Binary Analysis for Missed Vectorization Opportunities Detection

AMOS HERZ and ALESSANDRO LEGNANI, ETH Zürich, Switzerland

ACM Reference Format:
Amos Herz and Alessandro Legnani. 2024. Binary Analysis for Missed Vectorization Opportunities Detection. 1, 1 (March 2024), 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Modern compilers are often able to automatically vectorize code using SIMD instructions. Take, as an example, the following code snippet:

```
void copy(long *restrict a, long *restrict b, unsigned long n) {
    for (unsigned long i = 0ul; i < n; i++) {
        a[i] = b[i];
    }
}
```

We can compile it with the following set of compiler flags

- `-O3`: tells the compiler to use the highest level of optimization available.
- `-fno-tree-loop-distribute-patterns`: prevents replacing the loop with a call to `memcpy`
- `-fno-tree-vectorize`: prevents vectorization

Which will produce the following assembly code:

```
.L3:
    movq  (%rsi,%rax,8), %rcx
    movq  %rcx, (%rdi,%rax,8)
    addq  $1, %rax
    cmpq  %rax, %rdx
    jne   .L3
```

However, by compiling without the `-fno-tree-vectorize` flag, the compiler will produce the following vectorized code (note the use of wider instructions and registers):

```
.L4:
    movdqu  (%rsi,%rax), %xmm0
    movups  %xmm0, (%rdi,%rax)
    addq    $16, %rax
    cmpq    %rcx, %rax
    jne     .L4
```

REFERENCES

Authors' address: Amos Herz, amherz@ethz.ch; Alessandro Legnani, alegnani@ethz.ch, ETH Zürich, Zürich, Switzerland, 8092.

ACM XXXX-XXXX/2024/3-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn