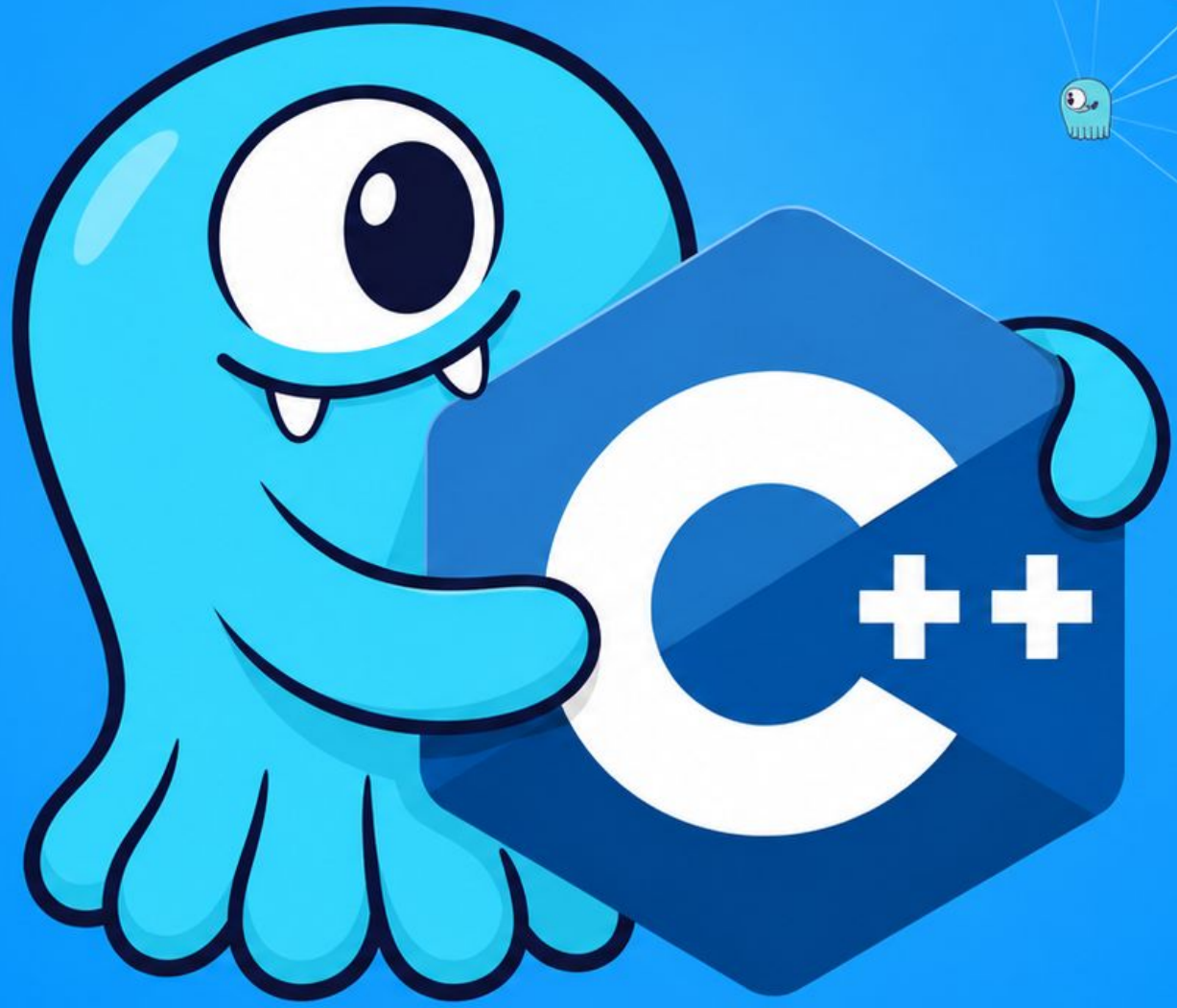
The Compiler Knows Best




I know better!

About Me:



CONNECT ON

LinkedIn

Alex Dathskovsky

SCAN TO CONNECT

linkedin.com/in/alexdathskovsky

VISIT MY WEBSITE

CPPNEXT



SCAN TO EXPLORE

www.cppnext.com

WATCH ON

YouTube

@cppnext-alexd

SCAN TO SUBSCRIBE

youtube.com/@cppnext-alexd

LISTEN ON SUNO

@calebxyz376



SCAN TO LISTEN

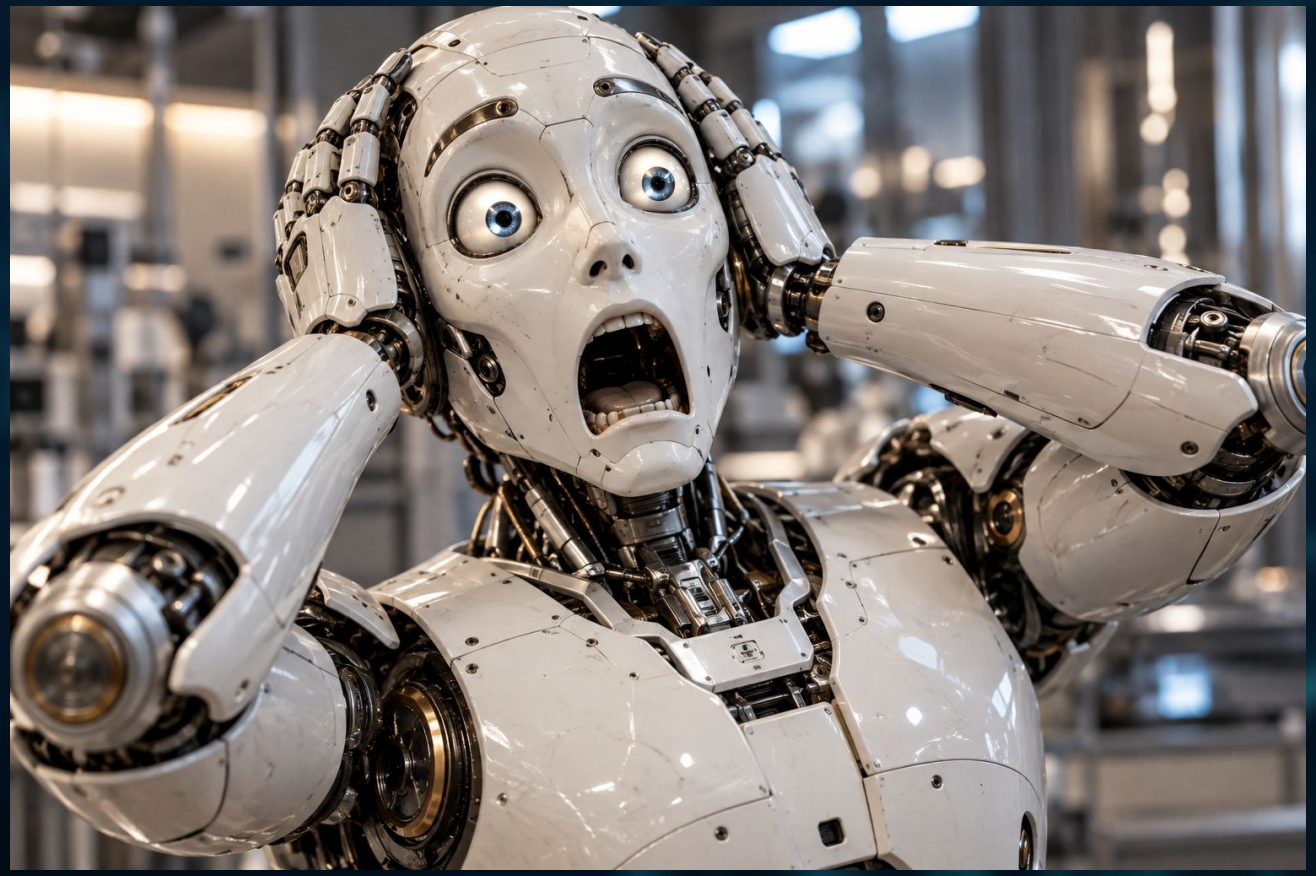
suno.com/@calebxyz376

Most Important Gotcha

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Most Important Gotcha

The program we write (usually) is not the program that runs!



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Most Important Gotcha

- The compiler must preserve the observable behavior required by C++:
  - Compilers reorder and change operations
  - Compilers simplify the code and make it more performant
  - CPU's use threading and OOO execution



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

What Would You Do If You Were The Compiler

```
3   template <unsigned N>
4   unsigned calculate(std::array<int, N> data, unsigned pivot,
5                      unsigned factor)
6   {
7       unsigned total = 0;
8
9       for (unsigned i = 0; i < N; ++i) {
10          unsigned weight = (factor * factor) + data[pivot];
11          total += data[i] * weight;
12      }
13
14      return total;
15  }
```



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

What Would You Do If You Were The Compiler

```
3   template <unsigned N>
4   unsigned calculate(std::array<int, N> data, unsigned pivot,
5                      unsigned factor)
6   {
7       unsigned total = 0;
8       const auto p = data[pivot];
9
10      for (unsigned i = 0; i < N; ++i) {
11          unsigned weight = (factor * factor) + p;
12          total += data[i] * weight;
13      }
14
15      return total;
16  }
```



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

What Would You Do If You Were The Compiler

```
3   template <unsigned N>
4   unsigned calculate(std::array<int, N> data, unsigned pivot,
5                      unsigned factor)
6   {
7       unsigned total = 0;
8       const auto p = data[pivot];
9       const auto weight = (factor * factor) + p;
10
11      for (unsigned i = 0; i < N; ++i) {
12          total += data[i] * weight;
13      }
14
15      return total;
16  }
```



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

What Would You Do If You Were The Compiler

```
3   template <unsigned N>
4   unsigned calculate(std::array<int, N> data, unsigned pivot,
5                      unsigned factor)
6   {
7       unsigned total = 0;
8       const auto p = data[pivot];
9       const auto weight = (factor * factor) + p;
10
11      for (unsigned i = 0; i < N; ++i) {
12          total += data[i];
13      }
14
15      return total*weight;
16  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Where Does The Magic Happens?

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Where Compiler Optimizations Happen



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Algebraic Simplifications

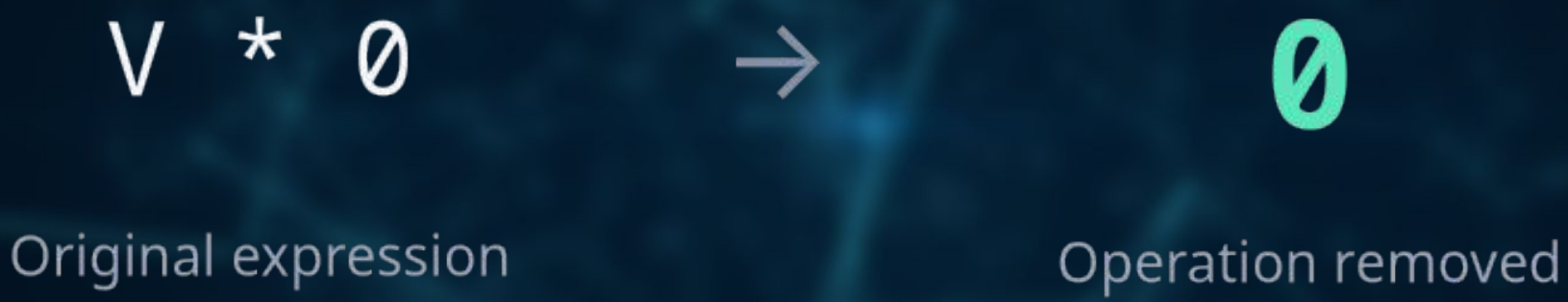
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Algebraic Simplifications: Remove Unnecessary Operations



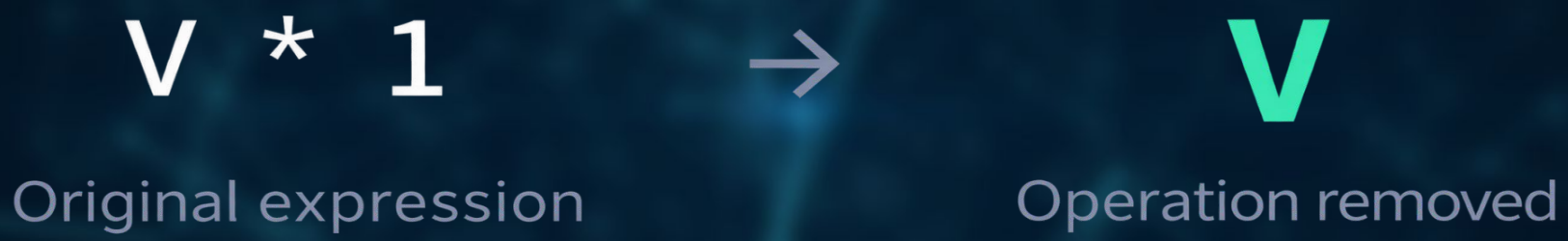
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Algebraic Simplifications: Result is Known



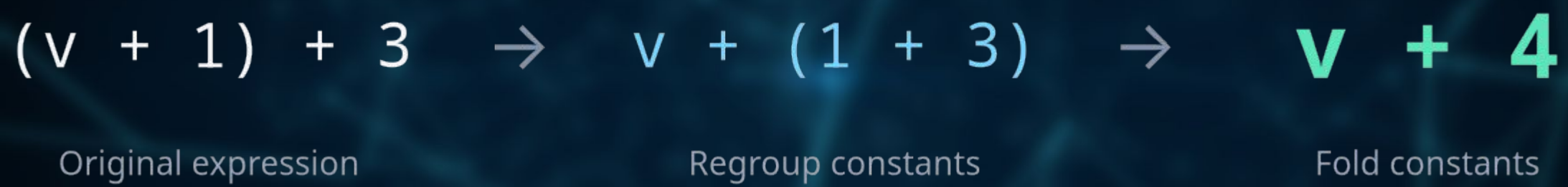
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Algebraic Simplifications: Result is Known



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

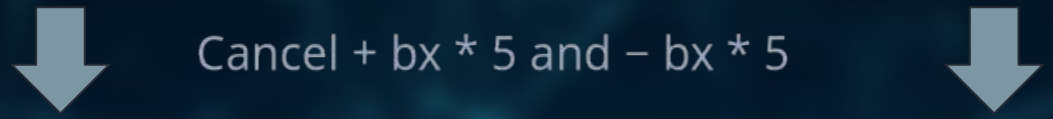
Algebraic Simplifications: constant Folding



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

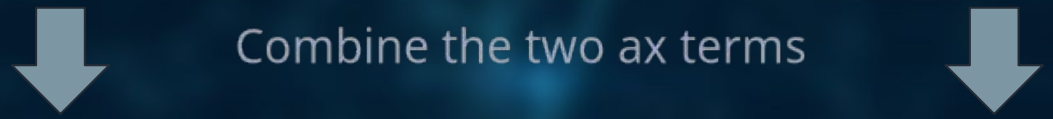
Algebraic Simplifications: Cancelation and Folding

```
((ax * 5 + bx * 5) - bx * 5 + ax * 5) * 2
```



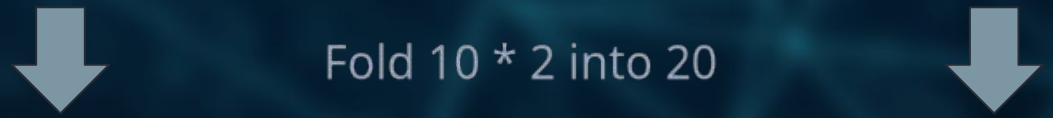
Cancel + bx * 5 and – bx * 5

```
(ax * 5 + ax * 5) * 2
```



Combine the two ax terms

```
(ax * 10) * 2
```



Fold 10 * 2 into 20

```
ax * 20
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Global Value Numbering

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Global Value Numbering

```
uint64_t recalc(uint64_t a, uint64_t b,
                uint64_t c){
    uint64_t x = a * b;
    uint64_t y = a * b + c;
    return x ^ y;
}
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Global Value Numbering

```
uint64_t recalc(uint64_t a, uint64_t b,
                uint64_t c){
    auto product = a * b;
    return product ^ (product + c);
}
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Global Value Numbering

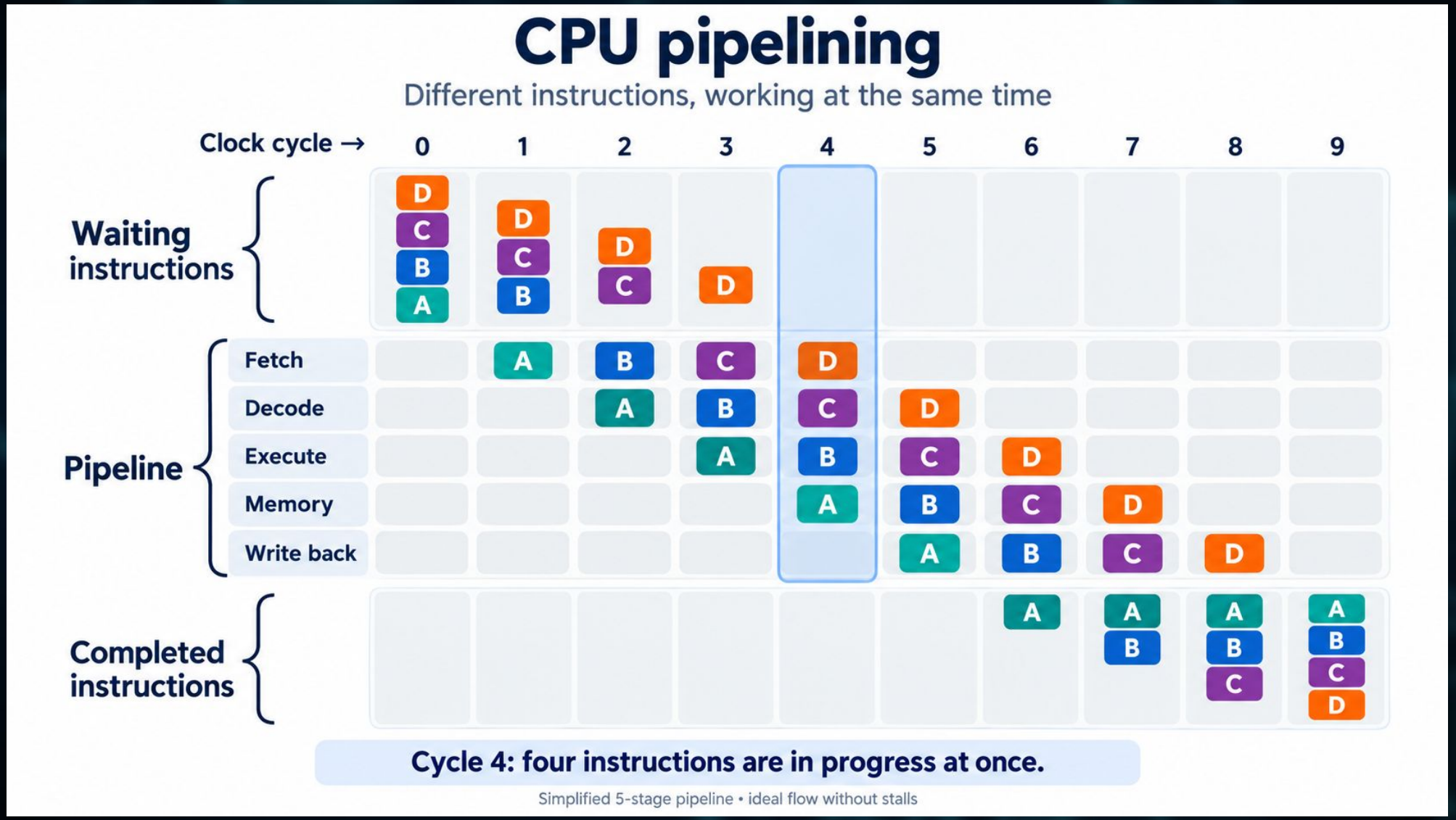
```
%4 = mul i64 %1, %0
%5 = add i64 %4, %2
%6 = xor i64 %5, %4
ret i64 %6
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

CPU Pipeline Detour

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

CPU Pipeline

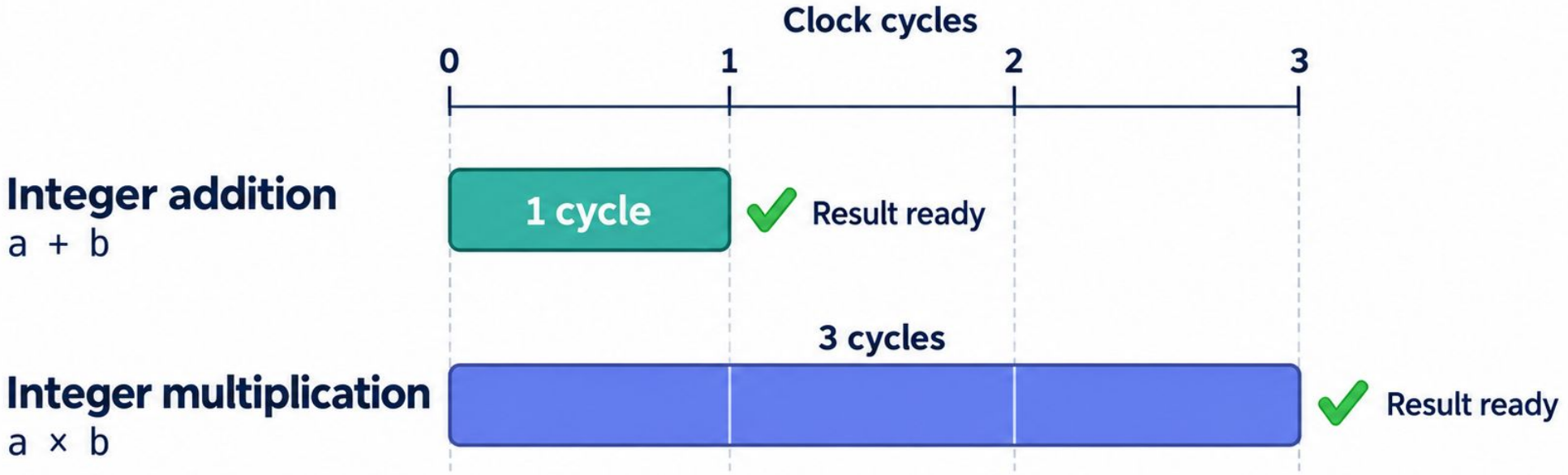


Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

CPU Pipeline

Some instructions take longer

Integer addition vs. multiplication

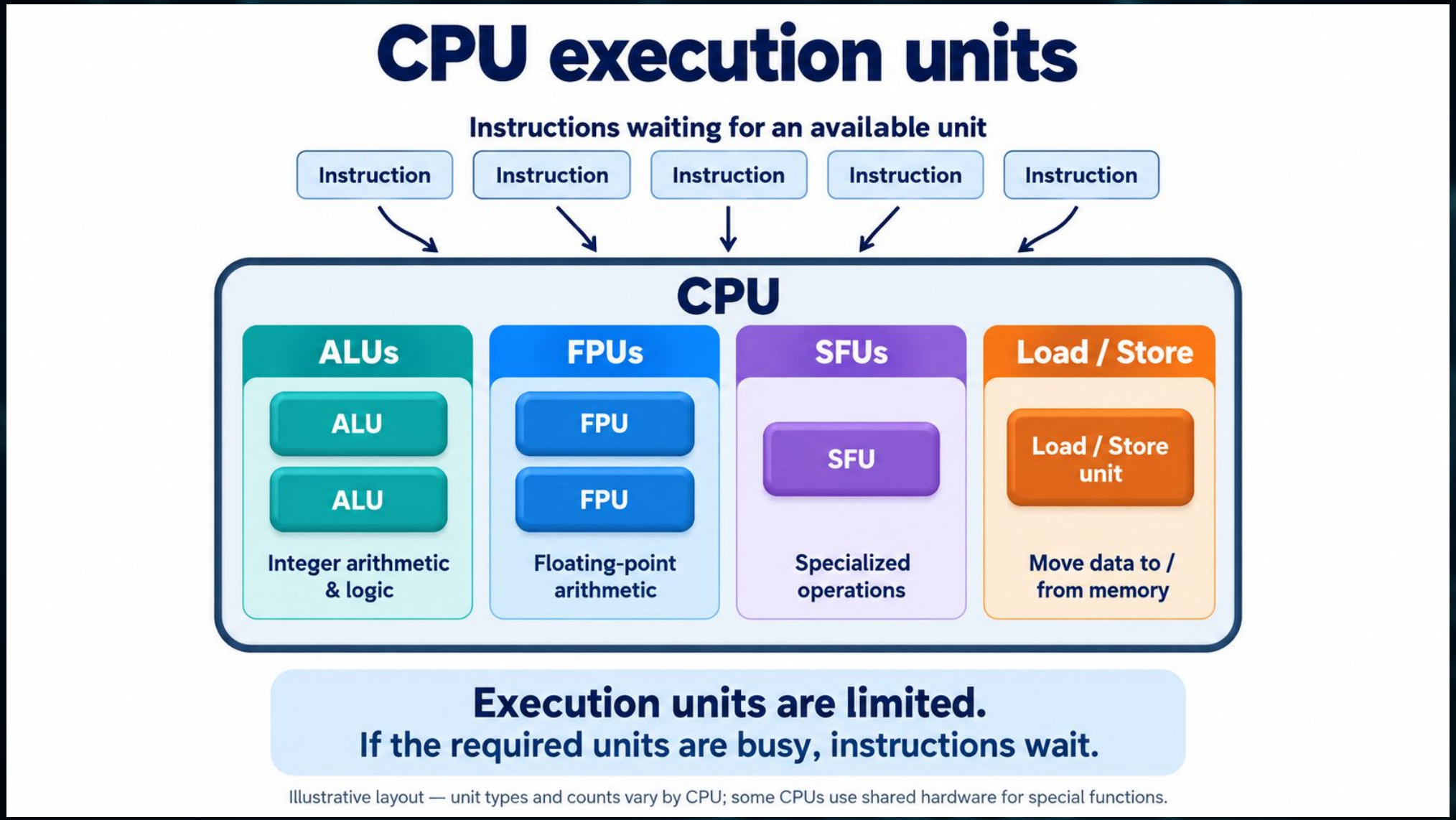


The next dependent instruction must wait for the result.

Illustrative latency: addition 1 cycle, multiplication 3 cycles. Actual timings vary by CPU and instruction.

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

CPU Execution Units



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Operation Simplification

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Strength Reduction

```
22    uint64_t devide_by_2(uint64_t n){
23        return n/2;
24    }
```



```
97    uint64_t shift_right_by_1(uint64_t n){
98        return n >> 1;
99    }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Strength Reduction

```
1    "devide_by_2(unsigned long)":
2            mov     rax, rdi
3            shr     rax
4            ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Strength Reduction

```
22  uint64_t devide_by_3(uint64_t n){
23      return n / 3;
24  }
```

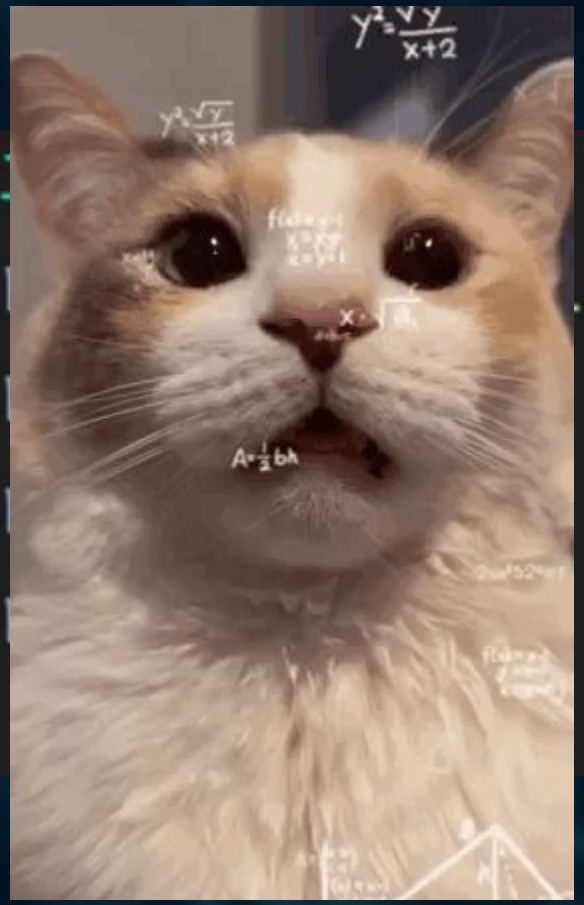


```
26  uint64_t devide_by_3_better(uint64_t n){
27      constexpr double y = 1./3.;
28      return n * y;
29  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Strength Reduction

```
"devide_by_3(uns                    :
        movabs                      .46912365172O5
        mul
        mov
        shr
        ret
```



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Division by 3, Without Division



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Division by 3, Without Division

The magic number is a scaled version of ⅓.



$$M = \lceil 2^{65} / 3 \rceil = \texttt{0xAAAAAAAAAAAAAAAB}$$

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Bit Operation Simplifications

```
110    uint32_t foo(uint32_t x) {
111        return ((x & 0x000000FFu) << 24) |
112               ((x & 0x0000FF00u) <<  8) |
113               ((x & 0x00FF0000u) >>  8) |
114               ((x & 0xFF000000u) >> 24);
115    }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Bit Operation Simplifications

```
45    reverse_bytes(unsigned int):
46            rev     w0, w0
47            ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Dead Code Elimination

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Dead Code Elimination

```
31  int dead_code(int x) {
32      // Result is never used
33      int unused = x * 42;
34
35      // Unreachable code
36      if (false) {
37          return x + 100;
38      }
39
40      return x + 1;
41  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Dead Code Elimination

```
33    "dead_code(int)":
34            lea     eax, [rdi+1]
35            ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Inlining

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Inlining

```
int add1(int a) {
    return a + 1;
}

int main(int c, char** argv){
    return add1(c);
}
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Inlining

```
main:                                   # @main
        lea     eax, [rdi + 1]
        ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Inlining

```
int add(int a, int n){
    if (n==0){
        return a;
    }
    return add(a+1, n-1);
}

int main(int c, char** argv){
    return add(c, c+10);
}
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Inlining

```
1    add(int, int):
2            lea     eax, [rdi + rsi]
3            ret
4    main:
5            lea     eax, [2*rdi + 10]
6            ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Inlining

Inlining has a budget

The compiler weighs function size against expected benefit.

Default limits



Larger functions may stay as calls.

Higher limits



More opportunities to inline.

GCC	`g++ -O3 -finline-limit=1000 demo.cpp`
Clang	`clang++ -O3 -mllvm -inline-threshold=1000 demo.cpp`

Higher thresholds encourage inlining; they do not force it.

Illustrative values; compiler-specific cost models.
Clang uses an internal LLVM tuning option.

Potentially fewer calls • **Potentially larger code**

More aggressive does not always mean faster.

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Compile Time Calculations

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Compile Time Calculations

```
43  template <std::size_t N>
44  constexpr int sum_array(const std::array<int, N>& values, int multiplier) {
45      int sum = 0;
46      for (int value : values) {
47          sum += value;
48      }
49      return sum * multiplier;
50  }
51
52  int main(){
53      constexpr int result = sum_array(std::array<int, 4>{1, 2, 3, 4}, 5);
54      return result;
55  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Compile Time Calculations

```
mov     eax, 50
ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reordering

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reordering

```
63  uint64_t reorder(uint64_t a, uint64_t b,
64                   uint64_t c, uint64_t d)
65  {
66      uint64_t first  = (a * b) + 7;
67      uint64_t second = (c * d) + 9;
68
69      return first ^ second;
70  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reordering

```
first  = a * b;
first += 7;
second = c * d;
second += 9;
```

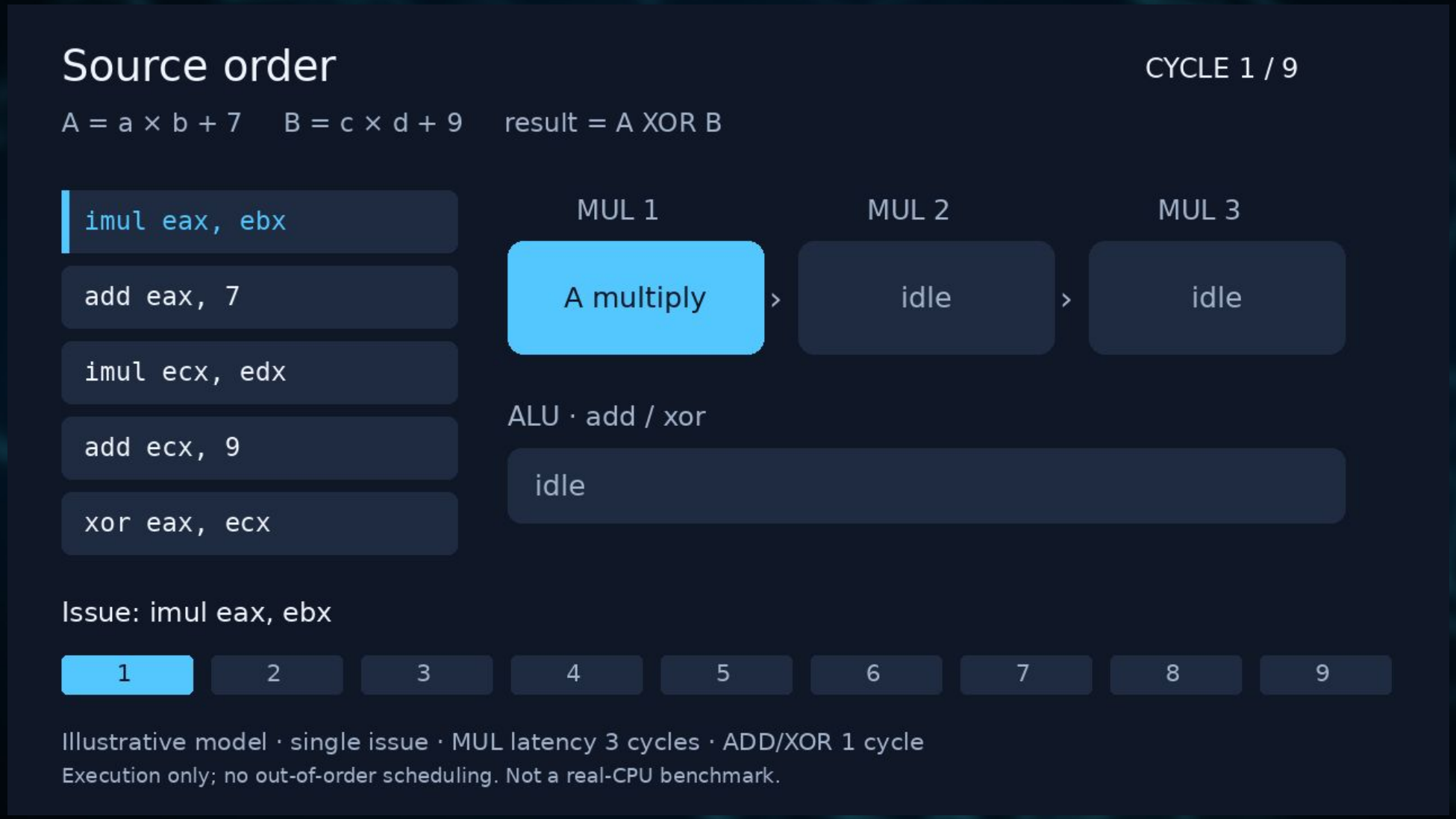
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reordering

```
1   "reorder(unsigned long, unsigned long, unsigned long,
2           imul    rdi, rsi
3           imul    rdx, rcx
4           lea     rax, [rdi+7]
5           add     rdx, 9
6           xor     rax, rdx
7           ret
```

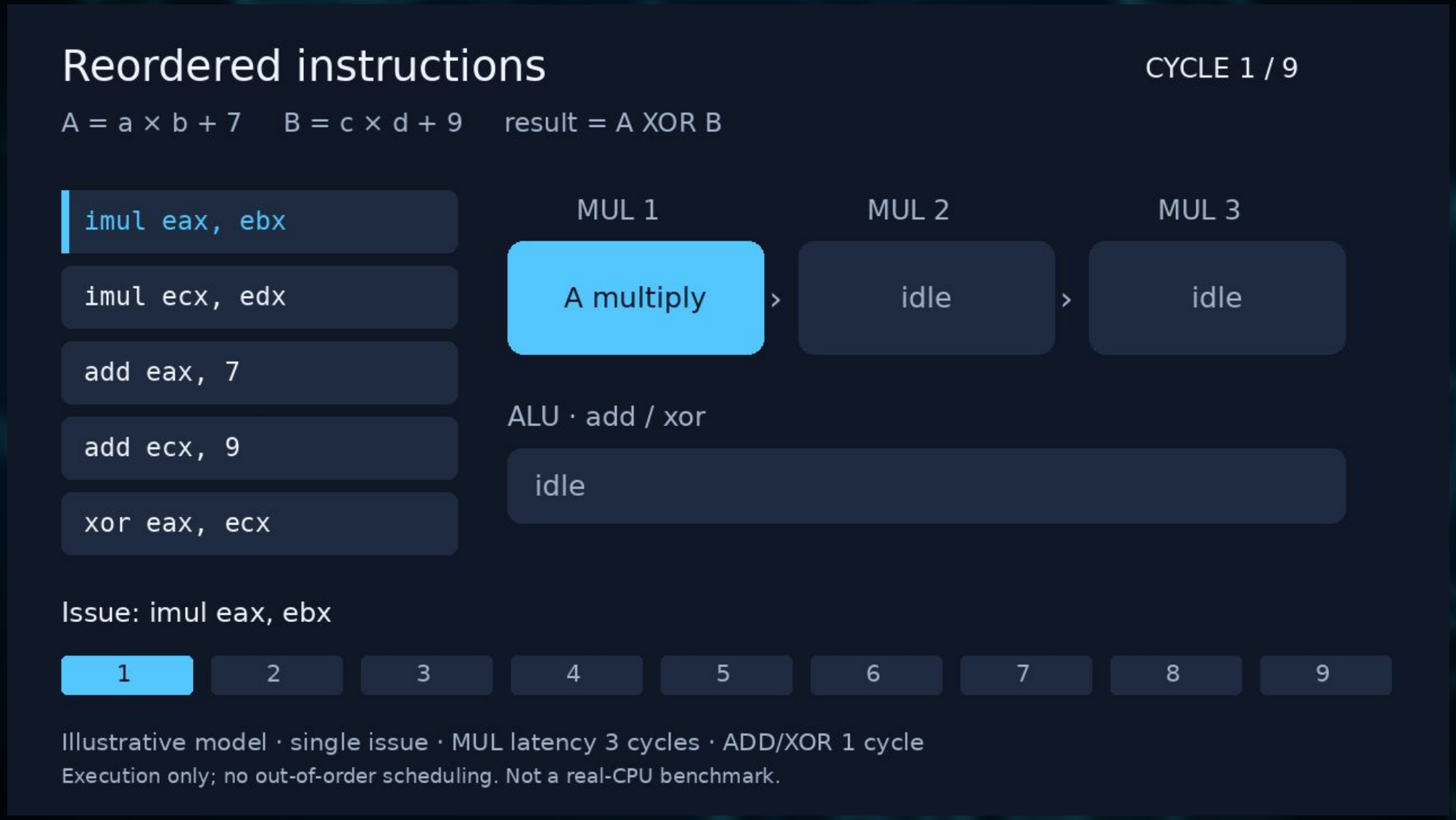
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reordering



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reordering



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Behind the Curtains of “if”

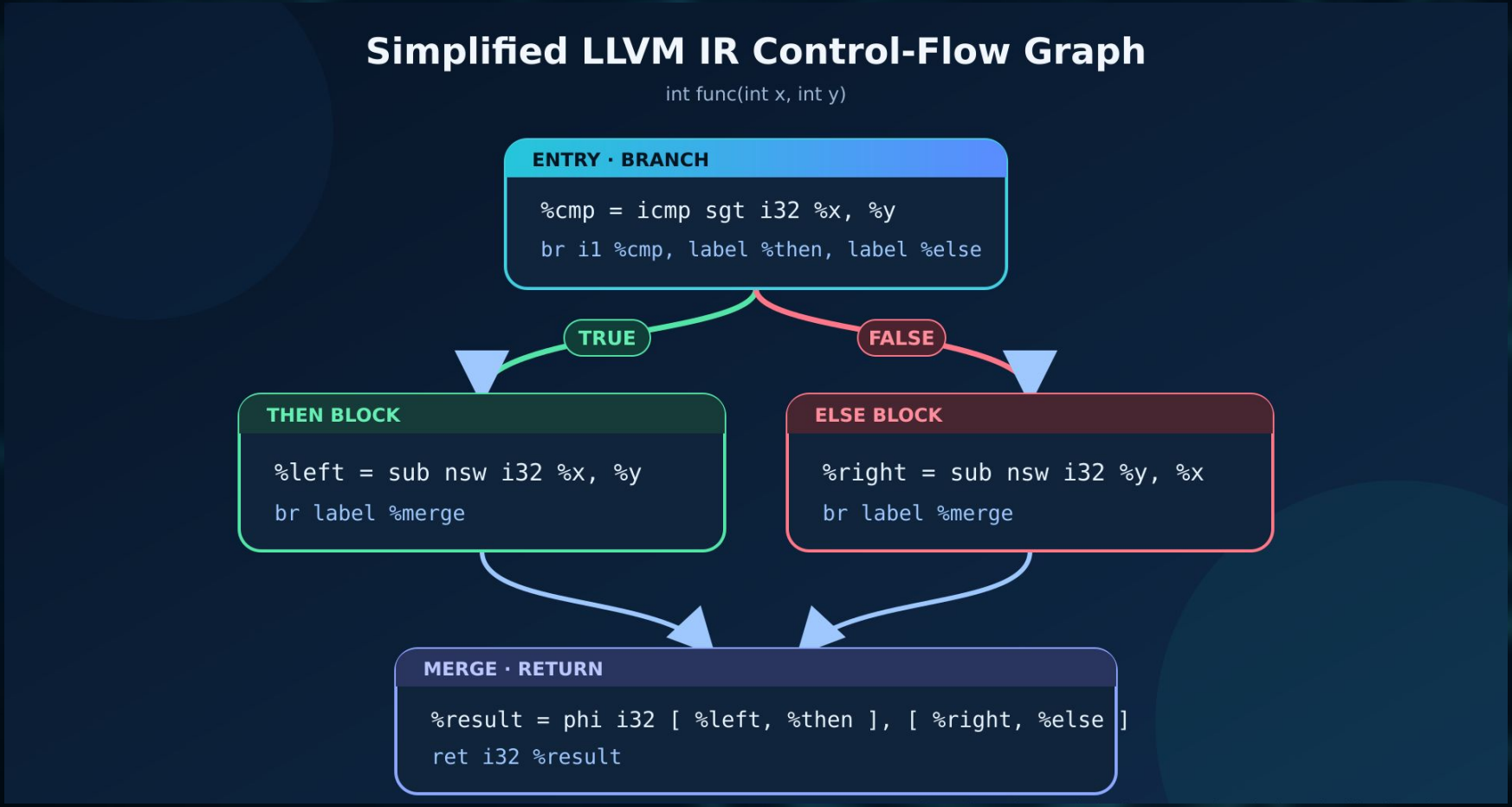
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

if Statement

```
3   int func(int x, int y){
4       if (x > y) {
5           return x-y;
6       } else {
7           return y-x;
8       }
9   }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Behind the Curtains of “if”



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Behind the Curtains of “if”

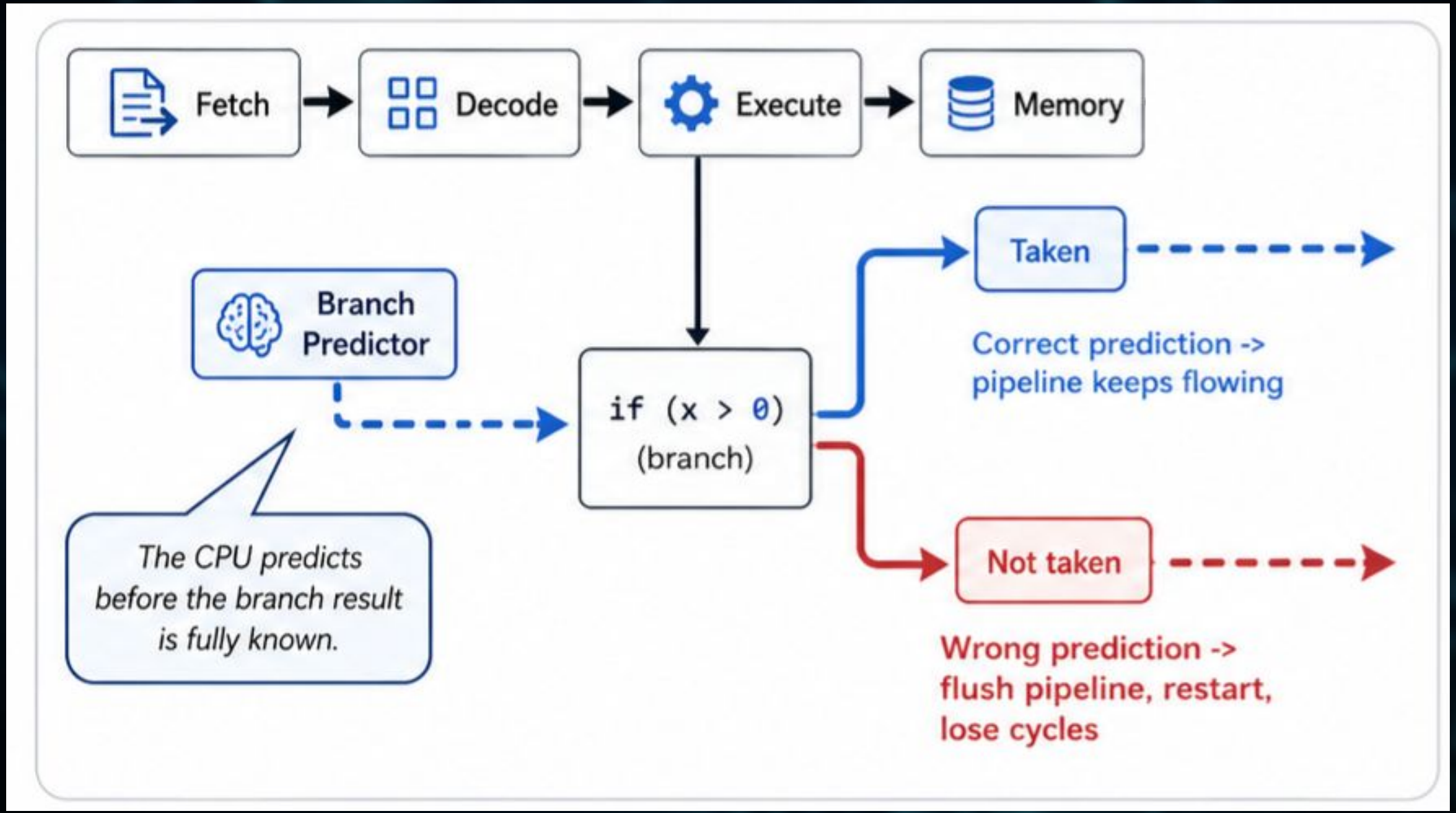
```
13  func(int, int):
14          push    rbp
15          mov     rbp, rsp
16          mov     dword ptr [rbp - 8], edi
17          mov     dword ptr [rbp - 12], esi
18          mov     eax, dword ptr [rbp - 8]
19          cmp     eax, dword ptr [rbp - 12]
20          jle     .LBB1_2
21          mov     eax, dword ptr [rbp - 8]
22          sub     eax, dword ptr [rbp - 12]
23          mov     dword ptr [rbp - 4], eax
24          jmp     .LBB1_3
25  .LBB1_2:
26          mov     eax, dword ptr [rbp - 12]
27          sub     eax, dword ptr [rbp - 8]
28          mov     dword ptr [rbp - 4], eax
29  .LBB1_3:
30          mov     eax, dword ptr [rbp - 4]
31          pop     rbp
32          ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branch Predictor

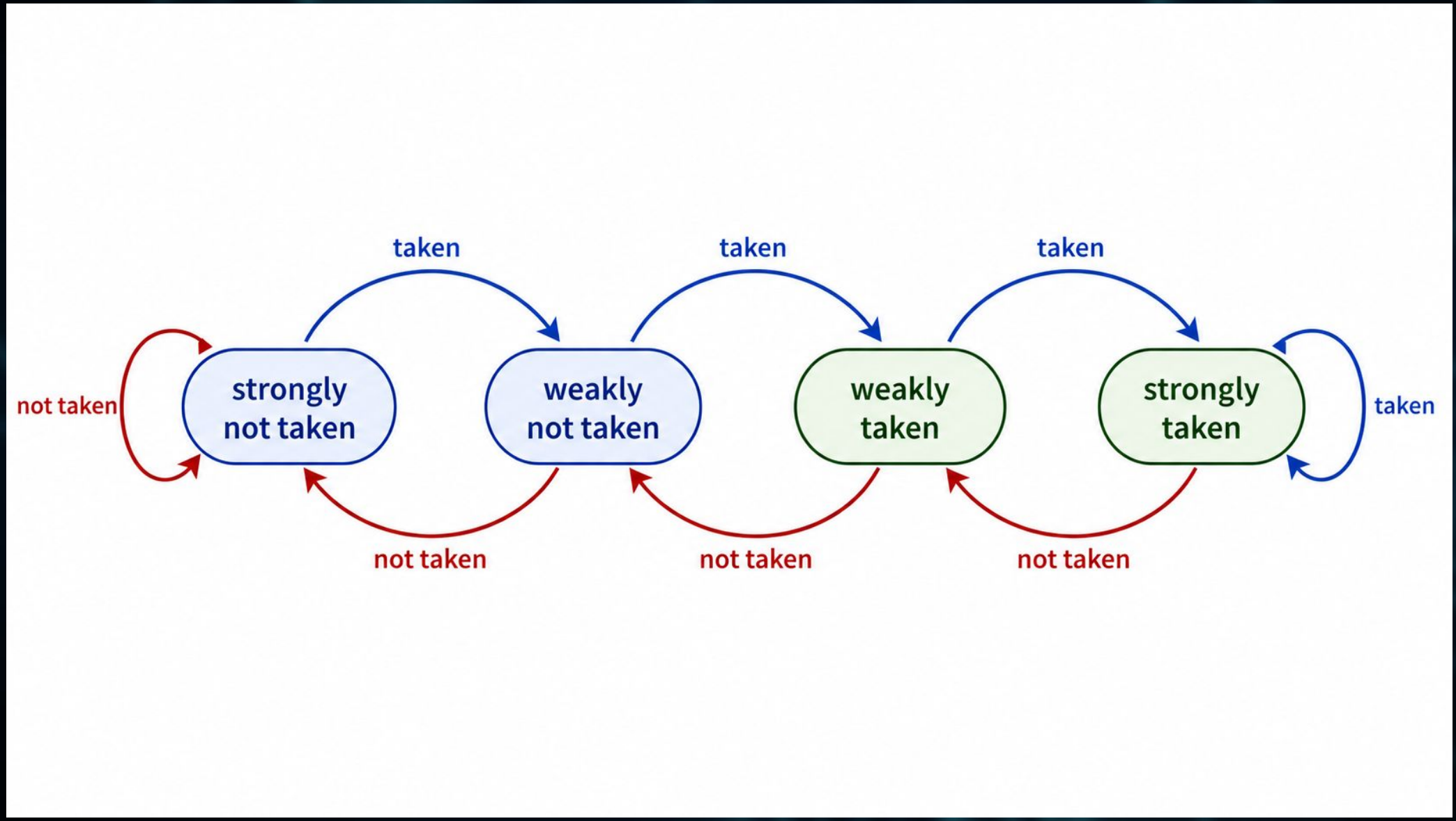
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branch Prediction



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branch Prediction: Simple Example

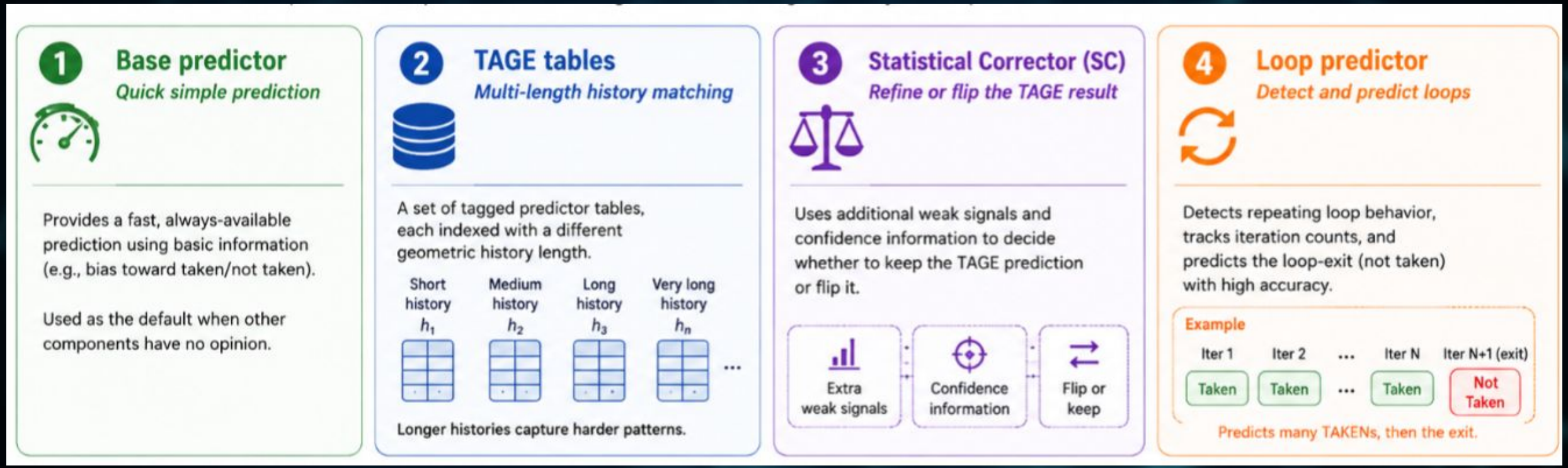


Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Modern Branch Prediction

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Modern Branch Prediction



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branchless Programming

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branchless Programming

```
52    int sub2(int x, int y){
53        return x > y ? x-y : y-x;
54    }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branchless Programming: Compiler is Our Friend

```
 8   sub2(int, int):
 9           mov     eax, edi
10           sub     eax, esi
11           sub     esi, edi
12           cmovge  eax, esi
13           ret
14
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Branchless Programming: Compiler is Our Friend

```
1   sub(int, int):
2           mov     eax, edi
3           sub     eax, esi
4           sub     esi, edi
5           cmovge  eax, esi
6           ret
7
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Simplify Branches

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Simplify Branches

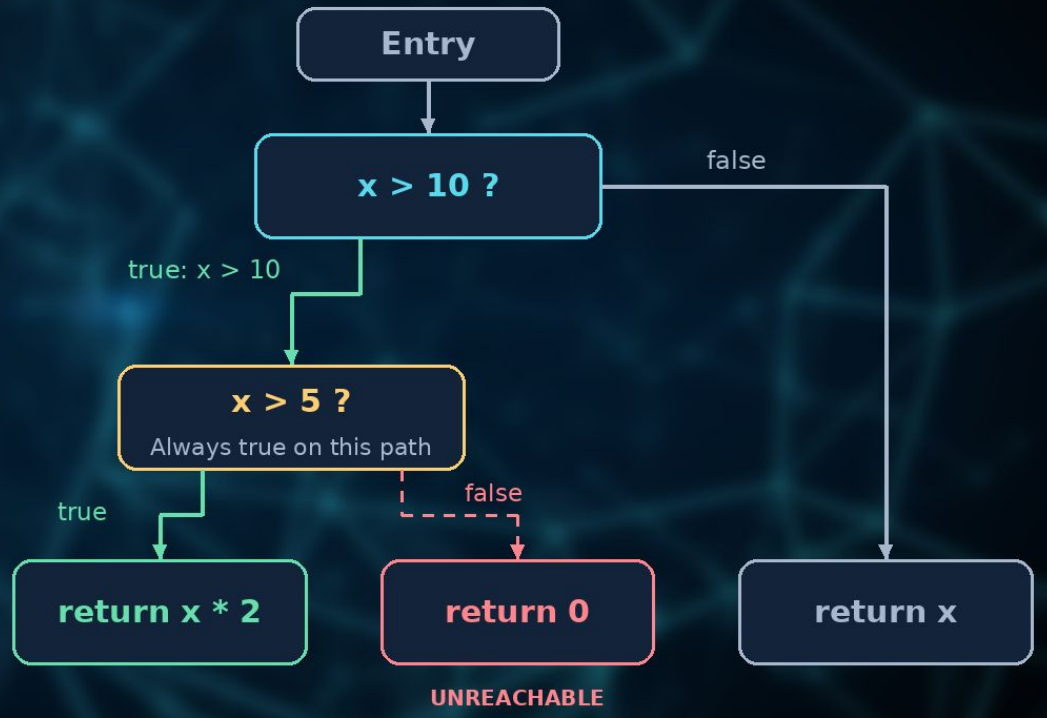
LLVM / CONTROL-FLOW SIMPLIFICATION

Two tests. One redundant decision.

The outer condition already tells us which way the inner branch must go.

C++ SOURCE

```
int calculate(int x) {
    if (x > 10) {
        if (x > 5)
            return x * 2;
        return 0;
    }
    return x;
}
```



Conceptual CFG • Each rounded box is a basic block

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Simplify Branches

LLVM / CONTROL-FLOW SIMPLIFICATION

One test. Same result.

Remove the redundant test and unreachable block; connect directly to the live return.

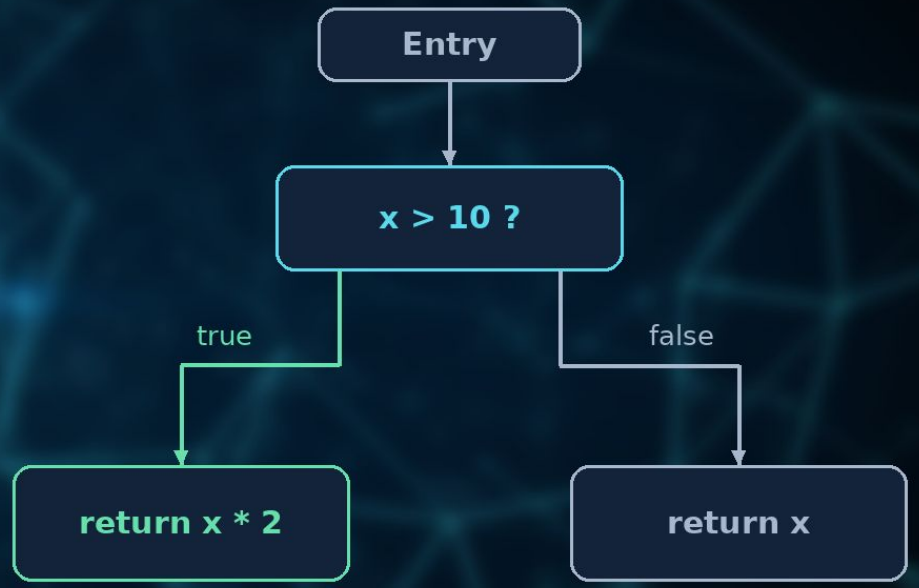
C++ SOURCE

```
int calculate(int x) {
    if (x > 10)
        return x * 2;
    return x;
}
```

PROVEN ON THE TRUE PATH

`x > 10  implies  x > 5`

No second decision is needed.



Redundant comparison + dead return removed

Conceptual CFG • Each rounded box is a basic block

02 / 02

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

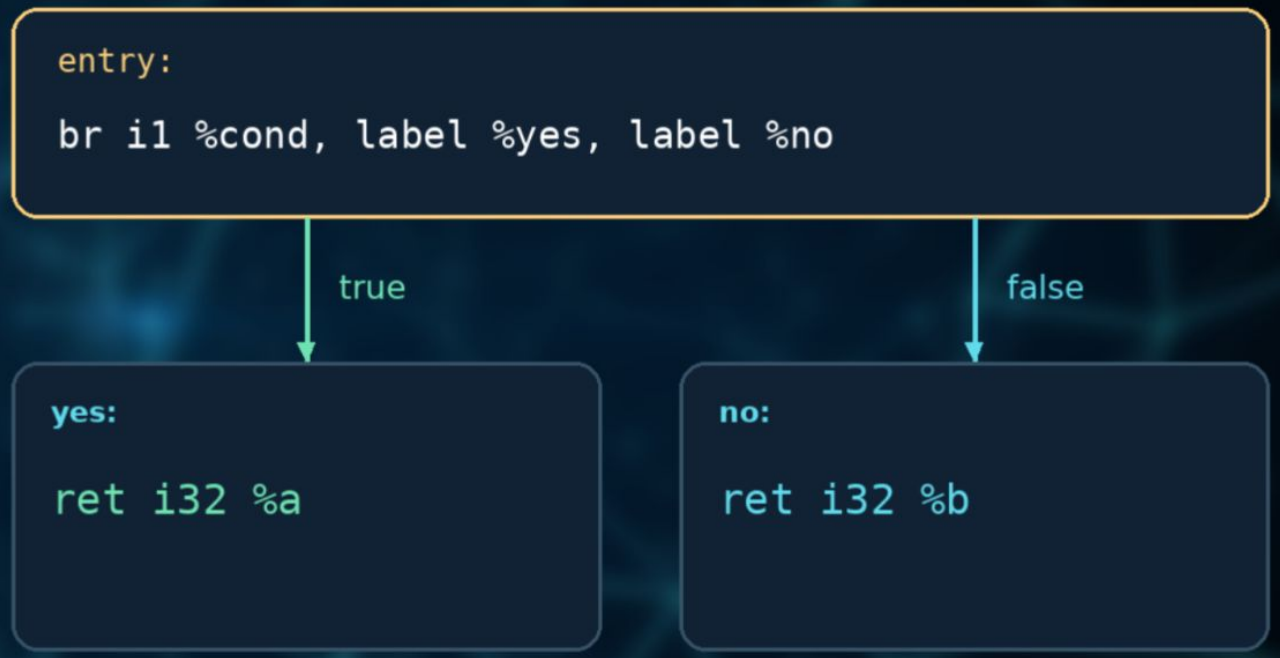
Keep Decision But Remove The Branch

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Replace Branches

C++ SOURCE

```
int choose(bool cond,
           int a, int b) {
    if (cond)
        return a;
    return b;
}
```



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Replace Branches

EQUIVALENT C++ EXPRESSION

```
int choose(bool cond,
           int a, int b) {
    return cond ? a : b;
}
```

OPTIMIZED LLVM IR

```
entry:
  %r = select i1 %cond, i32 %a, i32 %b
  ret i32 %r
```

One basic block. No conditional br.

true → choose %a false → choose %b

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Replace Branches: X86 backend

```
19    choose(bool, int, int):
20            tst     w0, #0x1
21            csel    w0, w1, w2, ne
22            ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Bypassing Jumps

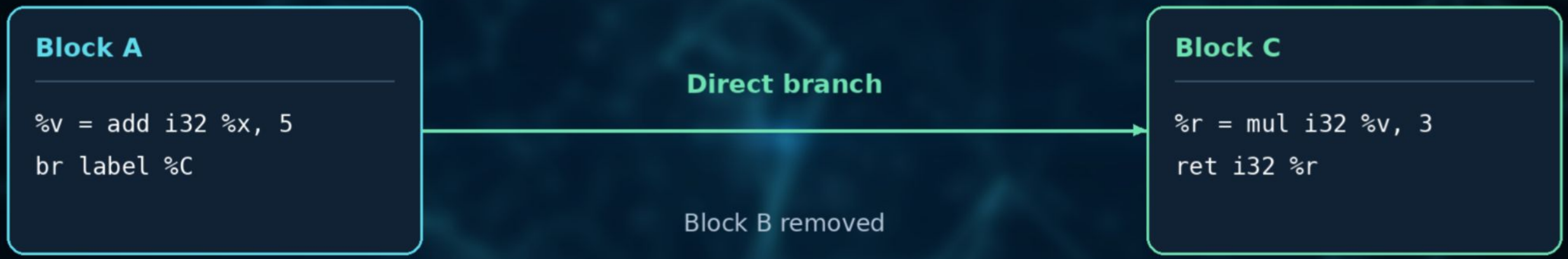
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Bypassing Branches



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Bypassing Branches

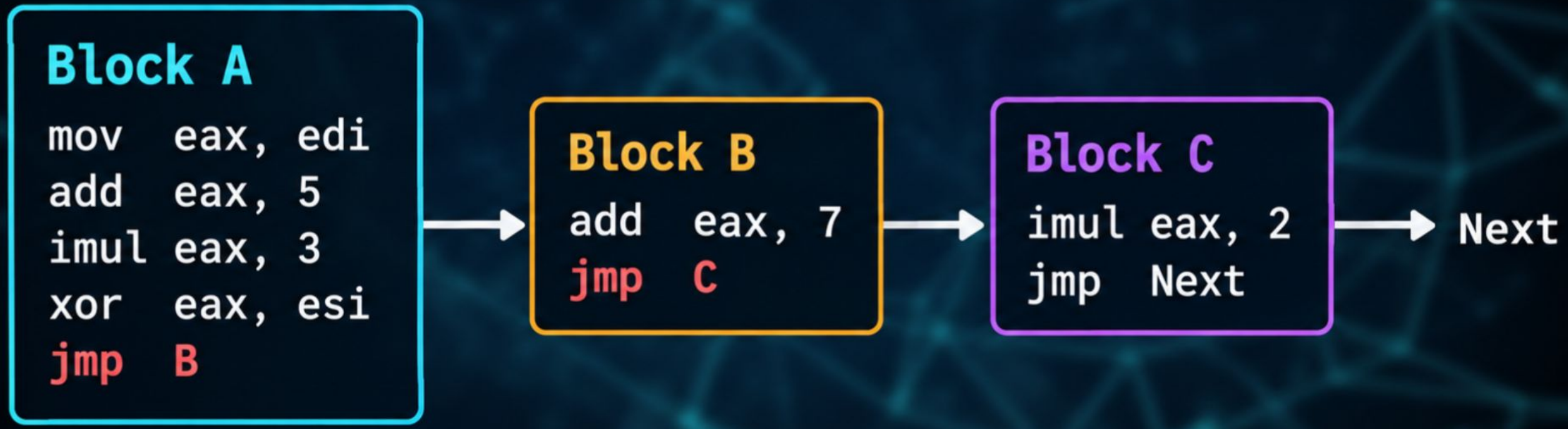


Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Basic-Block Merging

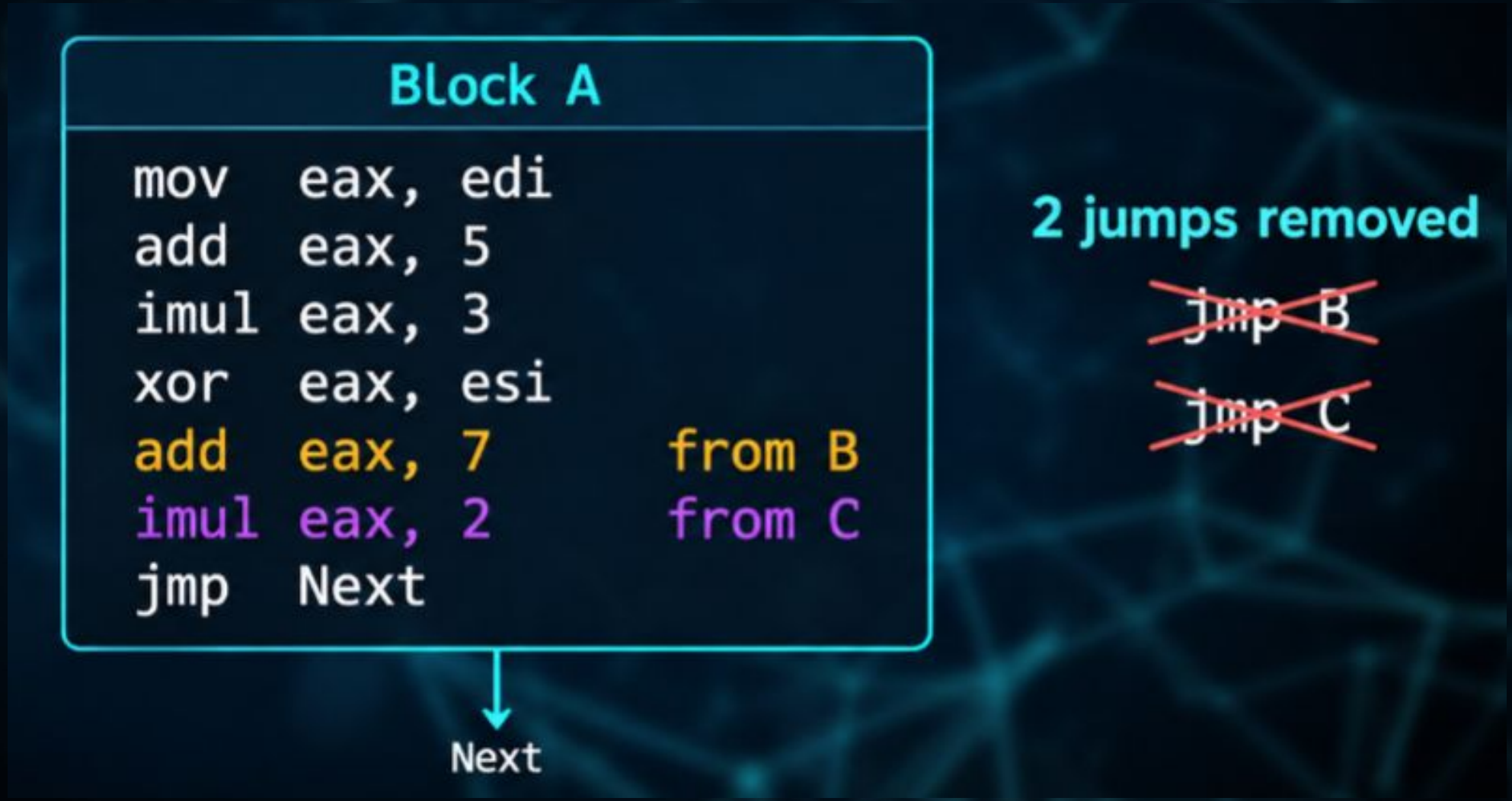
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Basic-Block Merging



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Basic-Block Merging



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

```
90
91  void set_z(std::vector<uint64_t>& vec){
92      for (auto& v : vec){
93          v = 0;
94      }
95  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

```
5 tail call void @llvm.memset.p0.i64(ptr align 8 %2, i8 0, i64 %12, i1 false)
```

```
20    set_z(std::vector<unsigned long, std::alloca
21              mov     x8, x0
22              ldr     x0, [x0]
23              ldr     x8, [x8, #8]
24              cmp     x0, x8
25              b.eq    .LBB3_2
26              sub     x8, x8, x0
27              mov     w1, wzr
28              sub     x8, x8, #8
29              and     x8, x8, #0xfffffffffffffff8
30              add     x2, x8, #8
31              b       memset
32    .LBB3_2:
33              ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

```
101  unsigned bit_width(uint32_t x) {
102      unsigned bits = 0;
103      while (x != 0) {
104          x >>= 1;
105          ++bits;
106      }
107      return bits;
108  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

```
%2 = tail call range(i32 0, 33) i32 @llvm.ctlz.i32(i32 %0, i1 false)
```

```
39   bit_width(unsigned int):
40           mov     w8, #32
41           clz     w9, w0
42           sub     w0, w8, w9
43           ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

```
117  size_t length(const char* s) {
118      const char* p = s;
119
120      while (*p != '\0')
121          ++p;
122
123      return p - s;
124  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Idiom

```
60      %2 = tail call i64 @strlen(ptr noundef nonnull dereferenceable(1) %0)
```

```
49    length(char const*):
50                  b       strlen
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Unrolling

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Unrolling

```
53  uint64_t sum8(const std::vector<uint64_t>& values) {
54      uint64_t sum = 0;
55
56      for (uint64_t i = 0; i < 8; ++i) {
57          sum += values[i];
58      }
59
60      return sum;
61  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Unrolling

```
"sum8(std::vector<unsigned long, std::allocator<unsigned
        mov     rdx, QWORD PTR [rdi]
        mov     rax, QWORD PTR [rdx+16]
        add     rax, QWORD PTR [rdx+8]
        add     rax, QWORD PTR [rdx+24]
        add     rax, QWORD PTR [rdx]
        add     rax, QWORD PTR [rdx+32]
        add     rax, QWORD PTR [rdx+40]
        add     rax, QWORD PTR [rdx+48]
        add     rax, QWORD PTR [rdx+56]
        ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Loop Unrolling

```
19    .LBB2_1:
20            add     rax, qword ptr [rcx + 8*rdx]
21            add     rax, qword ptr [rcx + 8*rdx + 8]
22            add     rax, qword ptr [rcx + 8*rdx + 16]
23            add     rax, qword ptr [rcx + 8*rdx + 24]
24            add     rax, qword ptr [rcx + 8*rdx + 32]
25            add     rax, qword ptr [rcx + 8*rdx + 40]
26            add     rax, qword ptr [rcx + 8*rdx + 48]
27            add     rax, qword ptr [rcx + 8*rdx + 56]
28            add     rdx, 8
29            cmp     rdx, 16
30            jne     .LBB2_1
31            ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SIMD

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SIMD

```
"sum8(std::vector<unsigned long, std::allocator<unsigned l
        mov     rax, QWORD PTR [rdi]
        movdqu  xmm0, XMMWORD PTR [rax+48]
        movdqu  xmm3, XMMWORD PTR [rax+32]
        movdqu  xmm1, XMMWORD PTR [rax]
        movdqu  xmm2, XMMWORD PTR [rax+16]
        paddq   xmm0, xmm3
        paddq   xmm1, xmm2
        paddq   xmm0, xmm1
        movdqa  xmm1, xmm0
        psrldq  xmm1, 8
        paddq   xmm0, xmm1
        movq    rax, xmm0
        ret
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SIMD

```
1   "sum8(std::vector<unsigned long, std::allocator<unsigned
2           mov      rax, QWORD PTR [rdi]
3           vmovdqu ymm1, YMMWORD PTR [rax+32]
4           vpaddq  ymm1, ymm1, YMMWORD PTR [rax]
5           vextracti128     xmm0, ymm1, 0x1
6           vpaddq  xmm0, xmm0, xmm1
7           vpsrldq xmm1, xmm0, 8
8           vpaddq  xmm0, xmm0, xmm1
9           vmovq   rax, xmm0
10          vzeroupper
11          ret
```

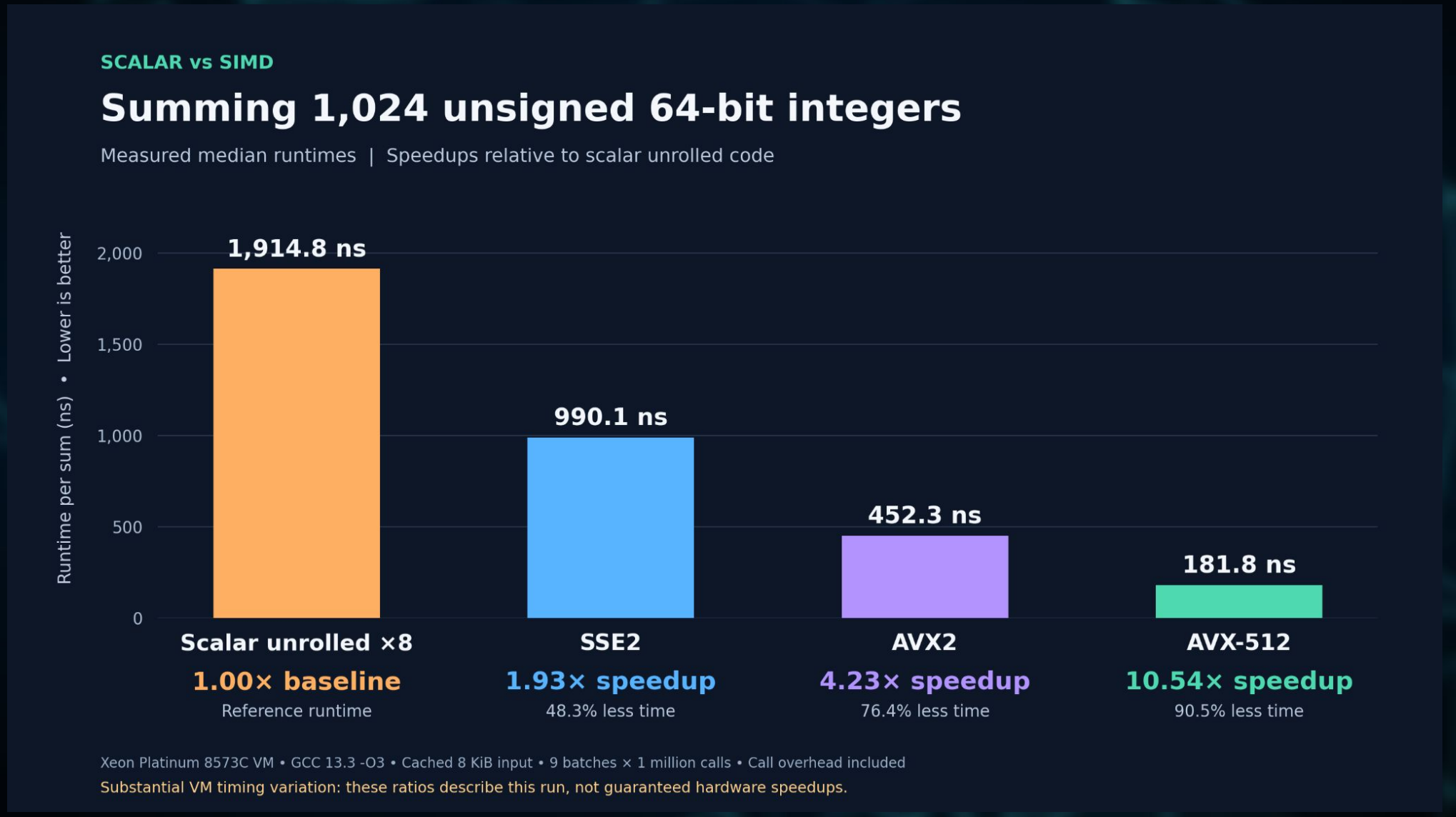
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SIMD: Hints Of Usage

Instruction set	Register	64-bit integers per register	Loads for 8 elements
SSE2	XMM, 128 bits	2	4
AVX2	YMM, 256 bits	4	2
AVX-512	ZMM, 512 bits	8	1

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SIMD: Performance



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SIMD: Complex Options

```
 8      simd::vec<float, 4> v1([](int i) -> float {
 9          return i + 1.5; });
10
11      simd::vec<float, 4> v2([](int i) -> float {
12          return (i + 1.5) * 2.1; });
13
14      auto v3 = v1 + v2;
15
16      float total = simd::reduce(v3);
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SROA

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SROA

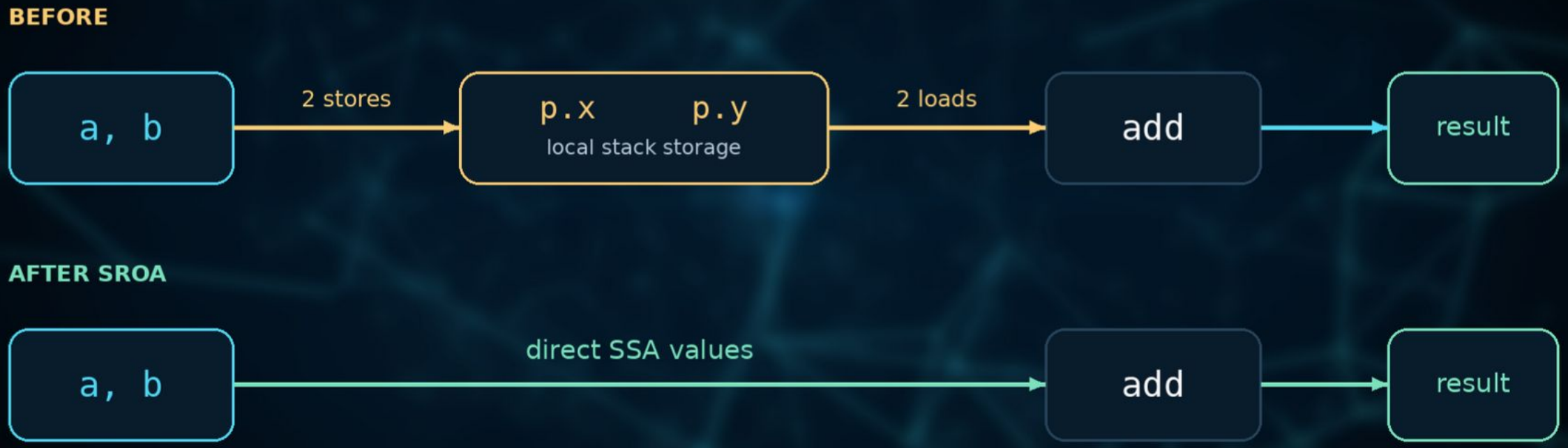
SROA removes local aggregate storage when its accesses can be analyzed.



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SROA

The values flow directly to the addition instead of through a local struct.



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SROA

BEFORE · simplified LLVM IR

```
%p = alloca { i32, i32 }
%px = getelementptr { i32, i32 }, ptr %p, i32 0, i32 0
%py = getelementptr { i32, i32 }, ptr %p, i32 0, i32 1
store i32 %a, ptr %px
store i32 %b, ptr %py
%x = load i32, ptr %px
%y = load i32, ptr %py
%sum = add i32 %x, %y
ret i32 %sum
```



AFTER SROA

```
%sum = add i32 %a, %b
ret i32 %sum
```

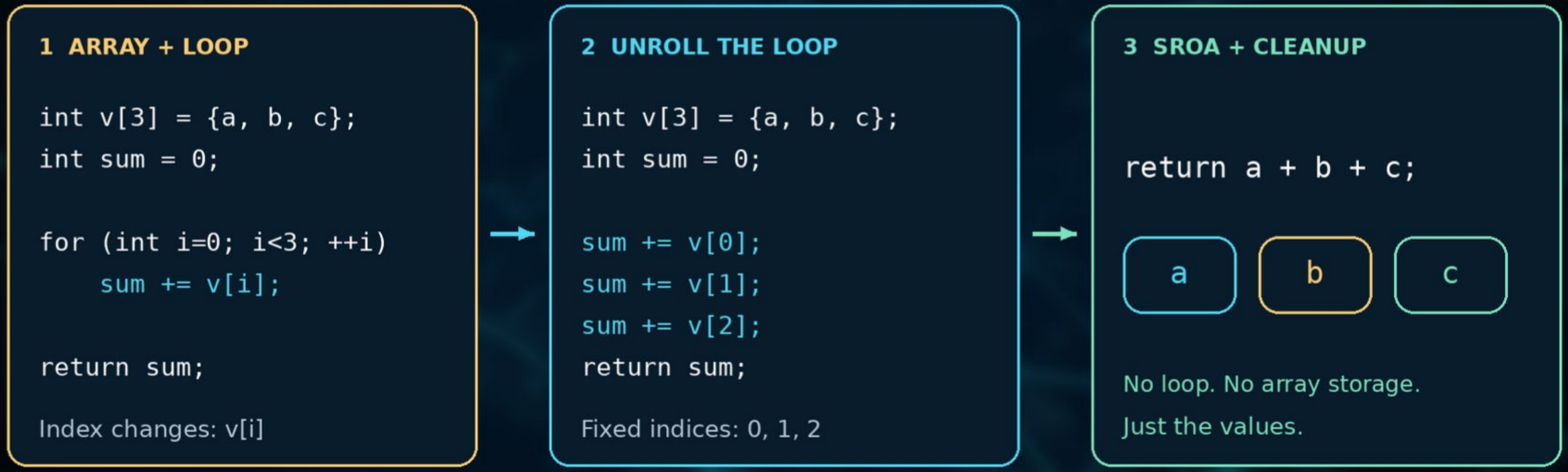
No allocation for p

No field addresses

No loads or stores

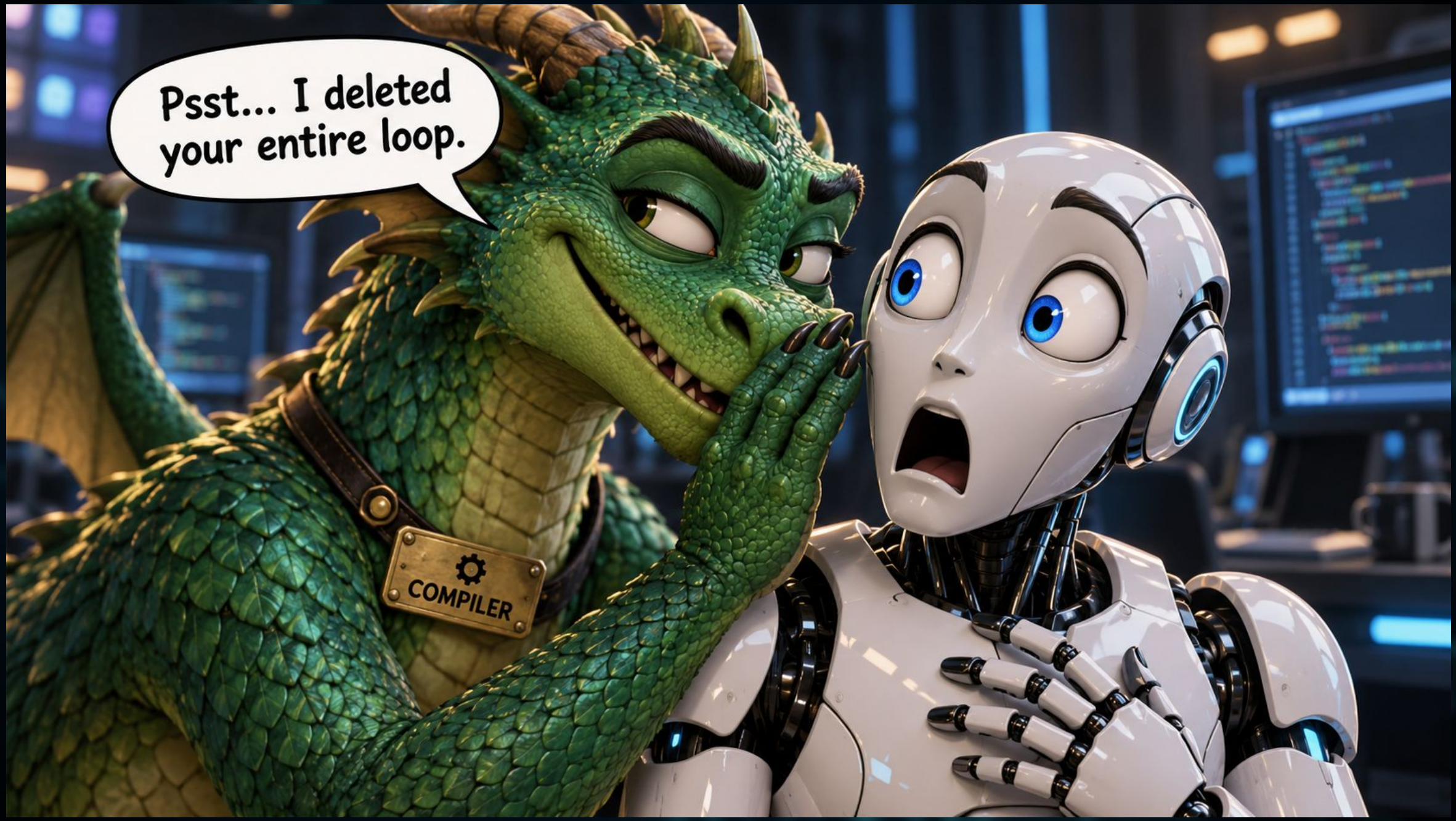
Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

SROA



Unrolling removes the loop. SROA removes the array.

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

C++26 Reflections

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Reflection Syntax



REFLECT

Turn a type or entity into metadata

```
constexpr auto r = ^^int;
```

r holds a `std::meta::info` value



SPLICE

Use the reflected type or entity in code

```
typename [:r:] value = 42;
```

Here, this declares an `int`

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Enumerators Printer

```
4   enum class Units{Velocity, Acceleration, Distance};
5
6   template <typename Value>
7   static void print(Value v) {
8       constexpr auto meta_vals = std::define_static_array(
9           std::meta::enumerators_of(^^Value));
10      template for (constexpr auto e: meta_vals) {
11          if (v == [:e:]) {
12              std::println("{}", std::meta::identifier_of(e));
13              return;
14          }
15      }
16  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Parsing Input Arguments

```
11  struct AppConfig {
12      std::string input; std::string output;
13      int threads = 1; bool verbose = false;
14  };
15
16  template <typename T>
17  T parse_args(std::unordered_map<std::string_view, std::string_view> args) {
18      T config{};
19      constexpr auto context = std::meta::access_context::current();
20      constexpr auto members =
21      std::define_static_array(std::meta::nonstatic_data_members_of(^^T, context));
22
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

Parsing Input Arguments

```
23      template for (constexpr auto member : members) {
24          constexpr auto name = std::meta::identifier_of(member);
25          if (args.contains(name)) {
26              using field_type = [: std::meta::type_of(member) :];
27              if constexpr (std::same_as<field_type, bool>) {
28                  config.[:member:] = true;
29              } else if constexpr (std::same_as<field_type, int>) {
30                  config.[:member:] = std::stoi(std::string(args[name]));
31              } else if constexpr (std::same_as<field_type, std::string>) {
32                  config.[:member:] = args[name];
33              }
34          }
35      }
36
37      return config;
38  }
```

Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

QUESTIONS



Alex Dathskovsky | alex.dathskovsky@scylladb.com | www.linkedin.com/in/alexdathskovsky

THANK YOU FOR LISTENING



