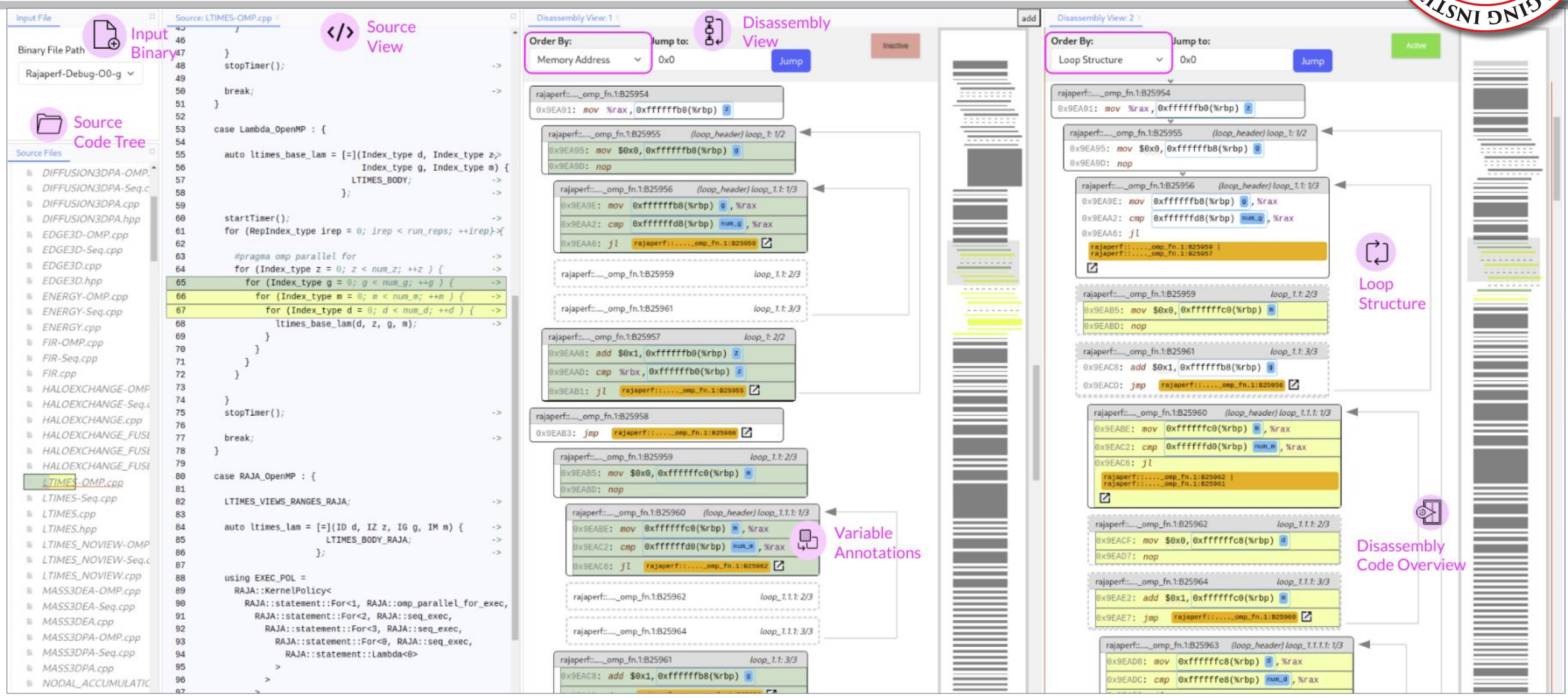
Interactive Visualization of Binary Code for Compiler Optimizations

Shadmaan Hye and Katherine E. Isaacs



Problem Statement

Program analysts spend huge amount of time navigating large binary codes to improve compiler performance. Assembly code is difficult to understand for lack of intuitive correlation with the source code, hampering compiler optimization.

Solution: ***DisViz*** aids understanding through interactive visual analysis of source and assembly code relationships.

Providing Structure to Disassembly Code

We aim to support large binaries. The example above has more than 260,000 assembly instructions, 1,000 source files & 3,000 loops.

```
int x = 0, a = 0;
while (x < 5) {
    a += 2;
    x++;
}
```

Source Code

```
0x1119: push %rbp
0x111A: mov  %rsp, %rbp
0x1126: movl $0, -8(%rbp)
0x112D: movl $0, -4(%rbp)
0x112E: jmp  0x1133
0x112F: addl $2, -4(%rbp)
0x1132: addl $1, -8(%rbp)
0x1133: cmp  $4, -8(%rbp)
0x1134: jle  0x112F
0x1135: movl $0, %eax
0x1136: pop  %rbp
0x1137: ret
```

Assembly Code

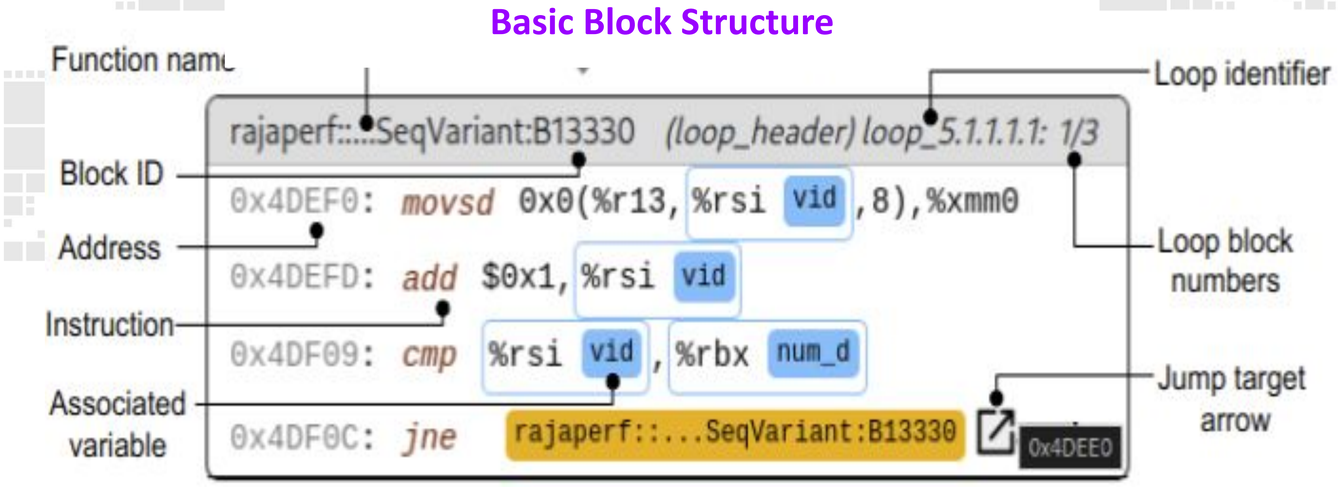
```
0x1119: push %rbp
0x111A: mov  %rsp, %rbp
0x1126: movl $0, -8(%rbp)
0x112D: movl $0, -4(%rbp)
0x112E: jmp  0x1133

0x112F: addl $2, -4(%rbp)
0x1132: addl $1, -8(%rbp)

0x1133: cmp  $4, -8(%rbp)
0x1134: jle  0x112F
0x1135: movl $0, %eax
0x1136: pop  %rbp
0x1137: ret
```

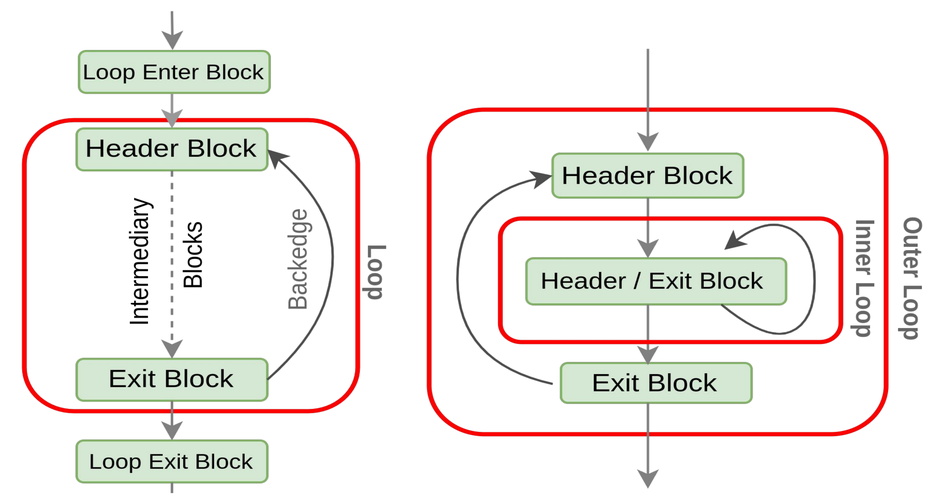
Assembly Code with *Basic blocks*

We visualize assembly code structured in basic blocks:

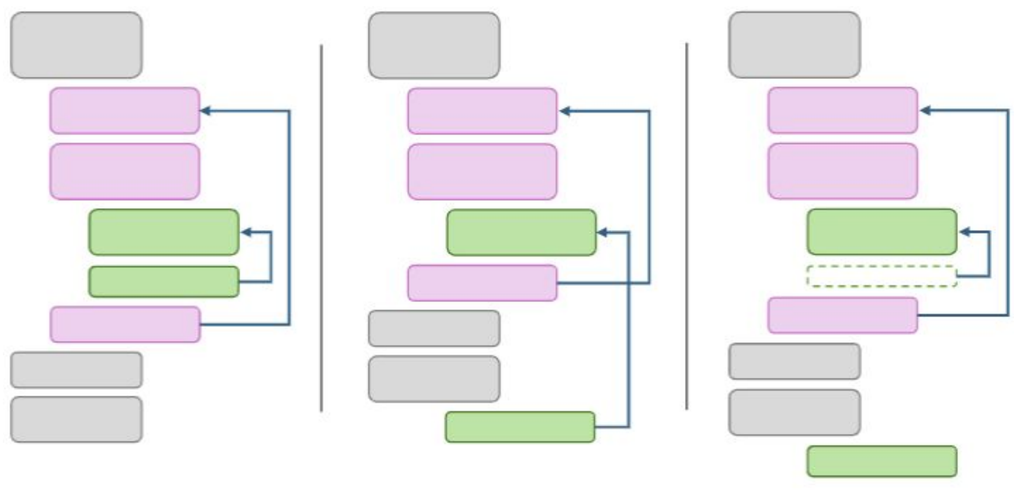


Novel Layout of Loops in Disassembly

Designed the loop blocks according to this loop design:



Design phase of pseudo blocks solving corner cases.



Designing Orders of Disassembly View

After pseudo blocks, it was observed that another order was necessary to understand control flow.
The example above shows the 2 orders:
Memory Address: The original order in which the block appears in the memory layout.
Loop Structure: The designed order with loop nesting.

THE UNIVERSITY OF UTAH
SCI www.sci.utah.edu