1;
d = c + 2;
d = c * 3;
d = 4 + d;
```

## Same Sequences of Operations

```
int c = 4;
int d = 5;
c = c + 1;
d = c + 2;
d = c * 3;
d = 4 + d;
```

. . .

```
int f = 4;
int e = 5;
e = e + 1;
f = e + 2;
f = e * 3;
f = 4 + f;
```

```cpp
int fn(const std::vector<int> &myVec) {
    for (auto it = myVec.begin(),
      et = myVec.end(); it != et; ++it) {
        if (*it & 1)
            return 0;
    }
    return 1;
}
```

```cpp
int fn(const std::vector<int> &myVec) {
    for (const int &x : myVec) {
        if (x % 2)
            return 1;
    }
    return 0;
}
```

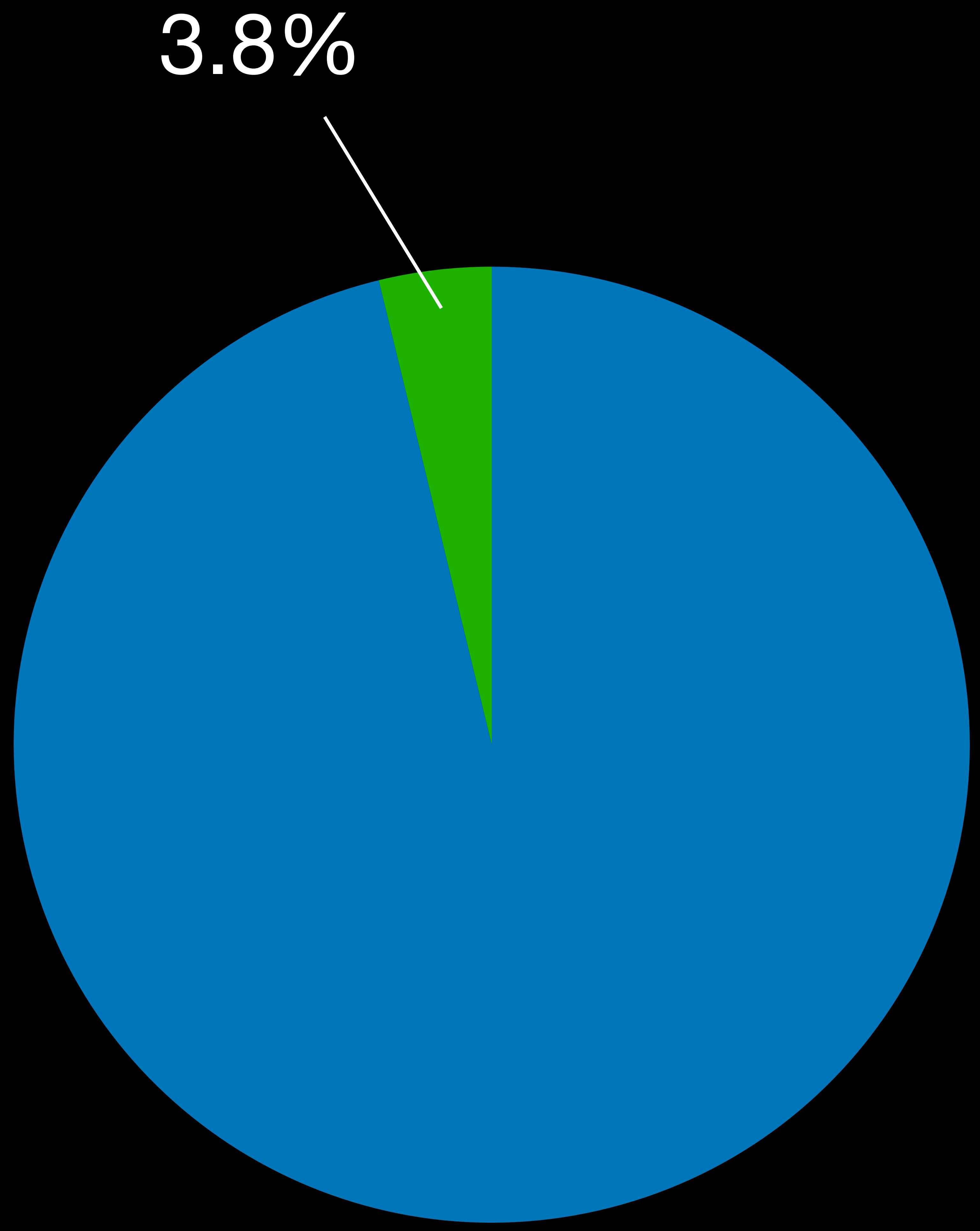
```cpp
int fn(const std::vector<int> &myVec) {
    for (auto it = myVec.begin(),
      et = myVec.end(); it != et; ++it) {
        if (*it & 1)
            return 0;
    }
    return 1;
}
```

```cpp
int fn(const std::vector<int> &myVec) {
    for (const int &x : myVec) {
        if (x % 2)
            return 1;
    }
    return 0;
}
```

```cpp
int fn(const std::vector<int> &myVec) {
    for (auto it = myVec.begin(),
      et = myVec.end(); it != et; ++it) {
        if (*it & 1)
            return 0;
    }
    return 1;
}
```

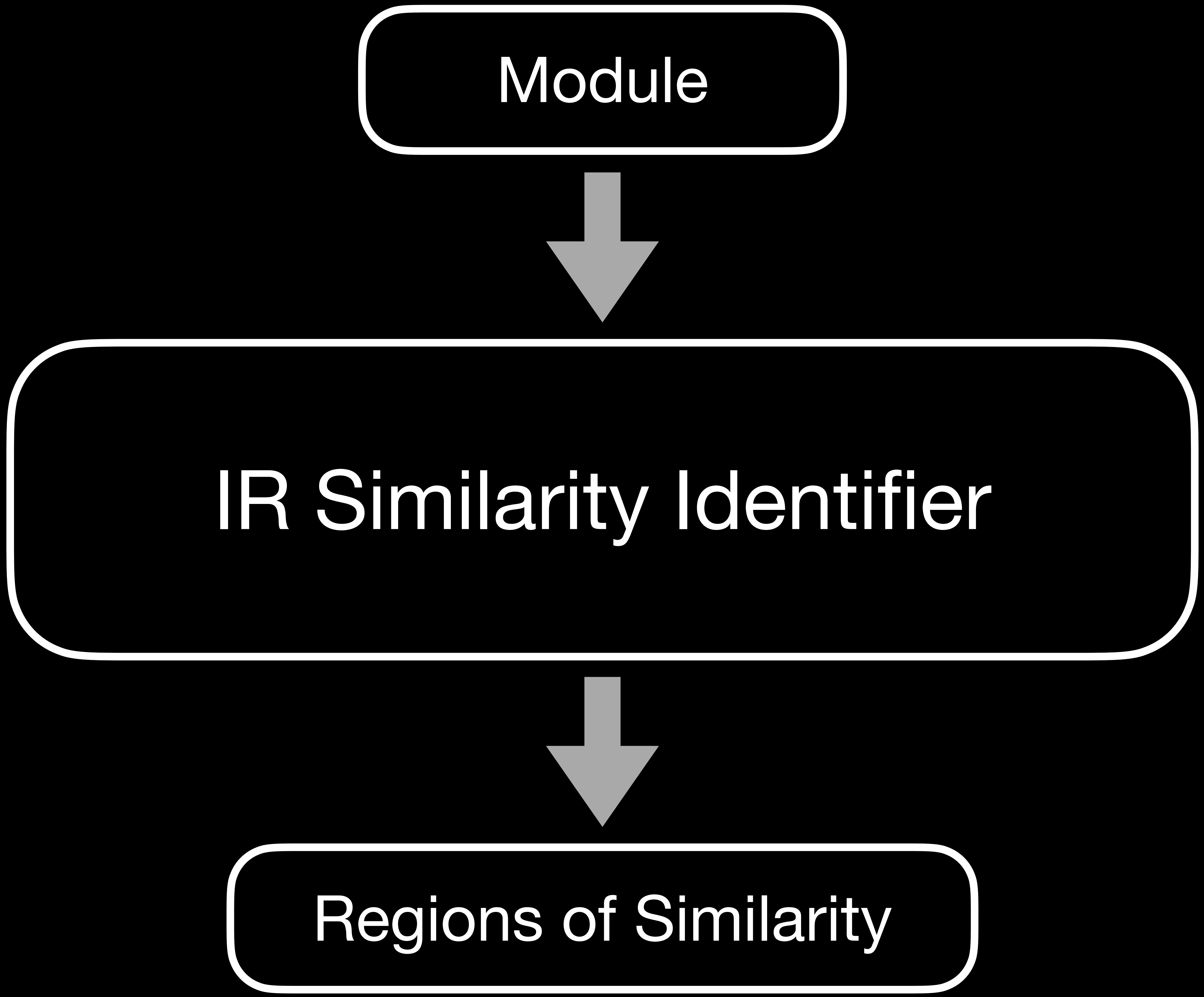
```cpp
int fn(const std::vector<int> &myVec) {
    for (const int &x : myVec) {
        if (x % 2)
            return 1;
    }
    return 0;
}
```

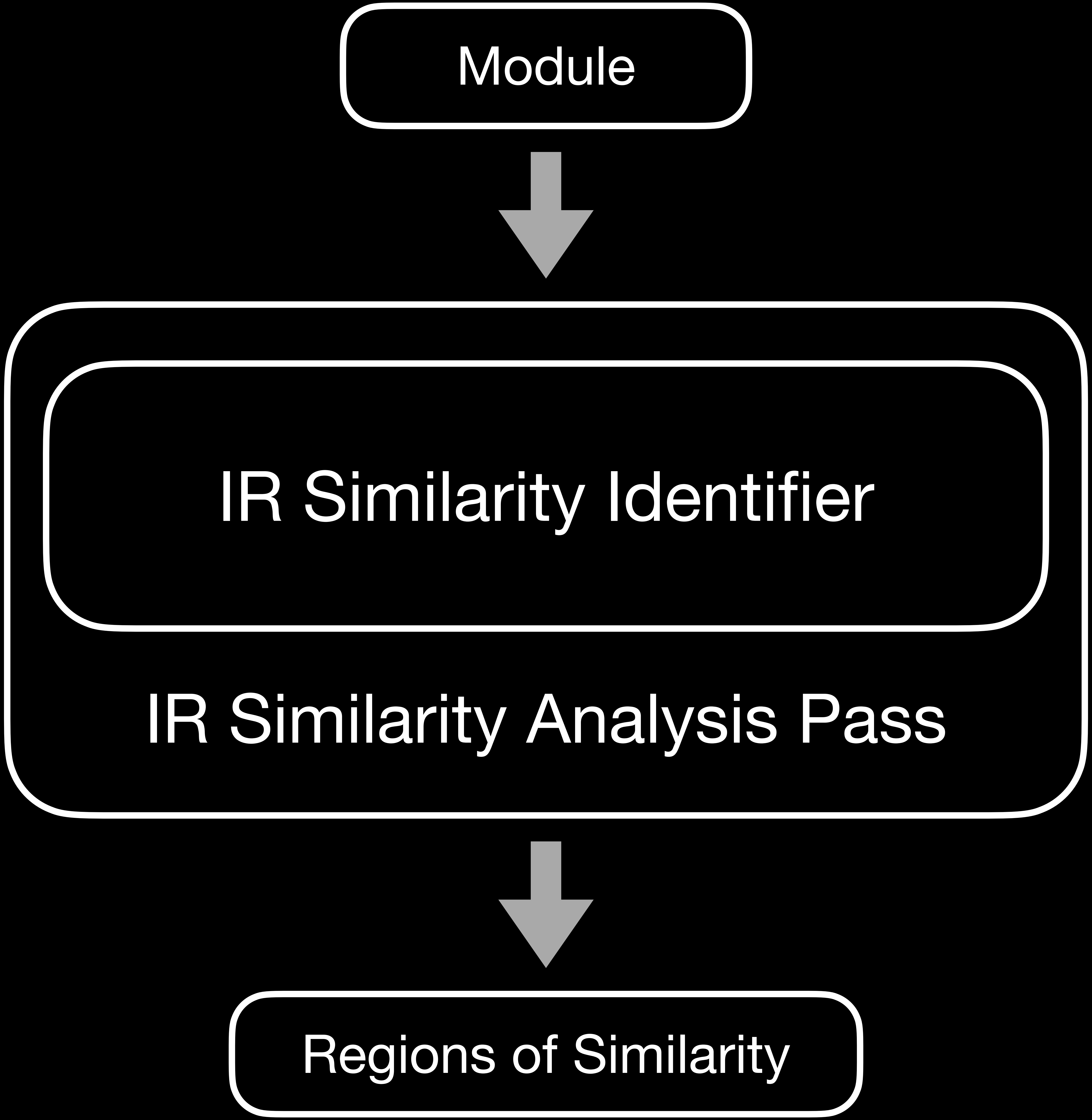
```llvm
%11 = load i32, i32* %10, align 4
%12 = and i32 %11, 1
%13 = icmp eq i32 %12, 0
%14 = getelementptr inbounds i32, i32* %10, i64 1
br i1 %13, label %7, label %15
```



LLVM Test Suite

## IR Similarity Identifier


Module
IR Similarity Identifier
Regions of Similarity


Module
IR Similarity Identifier
IR Similarity Analysis Pass
Regions of Similarity

## IR Mapping

```
%b = load i32, i32* %a, align 4
```

## IR Mapping

Opcode

```
%b = load i32, i32* %a, align 4
```

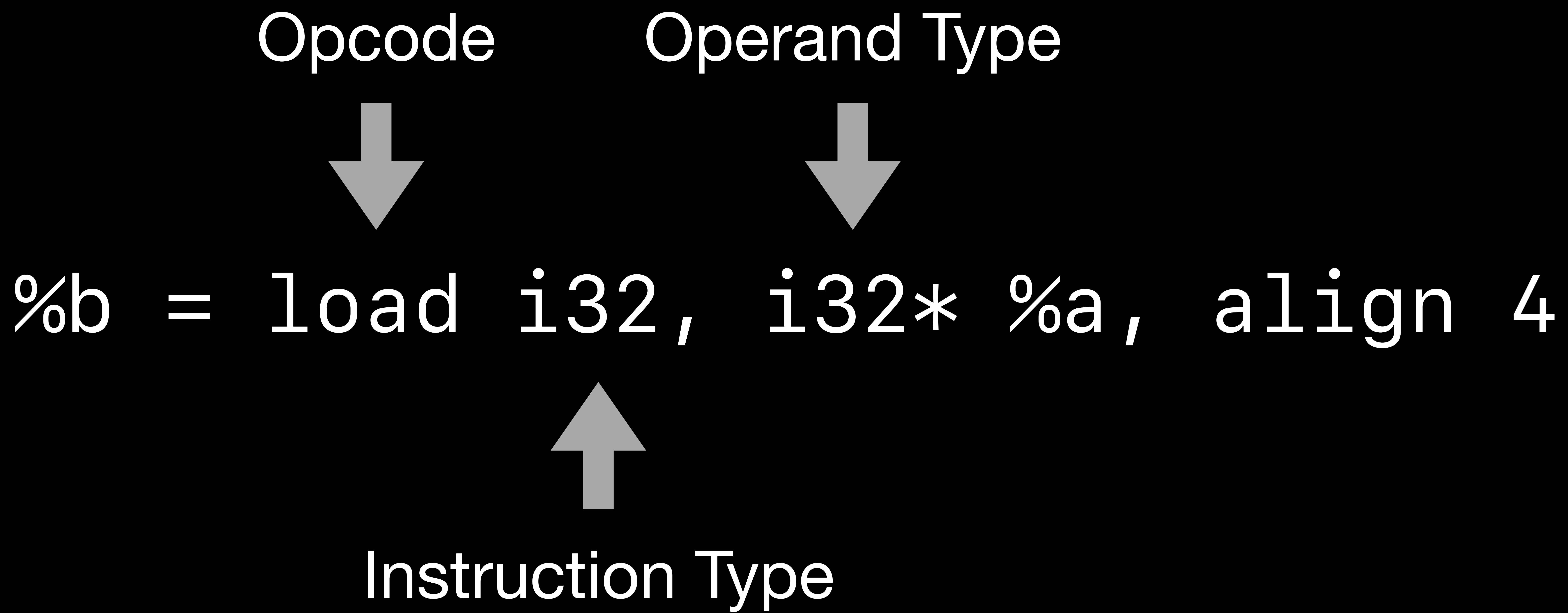
## IR Mapping

Opcode

```
%b = load i32, i32* %a, align 4
```

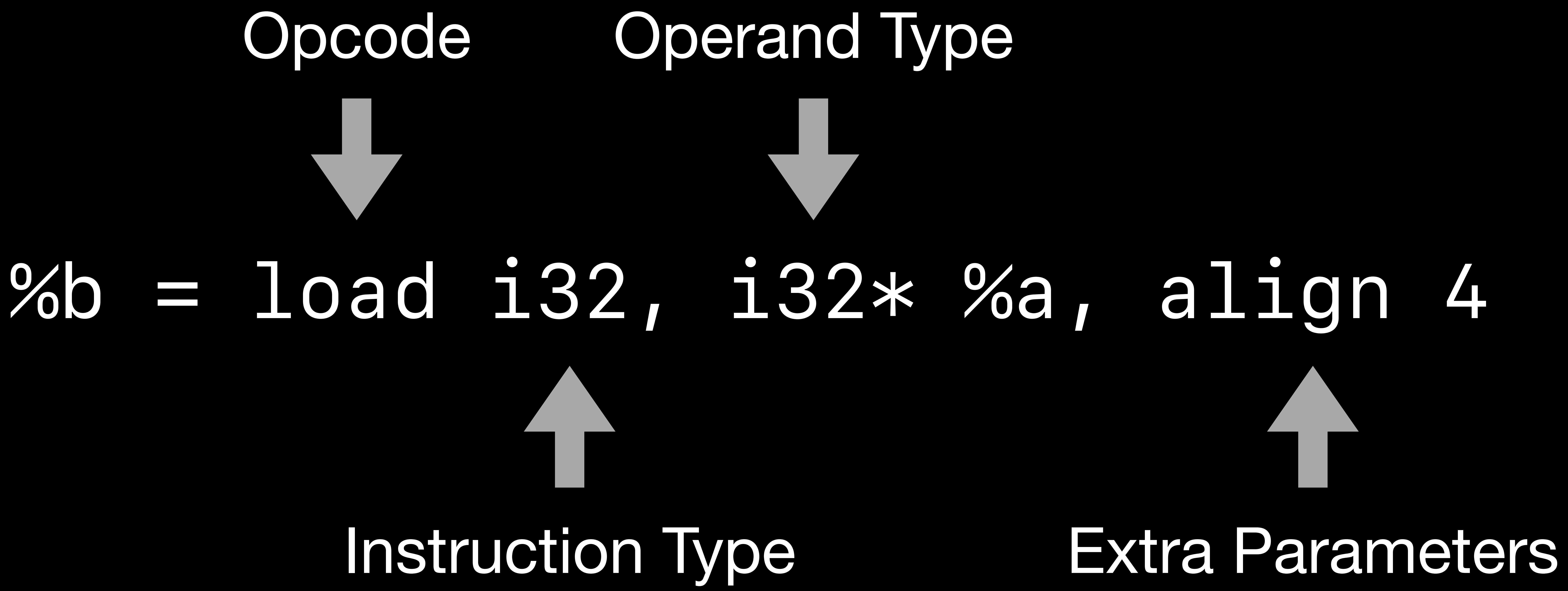
Instruction Type

## IR Mapping



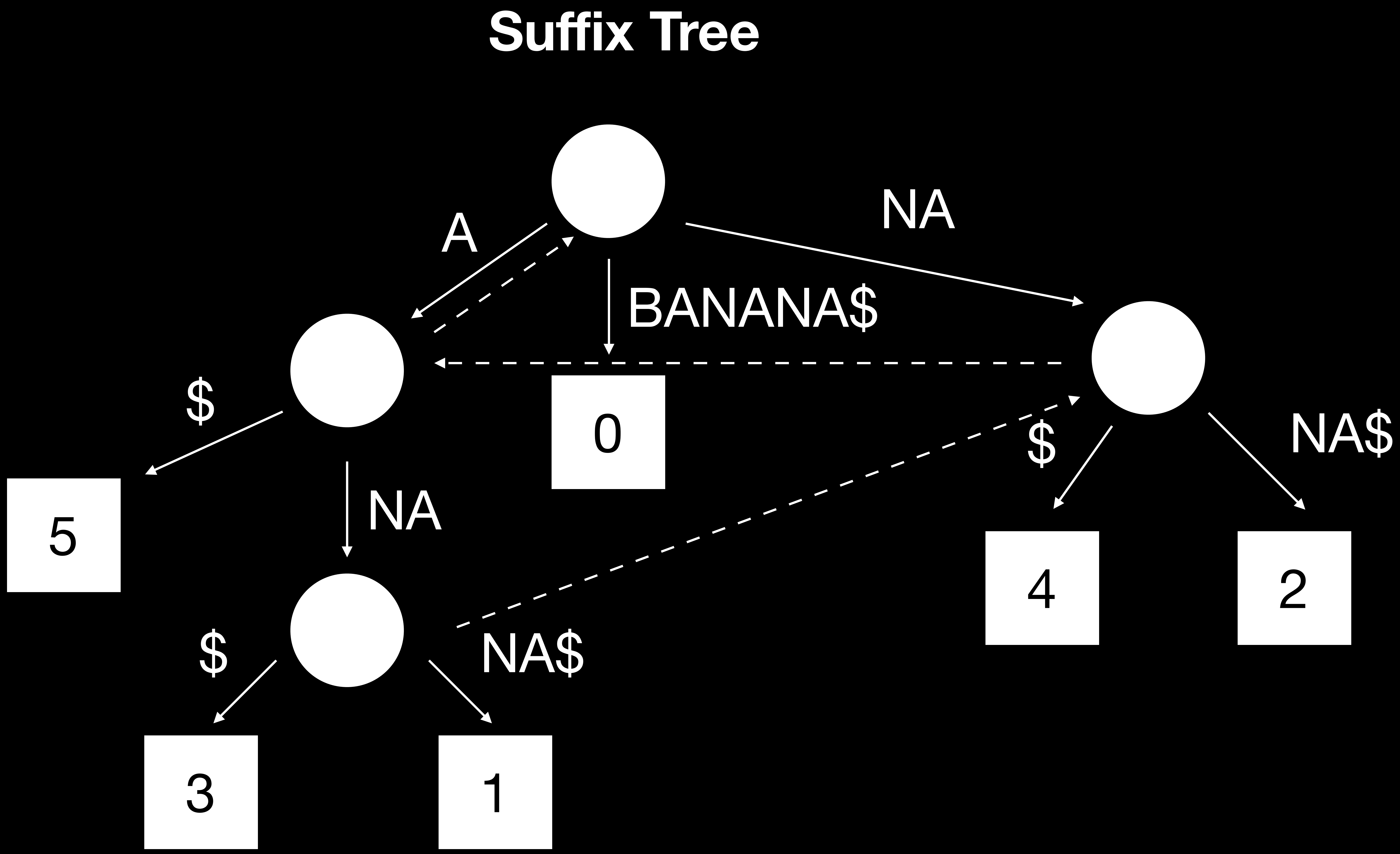
```
%b = load i32, i32* %a, align 4
```

## IR Mapping



```
%b = load i32, i32* %a, align 4
```

## Substring Detection



## Structural Analysis

```
%0 = load i32, i32* %c
%add = add i32 %0, 1
store i32 %add, i32* %c
```

?

```
%0 = load i32, i32* %d
%add = add i32 %0, 1
store i32 %add, i32* %d
```

?

```
%0 = load i32, i32* %d
%add = add i32 %0, 1
store i32 %add, i32* %e
```

## Structural Analysis

### Candidate A

```
%0 = load i32, i32* %c
%add = add i32 %0, 1
store i32 %add, i32* %c
%1 = load i32, i32* %b
%add1 = add i32 %1, 2
store i32 %add1, i32* %d
```

### Candidate B

```
%0 = load i32, i32* %a
%add = add i32 %0, 1
store i32 %add, i32* %a
%1 = load i32, i32* %b
%add1 = add i32 %1, 2
store i32 %add1, i32* %c
```

## Structural Analysis

**Candidate A**

```
%0 = load i32, i32* %c
%add = add i32 %0, 1
store i32 %add, i32* %c
%1 = load i32, i32* %b
%add1 = add i32 %1, 2
store i32 %add1, i32* %d
```

**Candidate B**

```
%0 = load i32, i32* %a
%add = add i32 %0, 1
store i32 %add, i32* %a
%1 = load i32, i32* %b
%add1 = add i32 %1, 2
store i32 %add1, i32* %c
```

## Structural Analysis

```
%0 = load i32, i32* %c
%add = add i32 %0, 1
store i32 %add, i32* %c
```

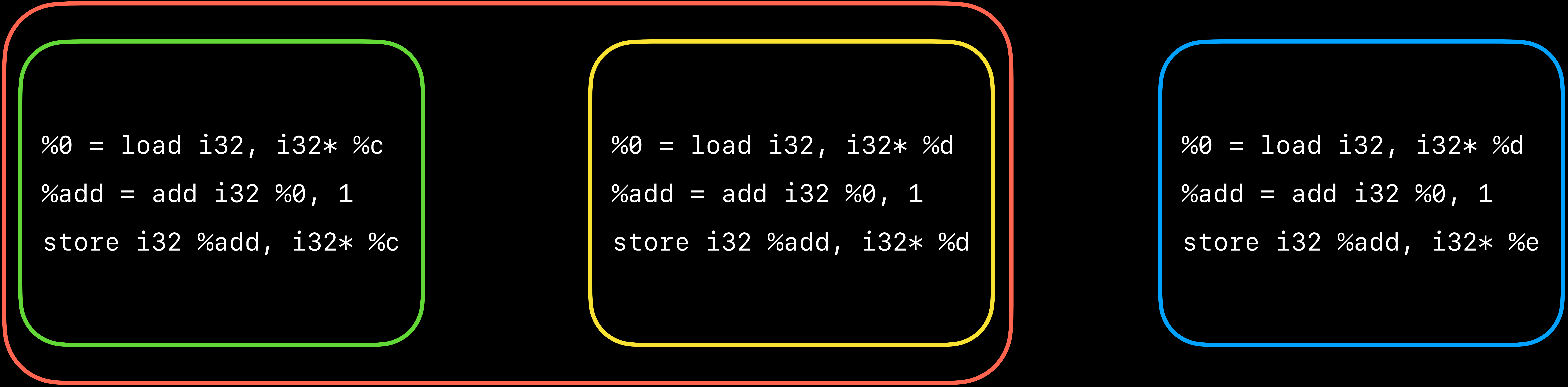
✅

```
%0 = load i32, i32* %d
%add = add i32 %0, 1
store i32 %add, i32* %d
```

❌

```
%0 = load i32, i32* %d
%add = add i32 %0, 1
store i32 %add, i32* %e
```

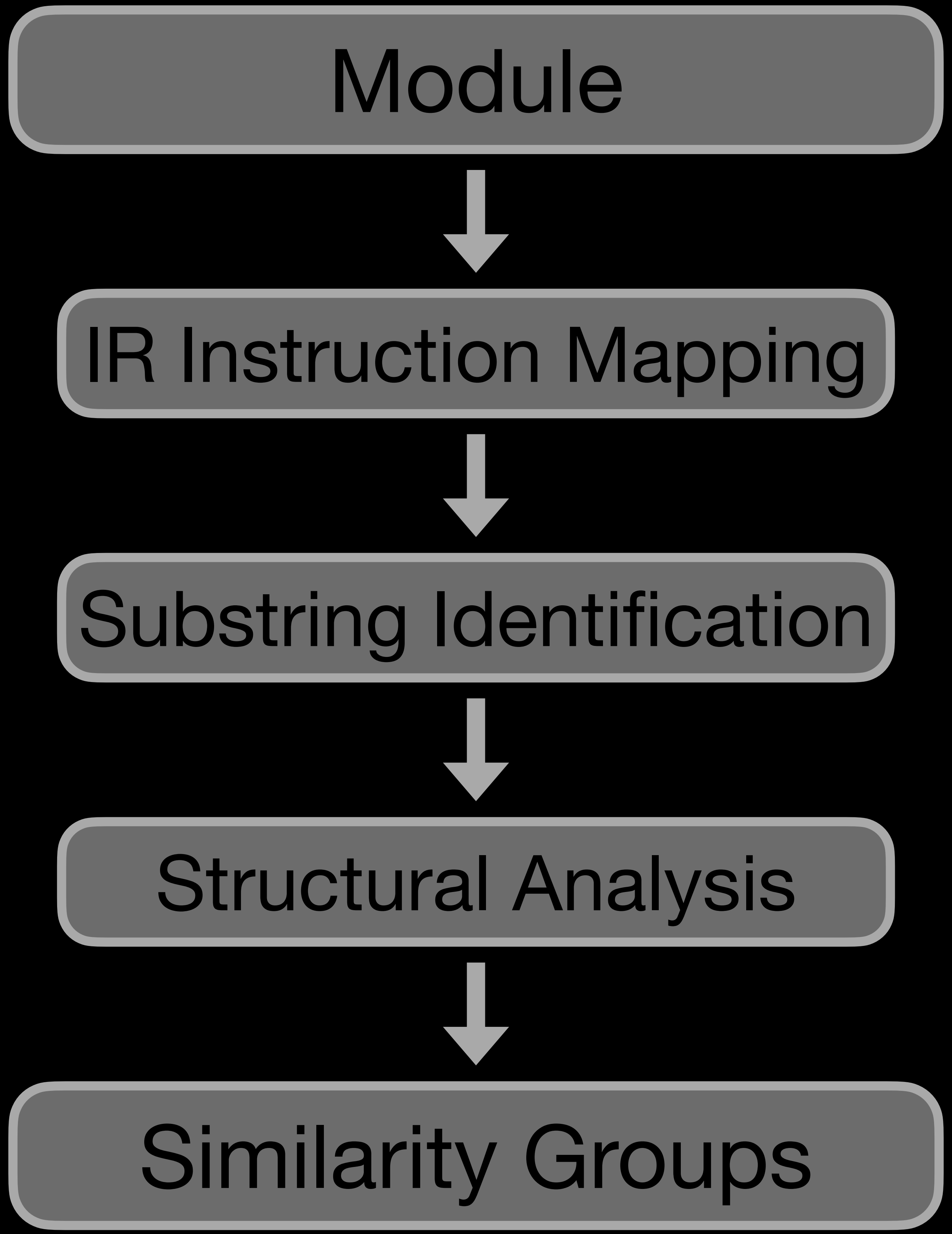
## Structural Analysis

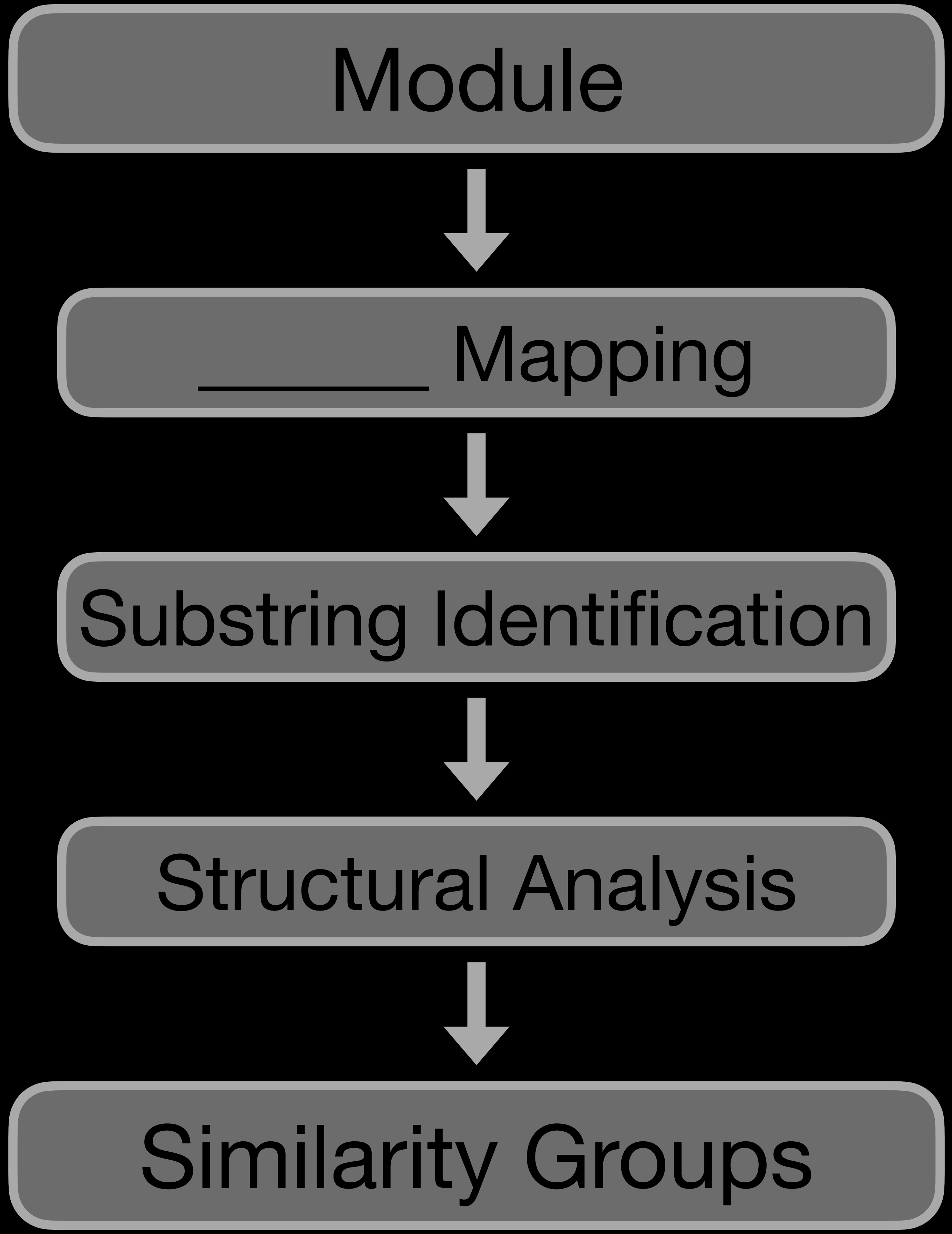


```
%0 = load i32, i32* %c
%add = add i32 %0, 1
store i32 %add, i32* %c
```

```
%0 = load i32, i32* %d
%add = add i32 %0, 1
store i32 %add, i32* %d
```

```
%0 = load i32, i32* %d
%add = add i32 %0, 1
store i32 %add, i32* %e
```

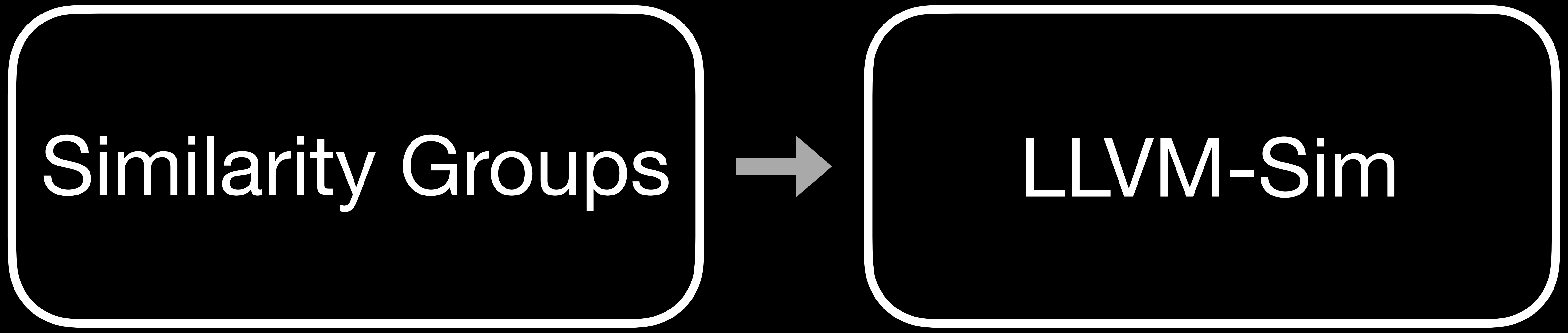

Module
IR Instruction Mapping
Substring Identification
Structural Analysis
Similarity Groups


Module
______ Mapping
Substring Identification
Structural Analysis
Similarity Groups

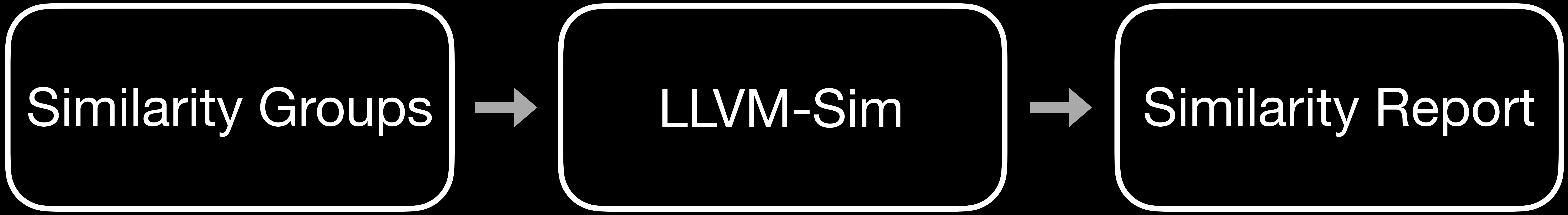
## Applications

## LLVM-Sim

## LLVM-Sim

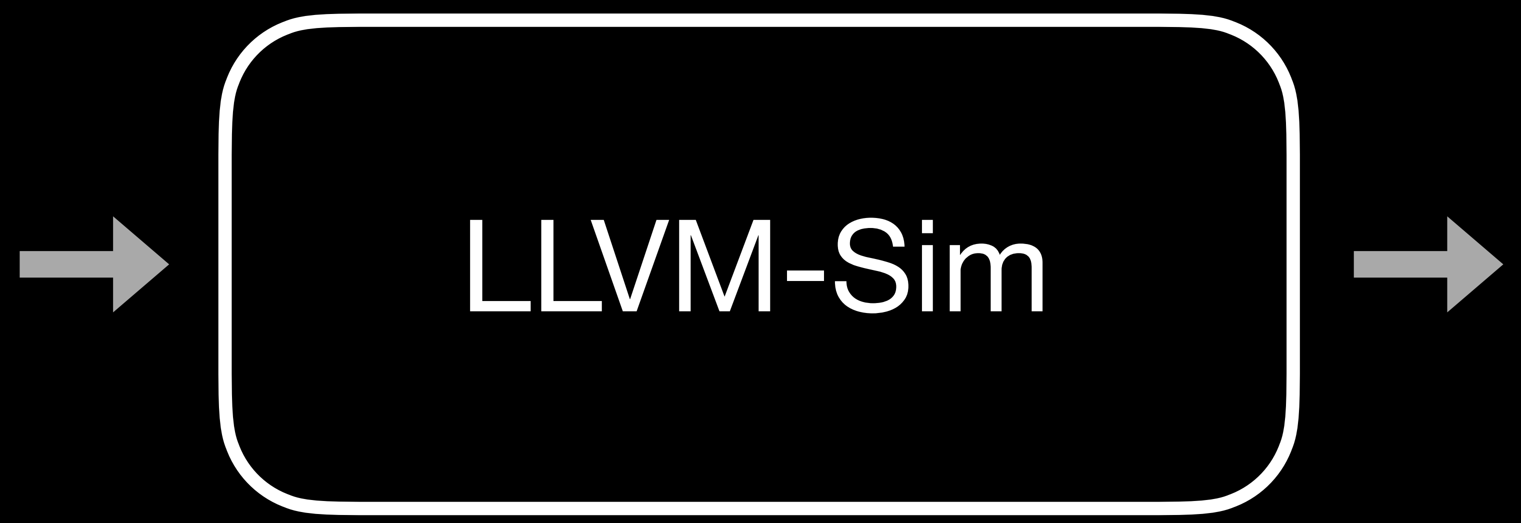


## LLVM-Sim



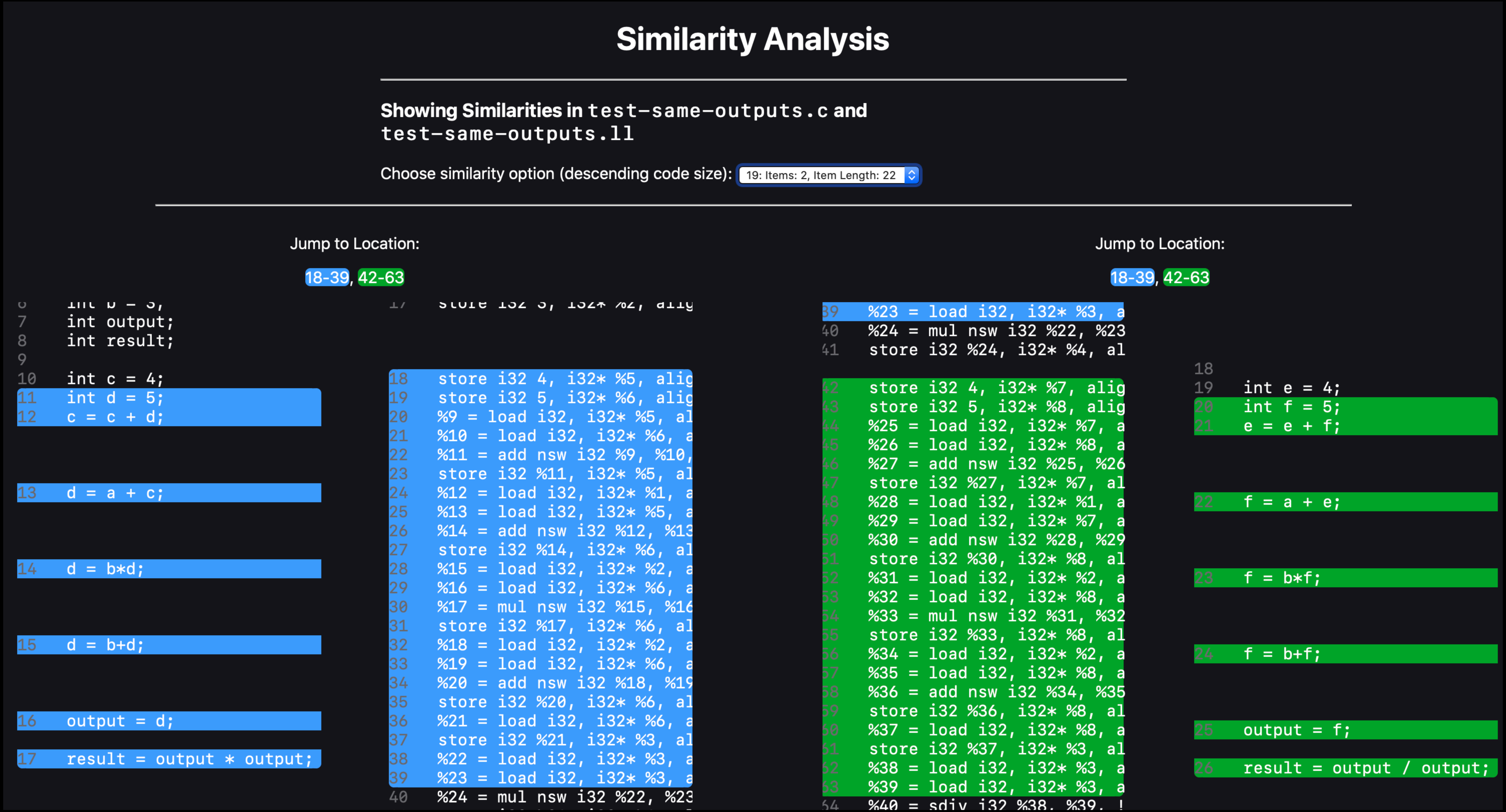
# LLVM-Sim

```
    define i32 @main() #0 {
    entry:
      . . .
4:    store i32 4, i32* %c, align 4
5:    store i32 5, i32* %d, align 4
6:    %0 = load i32, i32* %c, align 4
7:    %1 = load i32, i32* %d, align 4
8:    %add = add nsw i32 %0, %1
9:    store i32 %add, i32* %c, align 4
      . . .
10:   store i32 4, i32* %e, align 4
11:   store i32 5, i32* %f, align 4
12:   %2 = load i32, i32* %e, align 4
13:   %3 = load i32, i32* %f, align 4
14:   %add1 = add nsw i32 %2, %3
15:   store i32 %add1, i32* %e, align 4
16:   ret i32 0
    }
```



```
{
  1: [{"s": 4, "e": 9},
      {"s": 10, "e": 15}],
  2: [{"s": 5, "e": 9},
      {"s": 11, "e": 15}],
  3: [{"s": 6, "e": 9},
      {"s": 12, "e": 15}],
  4: [{"s": 7, "e": 9},
      {"s": 13, "e": 15}],
  5: [{"s": 8, "e": 9},
      {"s": 14, "e": 15}]
}
```

# LLVM-Sim



## Similarity Analysis

**Showing Similarities in** `test-same-outputs.c` **and** `test-same-outputs.ll`

Choose similarity option (descending code size): 19: Items: 2, Item Length: 22

Jump to Location: 18-39, 42-63

Jump to Location: 18-39, 42-63

```
6     int b = 3;
7     int output;
8     int result;
9
10    int c = 4;
11    int d = 5;
12    c = c + d;

13    d = a + c;

14    d = b*d;

15    d = b+d;

16    output = d;

17    result = output * output;
```

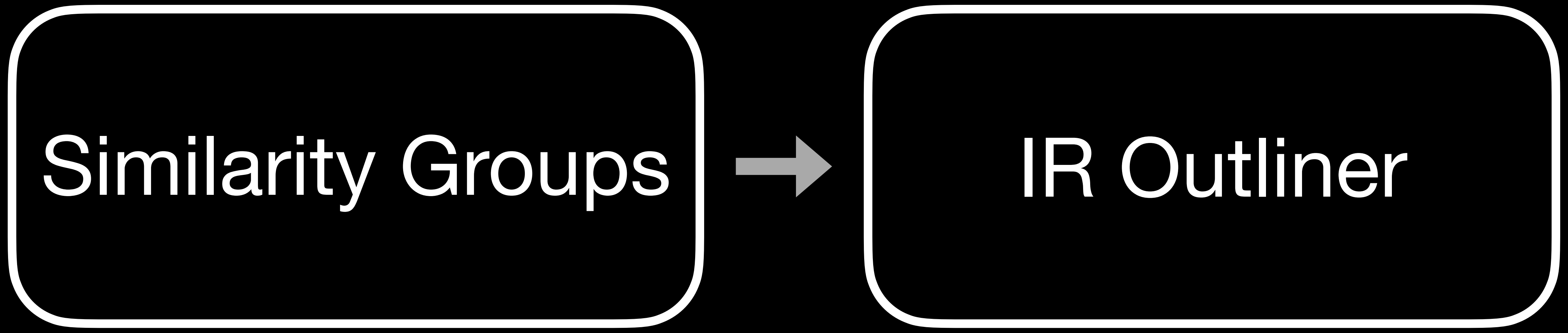
```
17    store i32 3, i32* %2, alig

18    store i32 4, i32* %5, alig
19    store i32 5, i32* %6, alig
20    %9 = load i32, i32* %5, al
21    %10 = load i32, i32* %6, a
22    %11 = add nsw i32 %9, %10,
23    store i32 %11, i32* %5, al
24    %12 = load i32, i32* %1, a
25    %13 = load i32, i32* %5, a
26    %14 = add nsw i32 %12, %13
27    store i32 %14, i32* %6, al
28    %15 = load i32, i32* %2, a
29    %16 = load i32, i32* %6, a
30    %17 = mul nsw i32 %15, %16
31    store i32 %17, i32* %6, al
32    %18 = load i32, i32* %2, a
33    %19 = load i32, i32* %6, a
34    %20 = add nsw i32 %18, %19
35    store i32 %20, i32* %6, al
36    %21 = load i32, i32* %6, a
37    store i32 %21, i32* %3, al
38    %22 = load i32, i32* %3, a
39    %23 = load i32, i32* %3, a
40    %24 = mul nsw i32 %22, %23
```

```
39    %23 = load i32, i32* %3, a
40    %24 = mul nsw i32 %22, %23
41    store i32 %24, i32* %4, al

42    store i32 4, i32* %7, alig
43    store i32 5, i32* %8, alig
44    %25 = load i32, i32* %7, a
45    %26 = load i32, i32* %8, a
46    %27 = add nsw i32 %25, %26
47    store i32 %27, i32* %7, al
48    %28 = load i32, i32* %1, a
49    %29 = load i32, i32* %7, a
50    %30 = add nsw i32 %28, %29
51    store i32 %30, i32* %8, al
52    %31 = load i32, i32* %2, a
53    %32 = load i32, i32* %8, a
54    %33 = mul nsw i32 %31, %32
55    store i32 %33, i32* %8, al
56    %34 = load i32, i32* %2, a
57    %35 = load i32, i32* %8, a
58    %36 = add nsw i32 %34, %35
59    store i32 %36, i32* %8, al
60    %37 = load i32, i32* %8, a
61    store i32 %37, i32* %3, al
62    %38 = load i32, i32* %3, a
63    %39 = load i32, i32* %3, a
64    %40 = sdiv i32 %38, %39, !
```

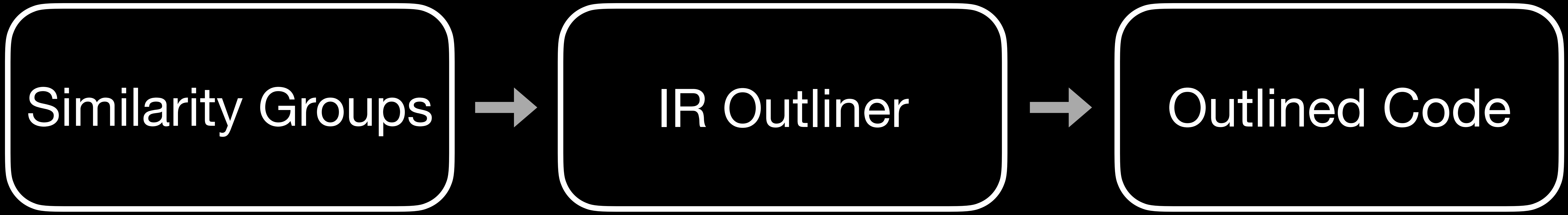
```
18
19    int e = 4;
20    int f = 5;
21    e = e + f;

22    f = a + e;

23    f = b*f;

24    f = b+f;

25    output = f;

26    result = output / output;
```

## IR Outliner

# IR Outliner



# IR Outliner



## IR Outliner

```
%0 = load i32, i32* %a
%add = add i32 %0, 2
%1 = load i32, i32* %b
%add1 = add i32 %1, 3

%sub = sub i32 %add, %v

%0 = load i32, i32* %c
%add2 = add i32 %0, 2
%1 = load i32, i32* %d
%add3 = add i32 %1, 3

%div = div i32 %add3, %v
```

## IR Outliner

```
call void @outlined.1(i32*
  %a, i32* %b, i32* %output1)
%add = load i32, i32* %output1

%sub = sub i32 %add, %v

call void @outlined.2(i32*
  %a, i32* %b, i32* %output2)
%add3 = load i32, i32* %output2

%div = div i32 %add3, %v
```

## IR Outliner

```
call void @outlined_function(i32*
  %a, i32* %b, i32* %output1, i32 0)
%add = load i32, i32* %output1

%sub = sub i32 %add, %v

call void @outlined_function(i32*
  %a, i32* %b, i32* %output2, i32 1)
%add3 = load i32, i32* %output2

%div = div i32 %add3, %v
```

```
define internal void @outlined_function(
  i32* %a, i32* %b, i32* %output, i32 %4) {
  %entry:
    %0 = load i32, i32* %a, align 4
    %add = add i32 %0, 2
    %1 = load i32, i32* %b, align 4
    %add1 = add i32 %1, 3
    switch i32 %4, label %final
        [ i32 0, label %output
          i32 1, label %output_1]
  %output:
    store i32 %add, i32* %output
  %output_1:
    store i32 %add, i32* %output
}
```

## IR Outliner

## IR Outliner

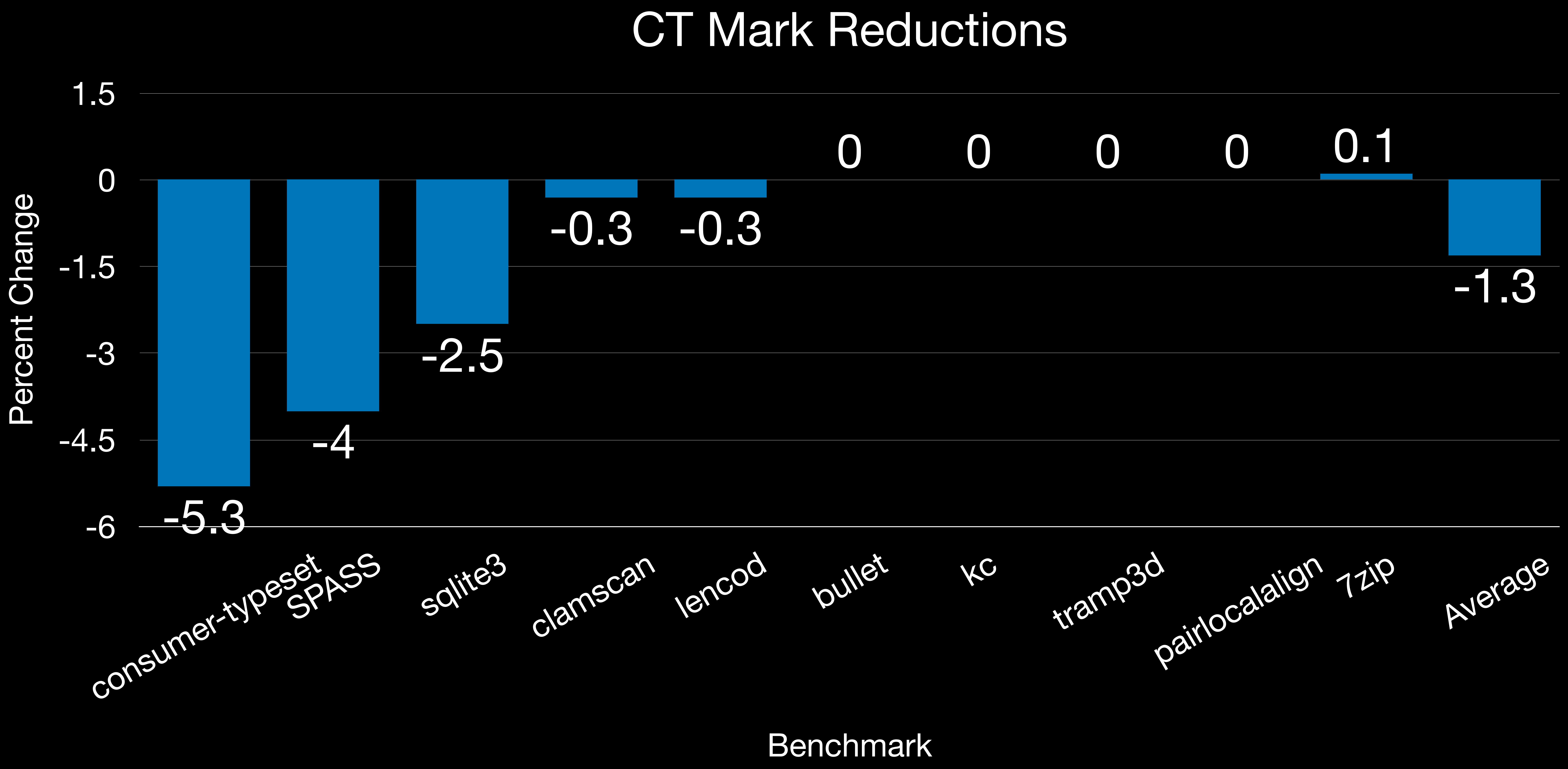
**1%** Average Reduction - LLVM Test Suite

## IR Outliner

**1%** Average Reduction - LLVM Test Suite

**1.3%** Average Reduction - CTMark

# IR Outliner



CT Mark Reductions
Percent Change
1.5
0
-1.5
-3
-4.5
-6
-5.3
-4
-2.5
-0.3
-0.3
0
0
0
0
0.1
-1.3
consumer-typeset
SPASS
sqlite3
clamscan
lencod
bullet
kc
tramp3d
pairlocalalign
7zip
Average
Benchmark

## Related Work

## Related Work

- 2016 - Machine Outliner

## Related Work

- 2016 - Machine Outliner
- 2017 - IR Outliner

## Future Work

- Encompass current Machine Outliner
- Expand to other levels of the compiler

## References

- Machine Outliner
  - http://lists.llvm.org/pipermail/llvm-dev/2016-August/104170.html
- Original IR Outliner
  - http://lists.llvm.org/pipermail/llvm-dev/2017-September/117153.html
- Mailing List for Framework + IR Outliner
  - http://lists.llvm.org/pipermail/llvm-dev/2020-September/144779.html