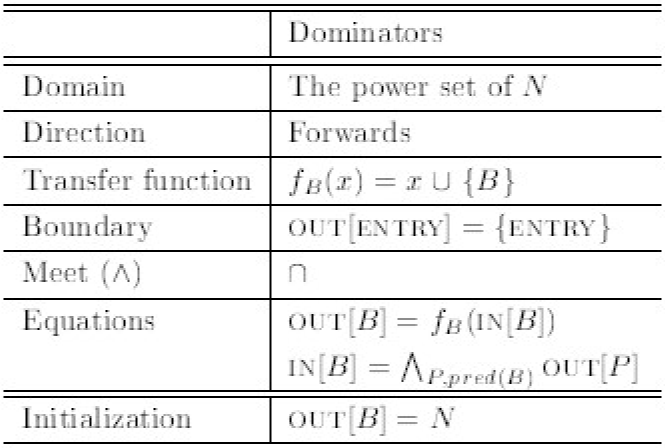
Compiler Optimizations
Assignment 3: Part 1

Group Members: Rohit Mehta, Shambhavi Kuthe
Email: rohitnm@vt.edu, shambhavianil@vt.edu

Dominators:

A node d, is considered to dominate another node, n, if every path from the entry point to n goes through d. It's important to mention that in this situation, every node dominates itself. The immediate dominator for a node is the only node that is the final dominator for that node on any path from the starting point to that node. To determine the dominators for basic blocks (nodes), we utilize the dataflow framework in our Dominators pass implementation. The dataflow algorithm for computing dominators is shown in Fig 1.



	Dominators
Domain	The power set of N
Direction	Forwards
Transfer function	$f_B(x) = x \cup \{B\}$
Boundary	$\text{OUT}[\text{ENTRY}] = \{\text{ENTRY}\}$
Meet ($\wedge$)	$\cap$
Equations	$\text{OUT}[B] = f_B(\text{IN}[B])$ $\text{IN}[B] = \bigwedge_{P,pred(B)} \text{OUT}[P]$
Initialization	$\text{OUT}[B] = N$

Fig 1: Dataflow parameters for Dominators pass

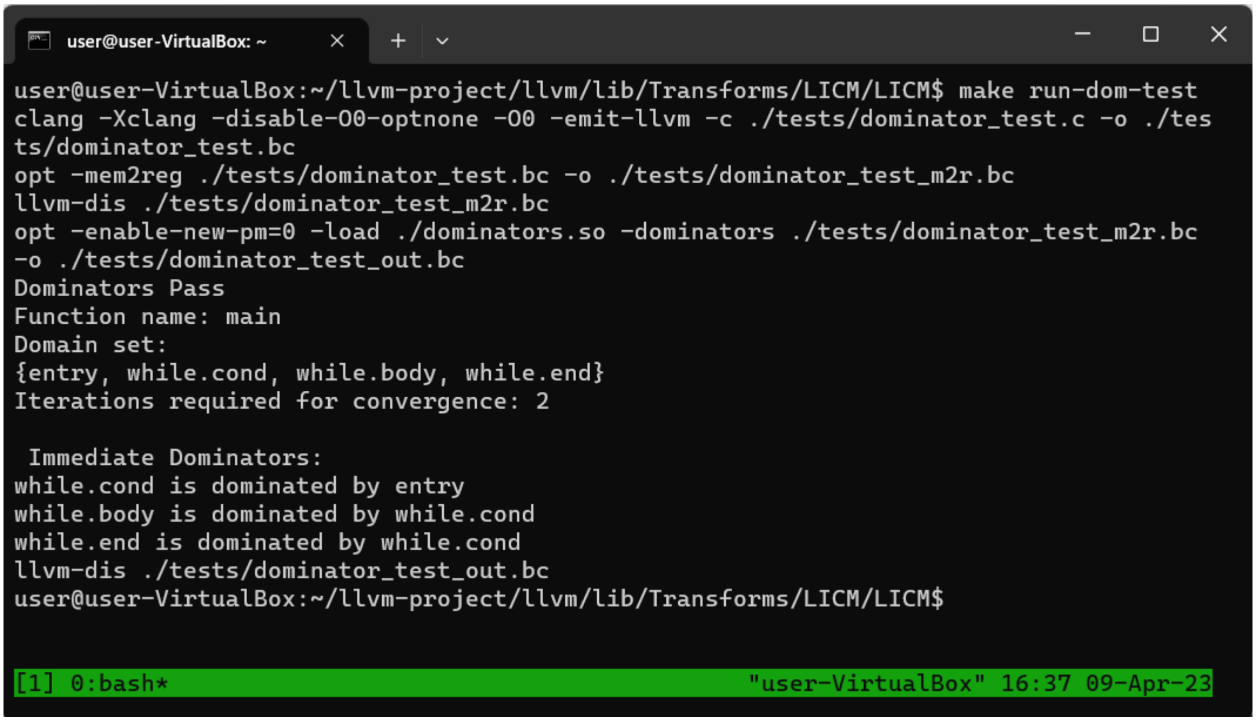
Once the Dominators pass has determined the dominators for each basic block, we can then use these results to find the immediate dominators for nodes. The concept of immediate dominators is based on the idea that a node, m, immediately dominates another node, n, if m is a dominator of n, m is not equal to n, and m does not strictly dominate any of the strict dominators of n.

To find the immediate dominator for a given node, we iterate over the dominators of the basic blocks in the domain. Here, B represents the basic block, and A iterates over its dominators. We check whether either of the following conditions is satisfied:

- A is equal to B
- A dominates C and C dominates B.

If either of these conditions is true, then A is not an immediate dominator of B. If both conditions are false, then A is the immediate dominator of B.

Output:



```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ make run-dom-test
clang -Xclang -disable-O0-optnone -O0 -emit-llvm -c ./tests/dominator_test.c -o ./tes
ts/dominator_test.bc
opt -mem2reg ./tests/dominator_test.bc -o ./tests/dominator_test_m2r.bc
llvm-dis ./tests/dominator_test_m2r.bc
opt -enable-new-pm=0 -load ./dominators.so -dominators ./tests/dominator_test_m2r.bc
-o ./tests/dominator_test_out.bc
Dominators Pass
Function name: main
Domain set:
{entry, while.cond, while.body, while.end}
Iterations required for convergence: 2

 Immediate Dominators:
while.cond is dominated by entry
while.body is dominated by while.cond
while.end is dominated by while.cond
llvm-dis ./tests/dominator_test_out.bc
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$


[1] 0:bash*                                        "user-VirtualBox" 16:37 09-Apr-23
```

Dead Code Elimination (DCE):

The purpose of this pass is to eliminate instructions that are no longer being used (i.e., "dead") while retaining those that are still being used (i.e., "live"), based on the results of faint analysis. The dataflow algorithm for determining which instructions are faint is depicted in Figure 3.

Domain	Set of local variables
Pass Direction	Backwards
Initial	Var
Boundary	Var
Meet Operator	∩ (Intersection)
Transfer Function	f(X) = (X - Kill$_n$ (X)) ∪ Gen$_n$(X)

$$ConstKill_n = \begin{cases} \{x\} & n \text{ is in } use(x) \\ \emptyset & otherwise \end{cases}$$

$$DepKill_n(x) = \begin{cases} Opd(e) \cap Var\,, & n \text{ is assignment } x = e, x \notin X \\ \emptyset, & otherwise \end{cases}$$

$$ConstGen_n = \begin{cases} \{x\}, & n \text{ is assignmnet } x = e, x \notin Opd(e) \\ \{x\}, & n \text{ is } read(x) \\ \emptyset, & otherwise \end{cases}$$

$DepGen_n = \emptyset$

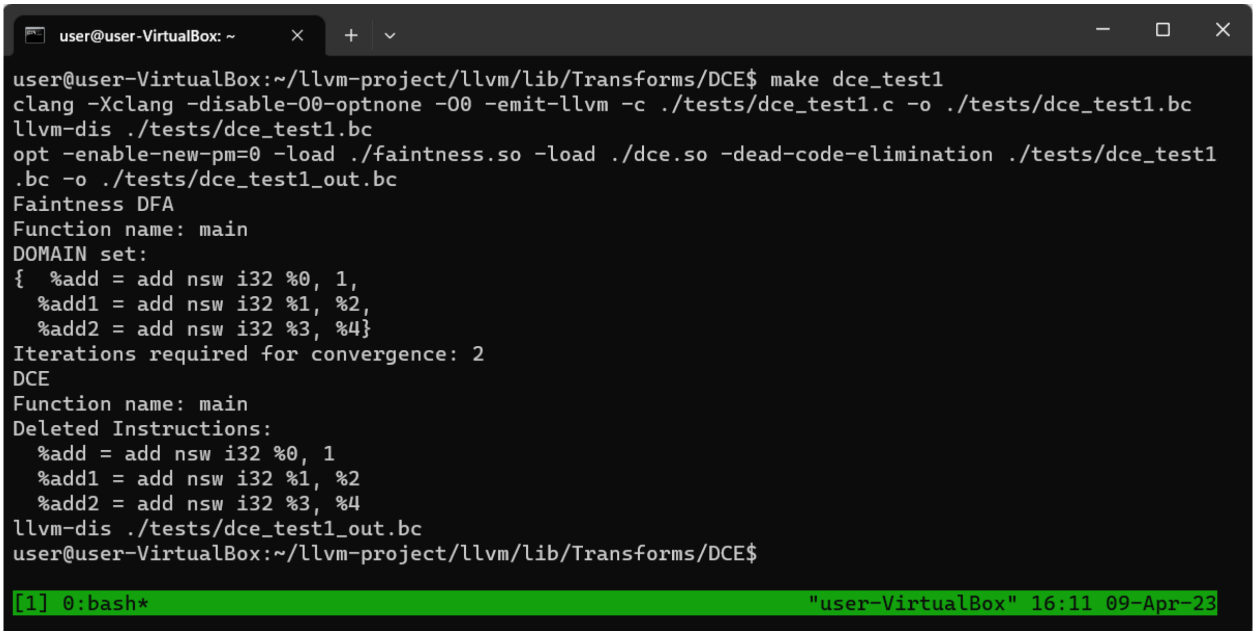
After performing faintness analysis, we use the results to identify which code can be eliminated through dead code elimination (DCE). If a variable x is "live," that means there is a path from program point p to a point where x is used. If there is no such path, then x is considered "dead" and can be eliminated using DCE.

As a part of DCE, we implement the following pseudo code:

```
Get faint analysis results, fa_results and domain set (Var)
for BBs in function:
        Get BB_instr_map (implemented as a vector<BitVector> for BB)
                For Is in BB:
                        Display IN and OUT
                        If I is not Terminator and is faint?
                                Add I to eliminate_set
                Remove faint instructions
```

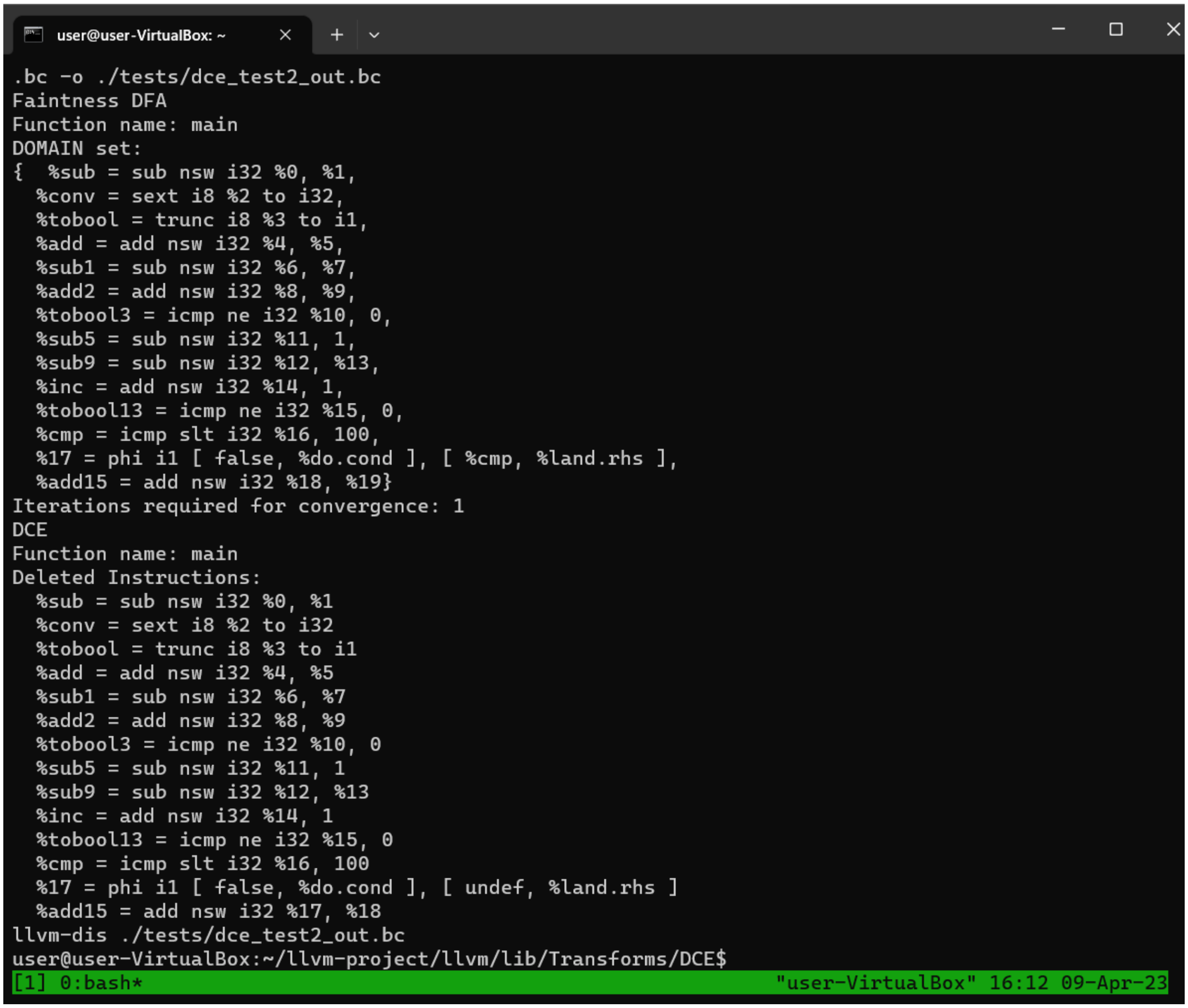
Output:

Test 1 -



```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$ make dce_test1
clang -Xclang -disable-O0-optnone -O0 -emit-llvm -c ./tests/dce_test1.c -o ./tests/dce_test1.bc
llvm-dis ./tests/dce_test1.bc
opt -enable-new-pm=0 -load ./faintness.so -load ./dce.so -dead-code-elimination ./tests/dce_test1
.bc -o ./tests/dce_test1_out.bc
Faintness DFA
Function name: main
DOMAIN set:
{  %add = add nsw i32 %0, 1,
  %add1 = add nsw i32 %1, %2,
  %add2 = add nsw i32 %3, %4}
Iterations required for convergence: 2
DCE
Function name: main
Deleted Instructions:
  %add = add nsw i32 %0, 1
  %add1 = add nsw i32 %1, %2
  %add2 = add nsw i32 %3, %4
llvm-dis ./tests/dce_test1_out.bc
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$

[1] 0:bash*                                                 "user-VirtualBox" 16:11 09-Apr-23
```

Test 2 -



```
.bc -o ./tests/dce_test2_out.bc
Faintness DFA
Function name: main
DOMAIN set:
{  %sub = sub nsw i32 %0, %1,
  %conv = sext i8 %2 to i32,
  %tobool = trunc i8 %3 to i1,
  %add = add nsw i32 %4, %5,
  %sub1 = sub nsw i32 %6, %7,
  %add2 = add nsw i32 %8, %9,
  %tobool3 = icmp ne i32 %10, 0,
  %sub5 = sub nsw i32 %11, 1,
  %sub9 = sub nsw i32 %12, %13,
  %inc = add nsw i32 %14, 1,
  %tobool13 = icmp ne i32 %15, 0,
  %cmp = icmp slt i32 %16, 100,
  %17 = phi i1 [ false, %do.cond ], [ %cmp, %land.rhs ],
  %add15 = add nsw i32 %18, %19}
Iterations required for convergence: 1
DCE
Function name: main
Deleted Instructions:
  %sub = sub nsw i32 %0, %1
  %conv = sext i8 %2 to i32
  %tobool = trunc i8 %3 to i1
  %add = add nsw i32 %4, %5
  %sub1 = sub nsw i32 %6, %7
  %add2 = add nsw i32 %8, %9
  %tobool3 = icmp ne i32 %10, 0
  %sub5 = sub nsw i32 %11, 1
  %sub9 = sub nsw i32 %12, %13
  %inc = add nsw i32 %14, 1
  %tobool13 = icmp ne i32 %15, 0
  %cmp = icmp slt i32 %16, 100
  %17 = phi i1 [ false, %do.cond ], [ undef, %land.rhs ]
  %add15 = add nsw i32 %17, %18
llvm-dis ./tests/dce_test2_out.bc
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$
[1] 0:bash*                                                "user-VirtualBox" 16:12 09-Apr-23
```

Loop invariant Code Motion (LICM):

These are the essential steps that need to be taken in order to perform LICM. By transforming the landing pad, computing reaching definitions and dominators, finding loop exits, and conducting a depth-first search, we can identify and move loop invariant code to the pre-header, where it can be executed only once rather than multiple times within the loop.

- Transform the landing pad
- Compute the reaching definitions
- Compute the dominators
- Find the loop exits
- Conduct a depth-first search of the block, and move any candidates to the pre-header if all the operations that it depends on have also been moved.

➢ ***Landing Pad Transformation:***

To implement the landing pad transformation, we use the LoopPass. This involves a series of steps that assign a new predecessor to each loop header. Specifically, we reassign any edges that enter the header to become in edges for the preheader.

During this transformation, we also consider other factors such as parent loops and phi nodes. By taking these into account, we can ensure that the transformation is performed in a manner that preserves the integrity of the code while effectively moving loop invariant code to the pre-header.

We implemented the landing pad transformation as described in the paper "Global Value Numbers and Redundant Computations", B. K. Rosen, M. N. Wegman, and F. K. Zadeck, ACM, 1988.

The paper describes a method for reducing redundant computations in computer programs by identifying and reusing common sub-expressions. The landing pad transformation is one technique described in the paper that involves inserting code into a program to ensure that any redundant computations are only performed once. The authors show that this technique can lead to significant improvements in program performance by reducing the number of computations that need to be performed.

➢ ***Reaching Definitions:***

This pass was implemented as a part of assignment2. Its implementation is based on the dataflow pass. The figure below shows the algorithm for reaching definitions:

Domain	Set of Definitions
Transfer Function $f_b(x)$	Forward: out[b] = f_b(in[b]) $f_b(x) = Gen_b \cup (x - Kill_b)$ Gen_b: definitions in b $Kill_b$: killed definitions
Meet Operation	in[b] = ∪ out[predecessors]
Boundary Condition	out[entry] = ∅
Initial interior points	out[b] = ∅

➢ ***Dominators:***

In the first part of this assignment, we have already implemented the landing pad transformation pass for LICM. However, in contrast to using only immediate dominators, we use the complete list of dominators to implement the LICM pass. This allows us to identify and move more loop invariant code to the pre-header, thereby improving the efficiency of the code.

➢ ***Loop Invariant Code Motion:***

This pass is implemented as a LoopPass and utilizes information from dominators and reaching definitions. However, the use of reaching definitions is of little use in case of SSA form.

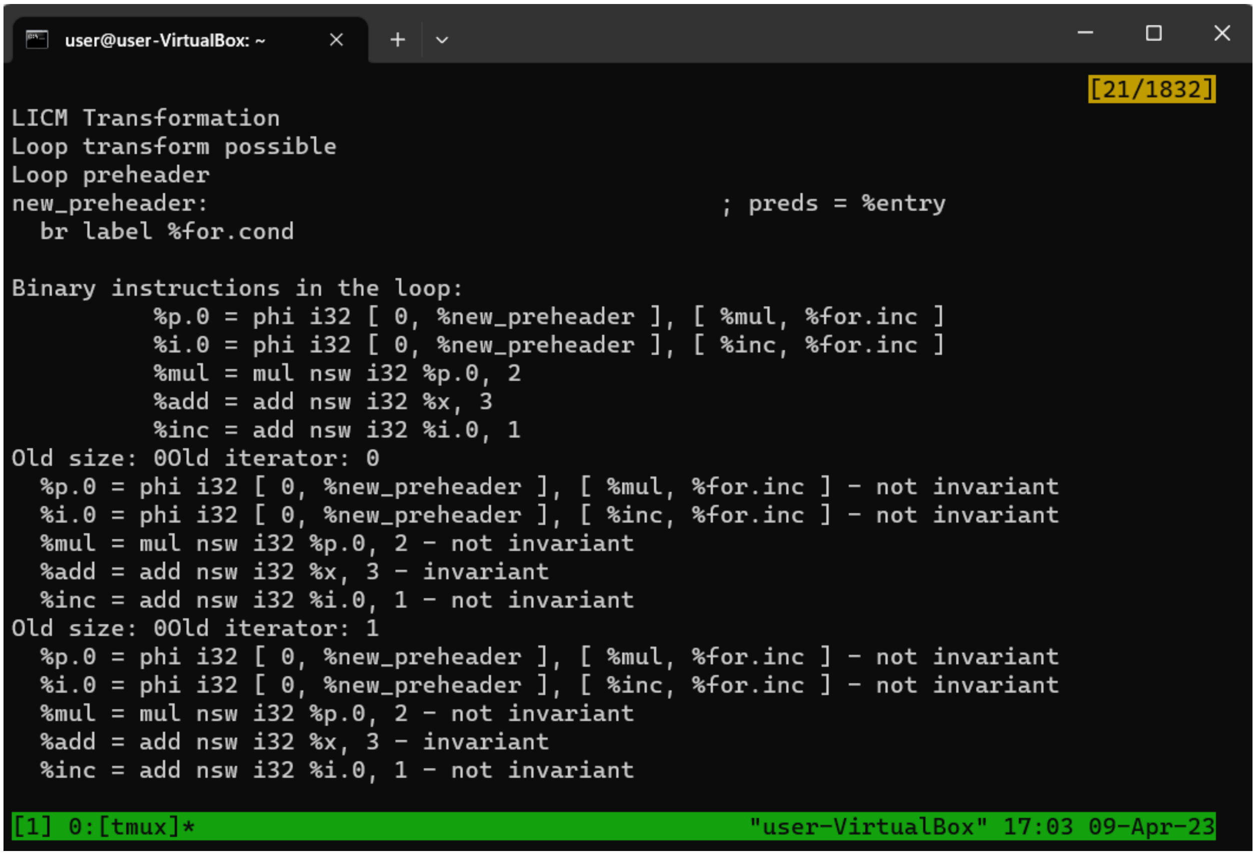
The pass works by iterating over the instructions in the loop until convergence, and determining which instructions are loop invariant. These instructions are then moved to the preheader based on the following conditions:

- If the block hosting the instruction is a dominator of the block where the instruction is being used, the instruction can be safely moved to the preheader.
- If the instruction does not have any uses, it can be moved to the preheader.
- If the instruction is a constant, it can be moved to the preheader.

By moving these loop invariant instructions to the preheader, we can reduce the number of times they need to be executed and improve the efficiency of the code.

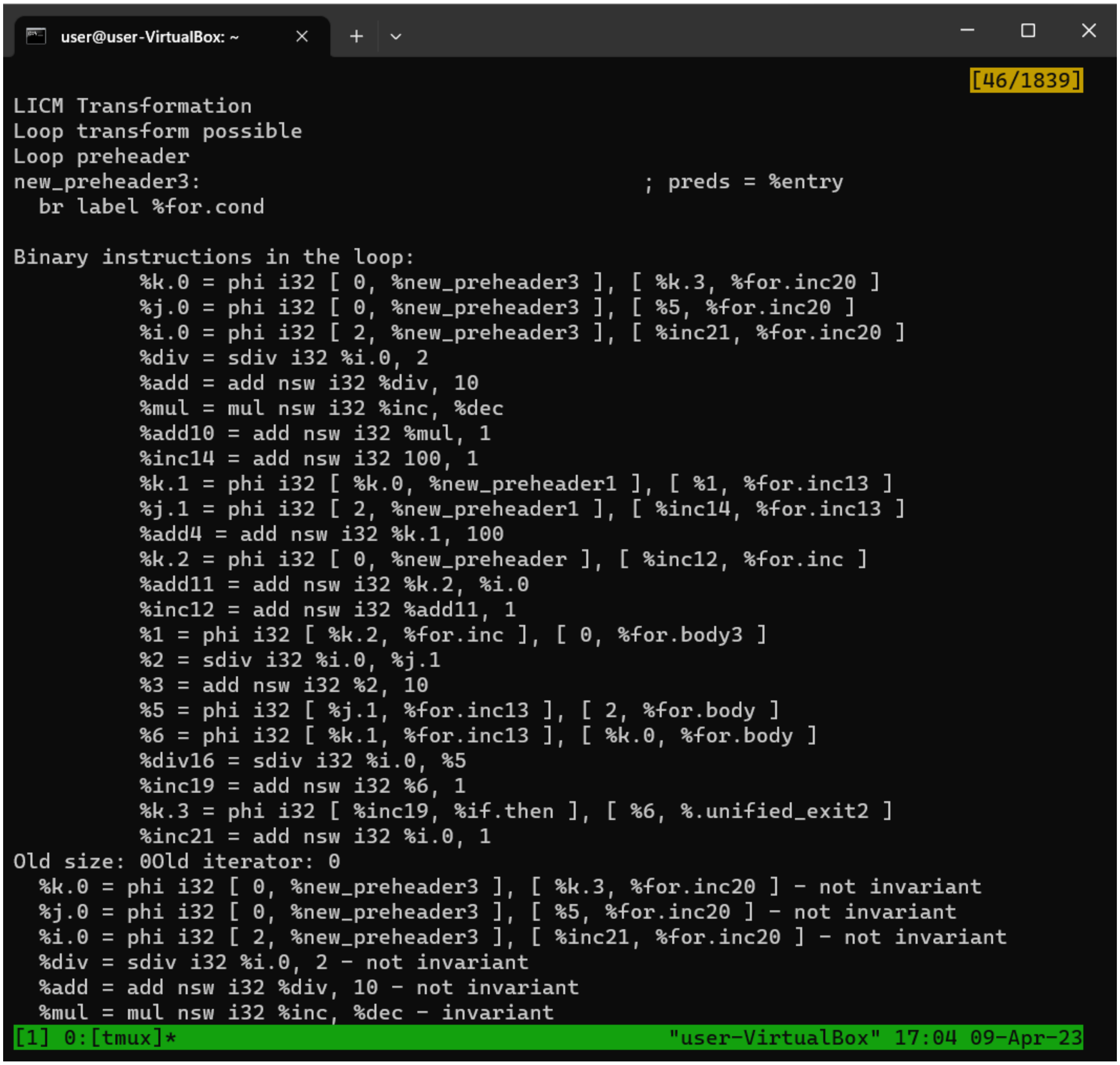
Output:

Test 1 -

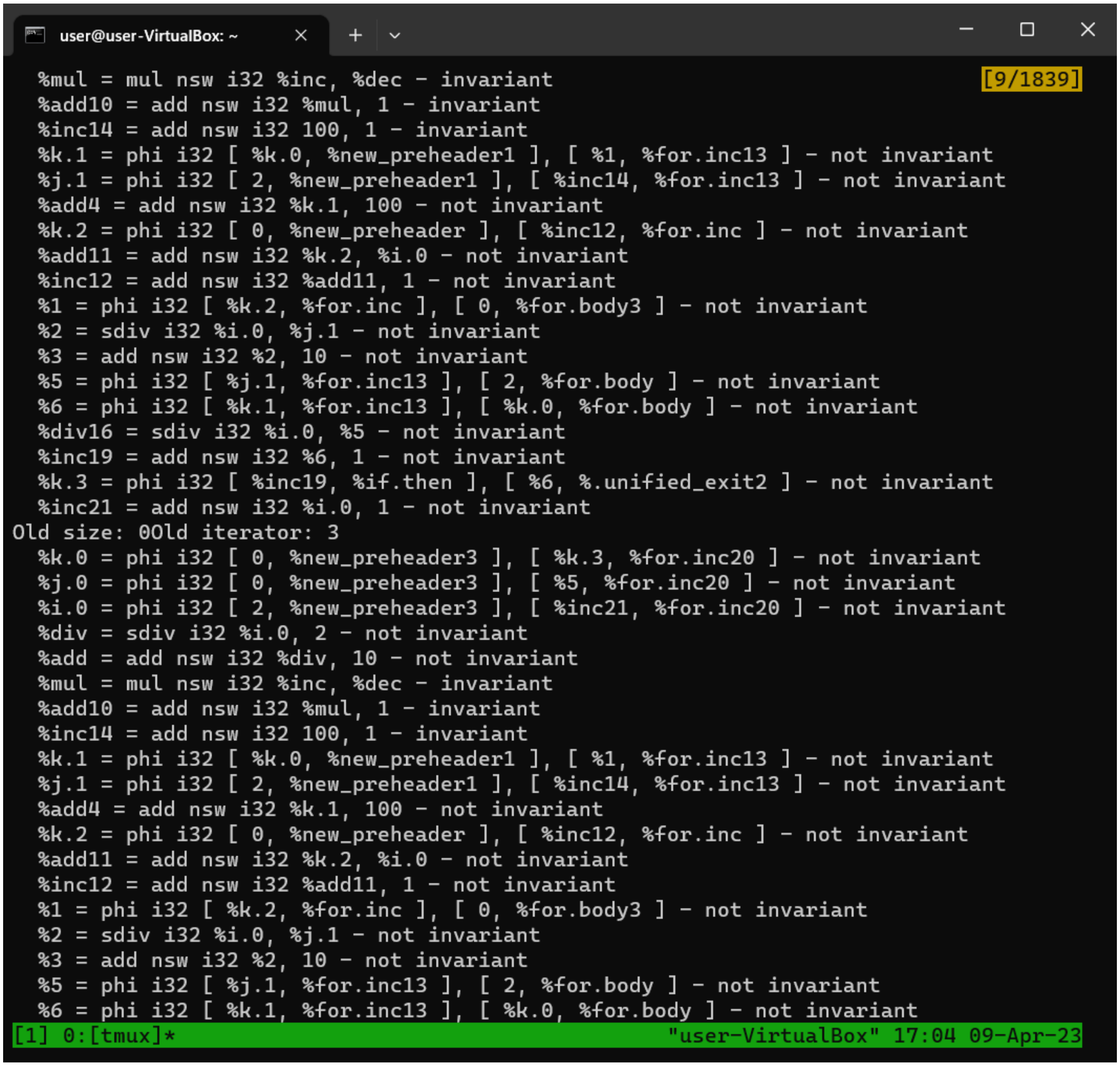


```
LICM Transformation
Loop transform possible
Loop preheader
new_preheader:                                    ; preds = %entry
  br label %for.cond

Binary instructions in the loop:
          %p.0 = phi i32 [ 0, %new_preheader ], [ %mul, %for.inc ]
          %i.0 = phi i32 [ 0, %new_preheader ], [ %inc, %for.inc ]
          %mul = mul nsw i32 %p.0, 2
          %add = add nsw i32 %x, 3
          %inc = add nsw i32 %i.0, 1
Old size: 0Old iterator: 0
  %p.0 = phi i32 [ 0, %new_preheader ], [ %mul, %for.inc ] - not invariant
  %i.0 = phi i32 [ 0, %new_preheader ], [ %inc, %for.inc ] - not invariant
  %mul = mul nsw i32 %p.0, 2 - not invariant
  %add = add nsw i32 %x, 3 - invariant
  %inc = add nsw i32 %i.0, 1 - not invariant
Old size: 0Old iterator: 1
  %p.0 = phi i32 [ 0, %new_preheader ], [ %mul, %for.inc ] - not invariant
  %i.0 = phi i32 [ 0, %new_preheader ], [ %inc, %for.inc ] - not invariant
  %mul = mul nsw i32 %p.0, 2 - not invariant
  %add = add nsw i32 %x, 3 - invariant
  %inc = add nsw i32 %i.0, 1 - not invariant

[1] 0:[tmux]*                                     "user-VirtualBox" 17:03 09-Apr-23
```

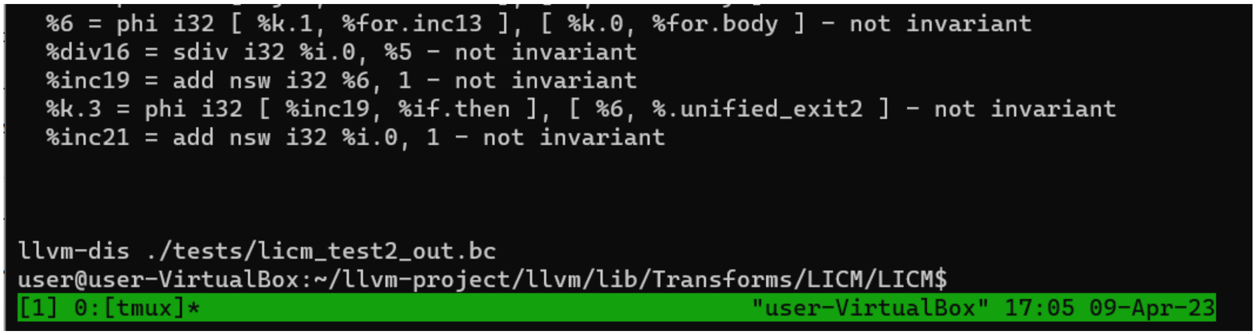
Test 2 -



```
                                                                            [46/1839]
LICM Transformation
Loop transform possible
Loop preheader
new_preheader3:                                   ; preds = %entry
  br label %for.cond

Binary instructions in the loop:
          %k.0 = phi i32 [ 0, %new_preheader3 ], [ %k.3, %for.inc20 ]
          %j.0 = phi i32 [ 0, %new_preheader3 ], [ %5, %for.inc20 ]
          %i.0 = phi i32 [ 2, %new_preheader3 ], [ %inc21, %for.inc20 ]
          %div = sdiv i32 %i.0, 2
          %add = add nsw i32 %div, 10
          %mul = mul nsw i32 %inc, %dec
          %add10 = add nsw i32 %mul, 1
          %inc14 = add nsw i32 100, 1
          %k.1 = phi i32 [ %k.0, %new_preheader1 ], [ %1, %for.inc13 ]
          %j.1 = phi i32 [ 2, %new_preheader1 ], [ %inc14, %for.inc13 ]
          %add4 = add nsw i32 %k.1, 100
          %k.2 = phi i32 [ 0, %new_preheader ], [ %inc12, %for.inc ]
          %add11 = add nsw i32 %k.2, %i.0
          %inc12 = add nsw i32 %add11, 1
          %1 = phi i32 [ %k.2, %for.inc ], [ 0, %for.body3 ]
          %2 = sdiv i32 %i.0, %j.1
          %3 = add nsw i32 %2, 10
          %5 = phi i32 [ %j.1, %for.inc13 ], [ 2, %for.body ]
          %6 = phi i32 [ %k.1, %for.inc13 ], [ %k.0, %for.body ]
          %div16 = sdiv i32 %i.0, %5
          %inc19 = add nsw i32 %6, 1
          %k.3 = phi i32 [ %inc19, %if.then ], [ %6, %.unified_exit2 ]
          %inc21 = add nsw i32 %i.0, 1
Old size: 0Old iterator: 0
  %k.0 = phi i32 [ 0, %new_preheader3 ], [ %k.3, %for.inc20 ] - not invariant
  %j.0 = phi i32 [ 0, %new_preheader3 ], [ %5, %for.inc20 ] - not invariant
  %i.0 = phi i32 [ 2, %new_preheader3 ], [ %inc21, %for.inc20 ] - not invariant
  %div = sdiv i32 %i.0, 2 - not invariant
  %add = add nsw i32 %div, 10 - not invariant
  %mul = mul nsw i32 %inc, %dec - invariant
[1] 0:[tmux]*                                      "user-VirtualBox" 17:04 09-Apr-23
```

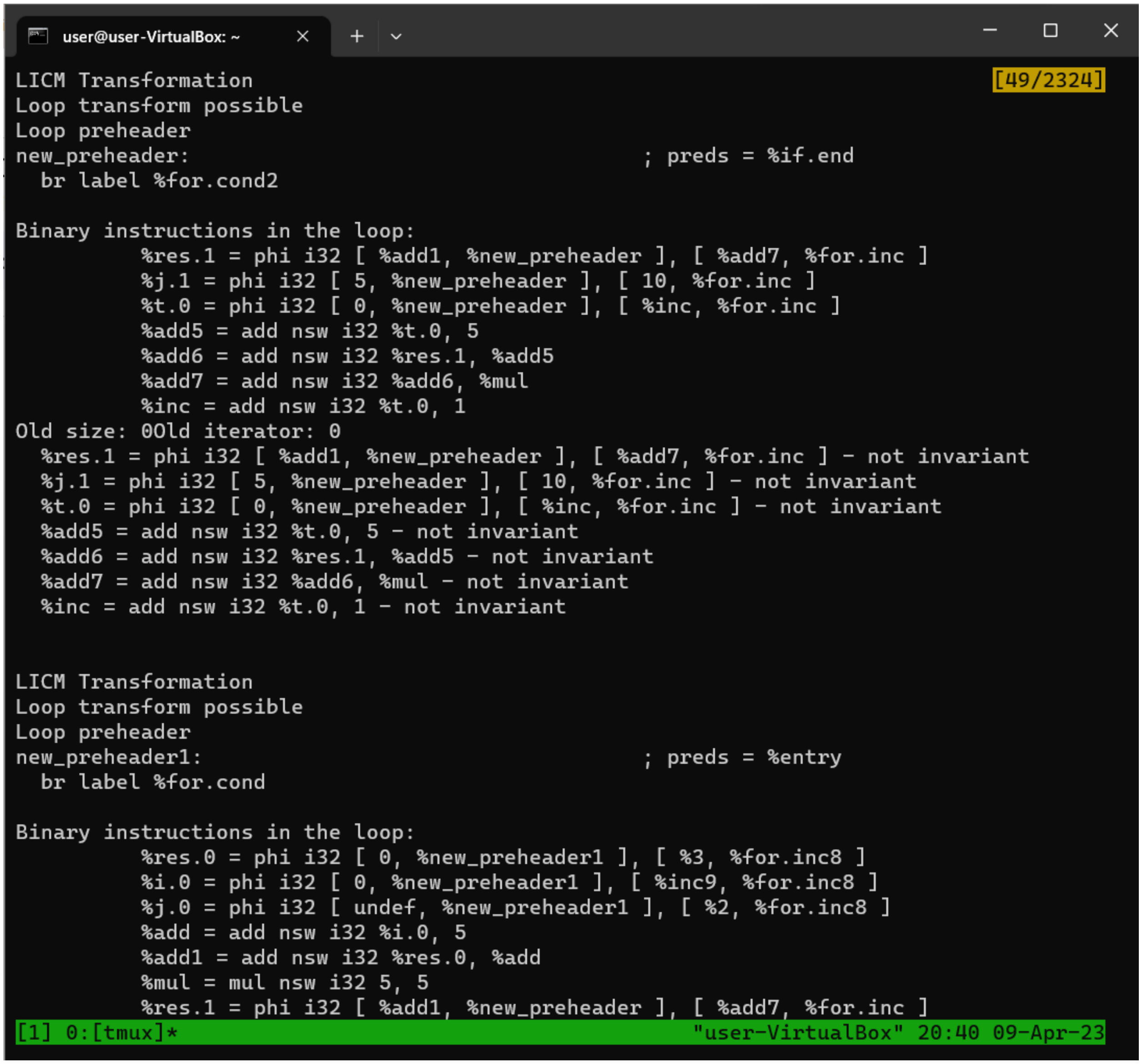


```
  %mul = mul nsw i32 %inc, %dec - invariant                                      [9/1839]
  %add10 = add nsw i32 %mul, 1 - invariant
  %inc14 = add nsw i32 100, 1 - invariant
  %k.1 = phi i32 [ %k.0, %new_preheader1 ], [ %1, %for.inc13 ] - not invariant
  %j.1 = phi i32 [ 2, %new_preheader1 ], [ %inc14, %for.inc13 ] - not invariant
  %add4 = add nsw i32 %k.1, 100 - not invariant
  %k.2 = phi i32 [ 0, %new_preheader ], [ %inc12, %for.inc ] - not invariant
  %add11 = add nsw i32 %k.2, %i.0 - not invariant
  %inc12 = add nsw i32 %add11, 1 - not invariant
  %1 = phi i32 [ %k.2, %for.inc ], [ 0, %for.body3 ] - not invariant
  %2 = sdiv i32 %i.0, %j.1 - not invariant
  %3 = add nsw i32 %2, 10 - not invariant
  %5 = phi i32 [ %j.1, %for.inc13 ], [ 2, %for.body ] - not invariant
  %6 = phi i32 [ %k.1, %for.inc13 ], [ %k.0, %for.body ] - not invariant
  %div16 = sdiv i32 %i.0, %5 - not invariant
  %inc19 = add nsw i32 %6, 1 - not invariant
  %k.3 = phi i32 [ %inc19, %if.then ], [ %6, %.unified_exit2 ] - not invariant
  %inc21 = add nsw i32 %i.0, 1 - not invariant
Old size: 0Old iterator: 3
  %k.0 = phi i32 [ 0, %new_preheader3 ], [ %k.3, %for.inc20 ] - not invariant
  %j.0 = phi i32 [ 0, %new_preheader3 ], [ %5, %for.inc20 ] - not invariant
  %i.0 = phi i32 [ 2, %new_preheader3 ], [ %inc21, %for.inc20 ] - not invariant
  %div = sdiv i32 %i.0, 2 - not invariant
  %add = add nsw i32 %div, 10 - not invariant
  %mul = mul nsw i32 %inc, %dec - invariant
  %add10 = add nsw i32 %mul, 1 - invariant
  %inc14 = add nsw i32 100, 1 - invariant
  %k.1 = phi i32 [ %k.0, %new_preheader1 ], [ %1, %for.inc13 ] - not invariant
  %j.1 = phi i32 [ 2, %new_preheader1 ], [ %inc14, %for.inc13 ] - not invariant
  %add4 = add nsw i32 %k.1, 100 - not invariant
  %k.2 = phi i32 [ 0, %new_preheader ], [ %inc12, %for.inc ] - not invariant
  %add11 = add nsw i32 %k.2, %i.0 - not invariant
  %inc12 = add nsw i32 %add11, 1 - not invariant
  %1 = phi i32 [ %k.2, %for.inc ], [ 0, %for.body3 ] - not invariant
  %2 = sdiv i32 %i.0, %j.1 - not invariant
  %3 = add nsw i32 %2, 10 - not invariant
  %5 = phi i32 [ %j.1, %for.inc13 ], [ 2, %for.body ] - not invariant
  %6 = phi i32 [ %k.1, %for.inc13 ], [ %k.0, %for.body ] - not invariant
[1] 0:[tmux]*                                          "user-VirtualBox" 17:04 09-Apr-23
```



```
  %6 = phi i32 [ %k.1, %for.inc13 ], [ %k.0, %for.body ] - not invariant
  %div16 = sdiv i32 %i.0, %5 - not invariant
  %inc19 = add nsw i32 %6, 1 - not invariant
  %k.3 = phi i32 [ %inc19, %if.then ], [ %6, %.unified_exit2 ] - not invariant
  %inc21 = add nsw i32 %i.0, 1 - not invariant



llvm-dis ./tests/licm_test2_out.bc
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$
[1] 0:[tmux]*                                          "user-VirtualBox" 17:05 09-Apr-23
```

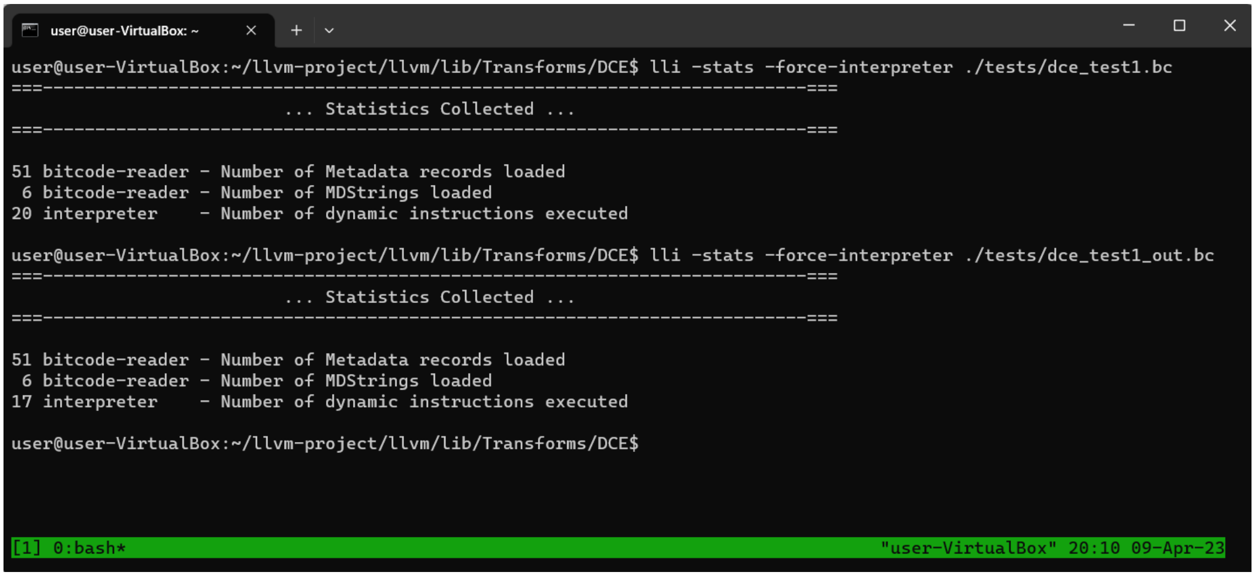
Test 3 -



```
LICM Transformation                                           [49/2324]
Loop transform possible
Loop preheader
new_preheader:                                    ; preds = %if.end
  br label %for.cond2

Binary instructions in the loop:
          %res.1 = phi i32 [ %add1, %new_preheader ], [ %add7, %for.inc ]
          %j.1 = phi i32 [ 5, %new_preheader ], [ 10, %for.inc ]
          %t.0 = phi i32 [ 0, %new_preheader ], [ %inc, %for.inc ]
          %add5 = add nsw i32 %t.0, 5
          %add6 = add nsw i32 %res.1, %add5
          %add7 = add nsw i32 %add6, %mul
          %inc = add nsw i32 %t.0, 1
Old size: 0Old iterator: 0
  %res.1 = phi i32 [ %add1, %new_preheader ], [ %add7, %for.inc ] - not invariant
  %j.1 = phi i32 [ 5, %new_preheader ], [ 10, %for.inc ] - not invariant
  %t.0 = phi i32 [ 0, %new_preheader ], [ %inc, %for.inc ] - not invariant
  %add5 = add nsw i32 %t.0, 5 - not invariant
  %add6 = add nsw i32 %res.1, %add5 - not invariant
  %add7 = add nsw i32 %add6, %mul - not invariant
  %inc = add nsw i32 %t.0, 1 - not invariant


LICM Transformation
Loop transform possible
Loop preheader
new_preheader1:                                   ; preds = %entry
  br label %for.cond

Binary instructions in the loop:
          %res.0 = phi i32 [ 0, %new_preheader1 ], [ %3, %for.inc8 ]
          %i.0 = phi i32 [ 0, %new_preheader1 ], [ %inc9, %for.inc8 ]
          %j.0 = phi i32 [ undef, %new_preheader1 ], [ %2, %for.inc8 ]
          %add = add nsw i32 %i.0, 5
          %add1 = add nsw i32 %res.0, %add
          %mul = mul nsw i32 5, 5
          %res.1 = phi i32 [ %add1, %new_preheader ], [ %add7, %for.inc ]
[1] 0:[tmux]*                                         "user-VirtualBox" 20:40 09-Apr-23
```

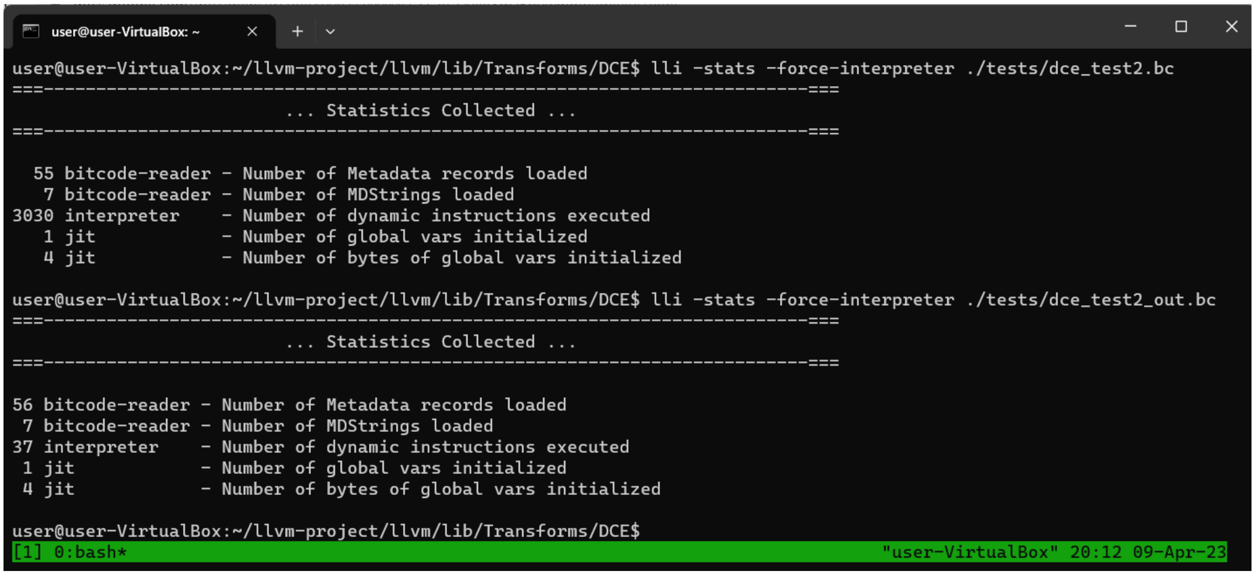
Performance:

Dce test 1



```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$ lli -stats -force-interpreter ./tests/dce_test1.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

51 bitcode-reader - Number of Metadata records loaded
 6 bitcode-reader - Number of MDStrings loaded
20 interpreter    - Number of dynamic instructions executed

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$ lli -stats -force-interpreter ./tests/dce_test1_out.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

51 bitcode-reader - Number of Metadata records loaded
 6 bitcode-reader - Number of MDStrings loaded
17 interpreter    - Number of dynamic instructions executed

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$



[1] 0:bash*                                                                         "user-VirtualBox" 20:10 09-Apr-23
```

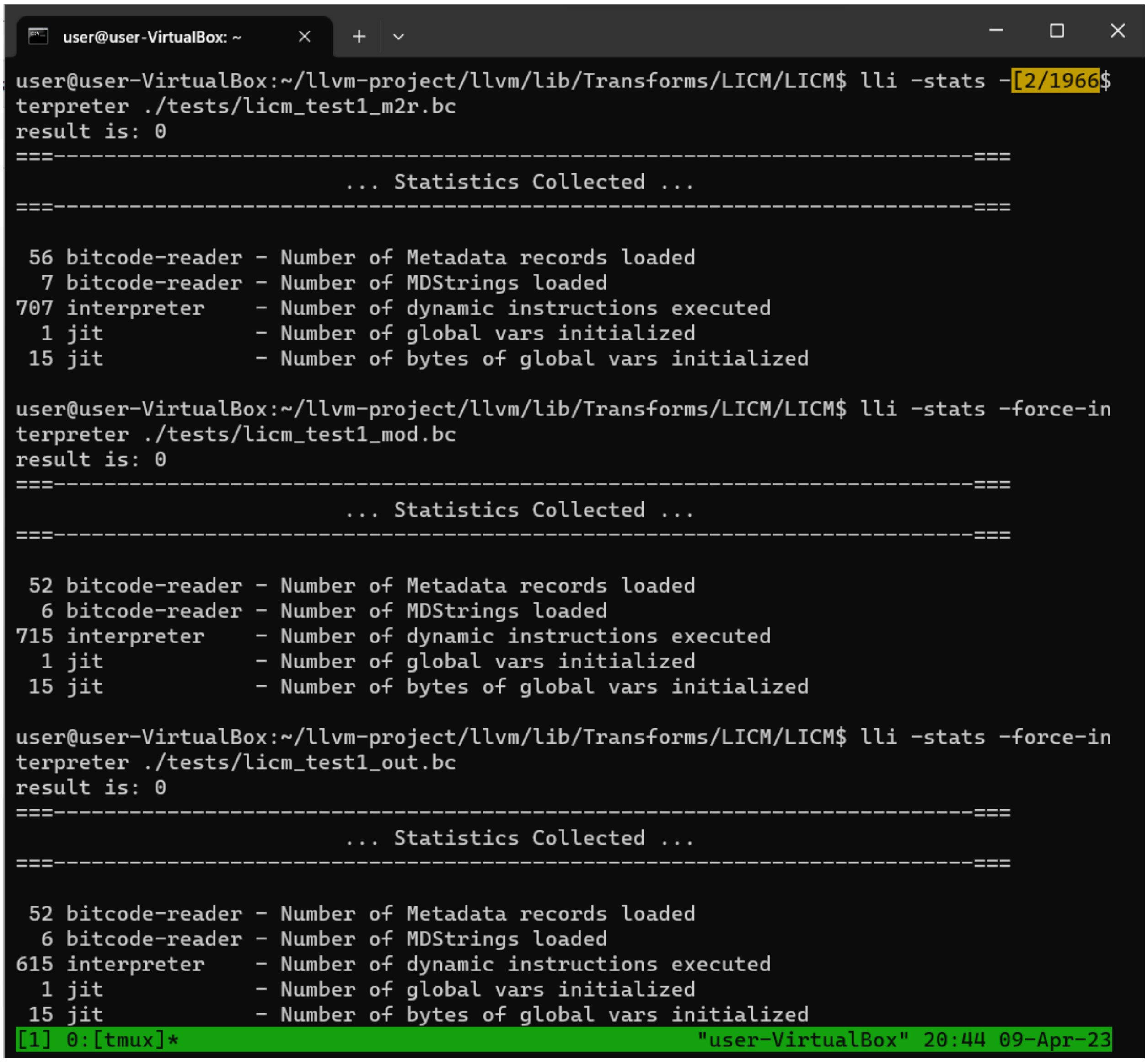
Dce test 2



```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$ lli -stats -force-interpreter ./tests/dce_test2.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

  55 bitcode-reader - Number of Metadata records loaded
   7 bitcode-reader - Number of MDStrings loaded
3030 interpreter    - Number of dynamic instructions executed
   1 jit            - Number of global vars initialized
   4 jit            - Number of bytes of global vars initialized

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$ lli -stats -force-interpreter ./tests/dce_test2_out.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

56 bitcode-reader - Number of Metadata records loaded
 7 bitcode-reader - Number of MDStrings loaded
37 interpreter    - Number of dynamic instructions executed
 1 jit            - Number of global vars initialized
 4 jit            - Number of bytes of global vars initialized

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/DCE$
[1] 0:bash*                                                                         "user-VirtualBox" 20:12 09-Apr-23
```

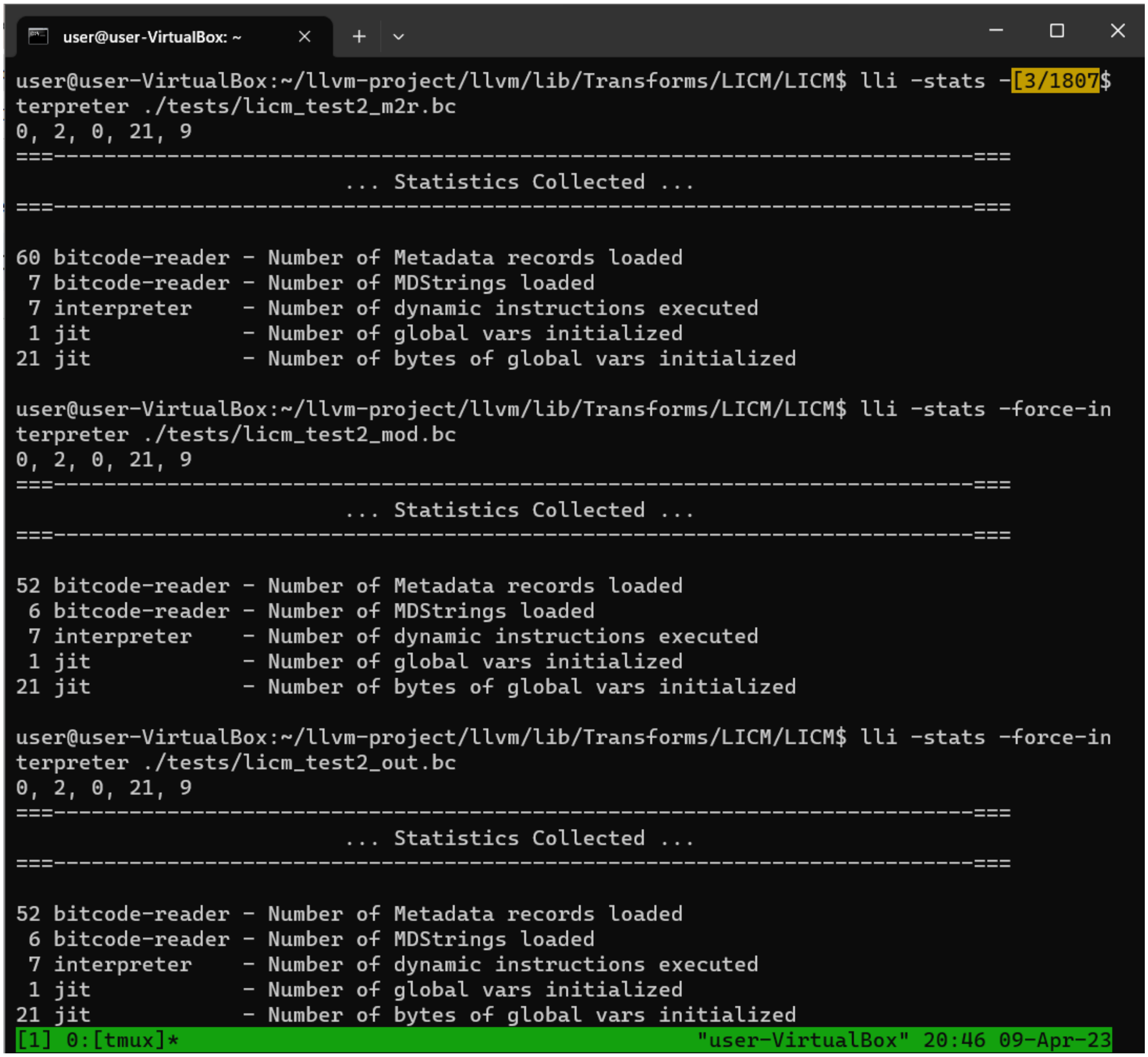
	Dynamic Instruction Count	
Test File	Original	Optimized
dce_test1.c	20	17
dce_test2.c	3030	37

LICM test 1 -



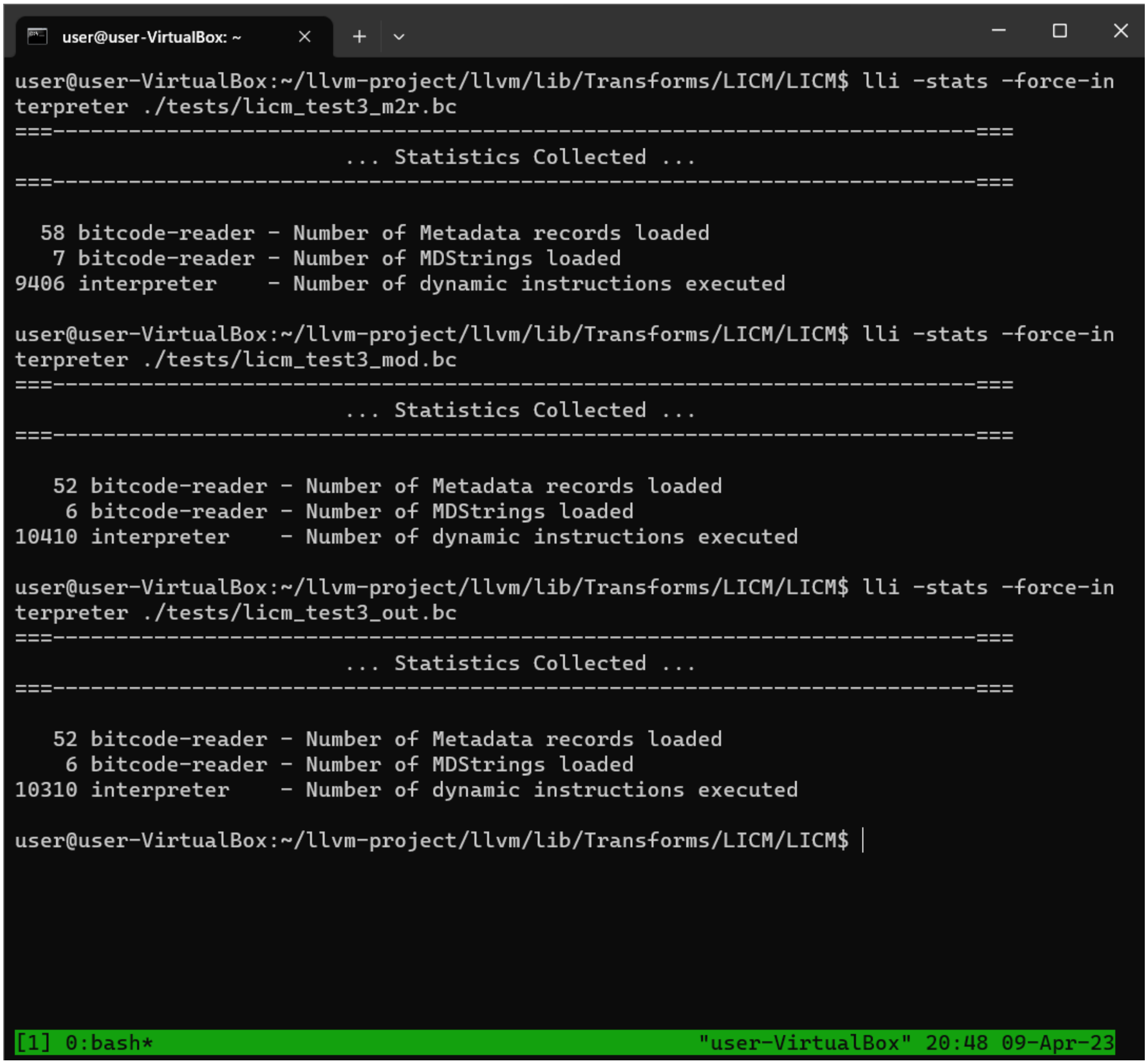
```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -[2/1966$
terpreter ./tests/licm_test1_m2r.bc
result is: 0
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

 56 bitcode-reader - Number of Metadata records loaded
  7 bitcode-reader - Number of MDStrings loaded
707 interpreter    - Number of dynamic instructions executed
  1 jit            - Number of global vars initialized
 15 jit            - Number of bytes of global vars initialized

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test1_mod.bc
result is: 0
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

 52 bitcode-reader - Number of Metadata records loaded
  6 bitcode-reader - Number of MDStrings loaded
715 interpreter    - Number of dynamic instructions executed
  1 jit            - Number of global vars initialized
 15 jit            - Number of bytes of global vars initialized

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test1_out.bc
result is: 0
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

 52 bitcode-reader - Number of Metadata records loaded
  6 bitcode-reader - Number of MDStrings loaded
615 interpreter    - Number of dynamic instructions executed
  1 jit            - Number of global vars initialized
 15 jit            - Number of bytes of global vars initialized
[1] 0:[tmux]*                                        "user-VirtualBox" 20:44 09-Apr-23
```

LICM test 2 -



```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -[3/1807$
terpreter ./tests/licm_test2_m2r.bc
0, 2, 0, 21, 9
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

60 bitcode-reader - Number of Metadata records loaded
 7 bitcode-reader - Number of MDStrings loaded
 7 interpreter    - Number of dynamic instructions executed
 1 jit            - Number of global vars initialized
21 jit            - Number of bytes of global vars initialized

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test2_mod.bc
0, 2, 0, 21, 9
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

52 bitcode-reader - Number of Metadata records loaded
 6 bitcode-reader - Number of MDStrings loaded
 7 interpreter    - Number of dynamic instructions executed
 1 jit            - Number of global vars initialized
21 jit            - Number of bytes of global vars initialized

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test2_out.bc
0, 2, 0, 21, 9
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

52 bitcode-reader - Number of Metadata records loaded
 6 bitcode-reader - Number of MDStrings loaded
 7 interpreter    - Number of dynamic instructions executed
 1 jit            - Number of global vars initialized
21 jit            - Number of bytes of global vars initialized
[1] 0:[tmux]*                                               "user-VirtualBox" 20:46 09-Apr-23
```

LICM test 3 -



```
user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test3_m2r.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

  58 bitcode-reader - Number of Metadata records loaded
   7 bitcode-reader - Number of MDStrings loaded
9406 interpreter    - Number of dynamic instructions executed

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test3_mod.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

   52 bitcode-reader - Number of Metadata records loaded
    6 bitcode-reader - Number of MDStrings loaded
10410 interpreter    - Number of dynamic instructions executed

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ lli -stats -force-in
terpreter ./tests/licm_test3_out.bc
===-------------------------------------------------------------------------===
                          ... Statistics Collected ...
===-------------------------------------------------------------------------===

   52 bitcode-reader - Number of Metadata records loaded
    6 bitcode-reader - Number of MDStrings loaded
10310 interpreter    - Number of dynamic instructions executed

user@user-VirtualBox:~/llvm-project/llvm/lib/Transforms/LICM/LICM$ |





[1] 0:bash*                                       "user-VirtualBox" 20:48 09-Apr-23
```

	Dynamic Instruction Count	
Test File	Original (Mod)	Optimized
licm_test1.c	715	615
licm_test2.c	7	7
licm_test3.c	10410	10310