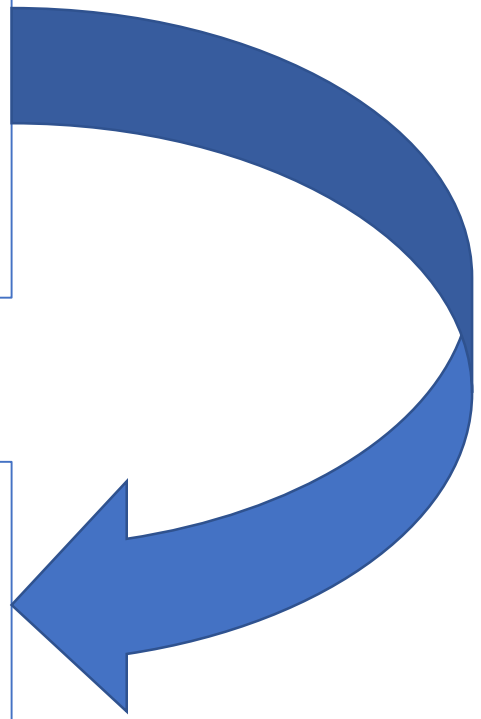
Compiler customer service

A story about vectorisation and compiler bug report

Arnaud Desitter

ACCU Oxford lightning talks – 23 May 2019

```
for (int i1 = 0; i1 < n1; ++i1)
{
  for (int i2 = 0; i2 < n2; ++i2)
  {
    res[i1*n2+i2] = a[i1*n2+i2] - b[i1*n2+i2];
  }
}
```



?

```
for (int index = 0; index < n1*n2; ++index)
{
  res[index] = a[index] - b[index];
}
```

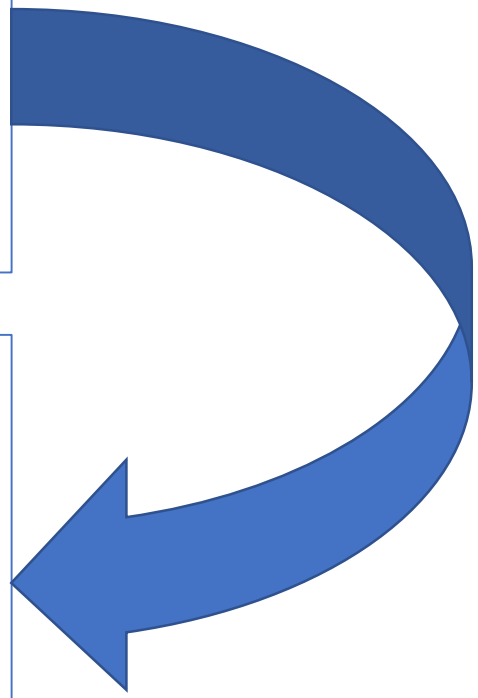
With n1=100,000 and n2=3

	penalty w.r.t. one flattened loop
gcc 9.1 -O3	+50%
gcc 9.1 -O2	+24%
clang 8.0 -O3	+34%
clang 8.0 -O2	+38%

Compilers only vectorise the inner loop.

Let’s try OpenMP

```
for (int i1 = 0; i1 < n1; ++i1)
{
  for (int i2 = 0; i2 < n2; ++i2)
  {
    res[i1*n2+i2] = a[i1*n2+i2] - b[i1*n2+i2];
  }
}
```



```
#pragma omp simd collapse(2)
for (int i1 = 0; i1 < n1; ++i1)
{
  for (int i2 = 0; i2 < n2; ++i2)
  {
    res[i1*n1+i2] = a[i1*n1+i2] - b[i1*n1+i2];
  }
}
```

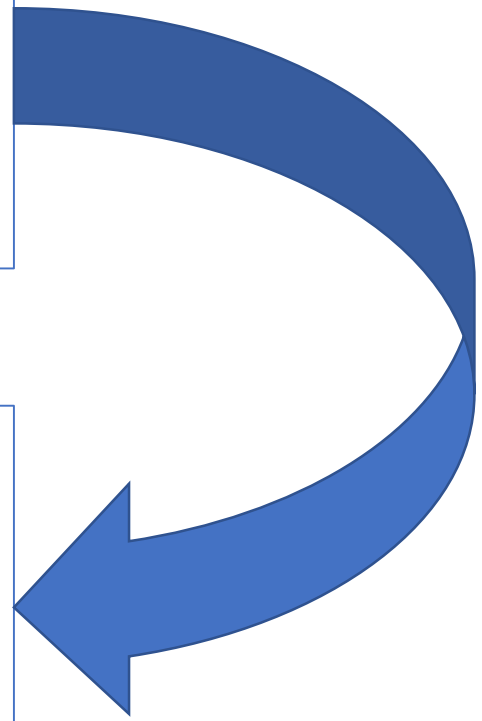
With n1=100,000 and n2=3

	penalty w.r.t. one flattened loop
gcc 9.1 -fopenmp-simd -O3	+100% (1)
clang 7.0 -fopenmp-simd -O3	+1230%
clang 8.0 -fopenmp-simd -O3	**+0%**

(1) https://gcc.gnu.org/bugzilla/show_bug.cgi?id=89371

Let’s try to collapse the loops manually

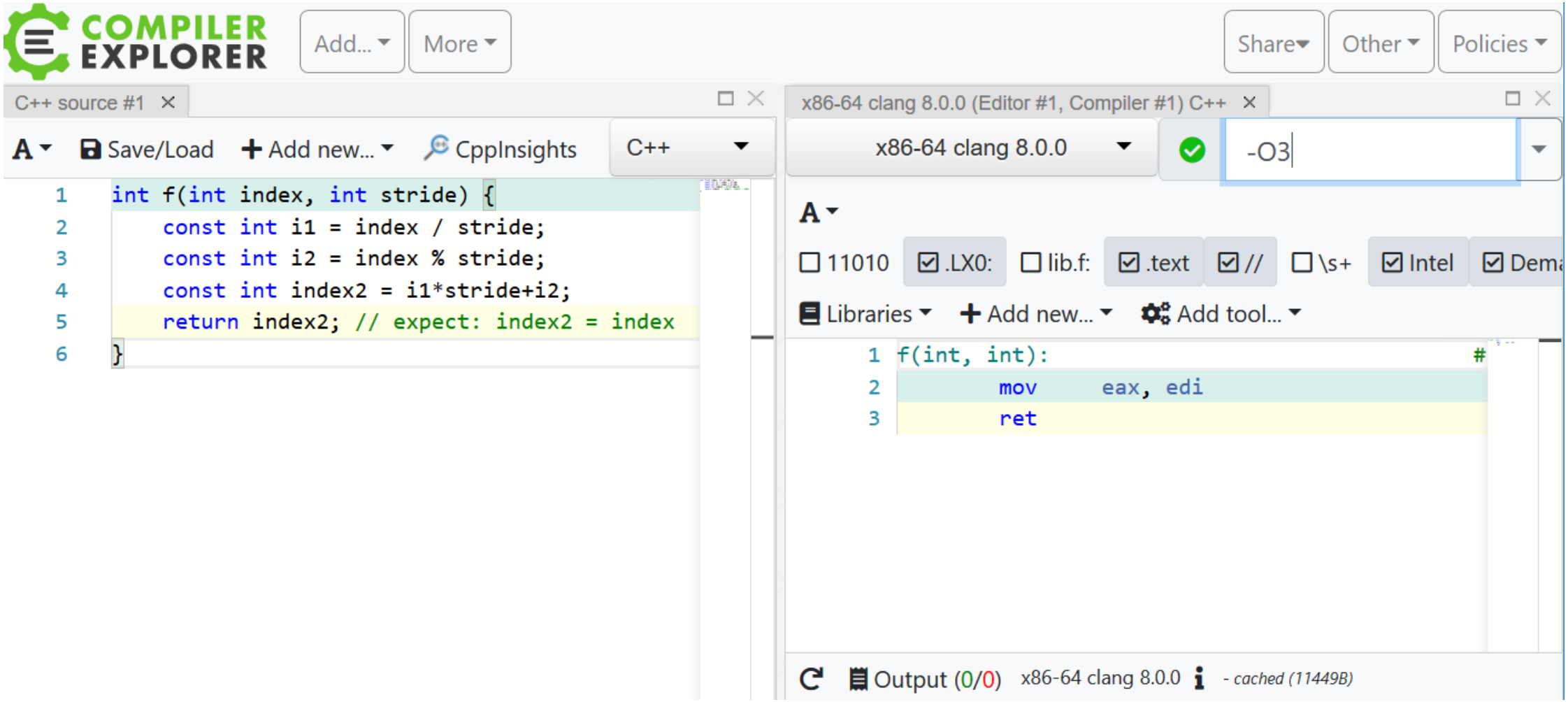
```
for (int i1 = 0; i1 < n1; ++i1)
{
  for (int i2 = 0; i2 < n2; ++i2)
  {
    res[i1*n2+i2] = a[i1*n2+i2] - b[i1*n2+i2];
  }
}
```

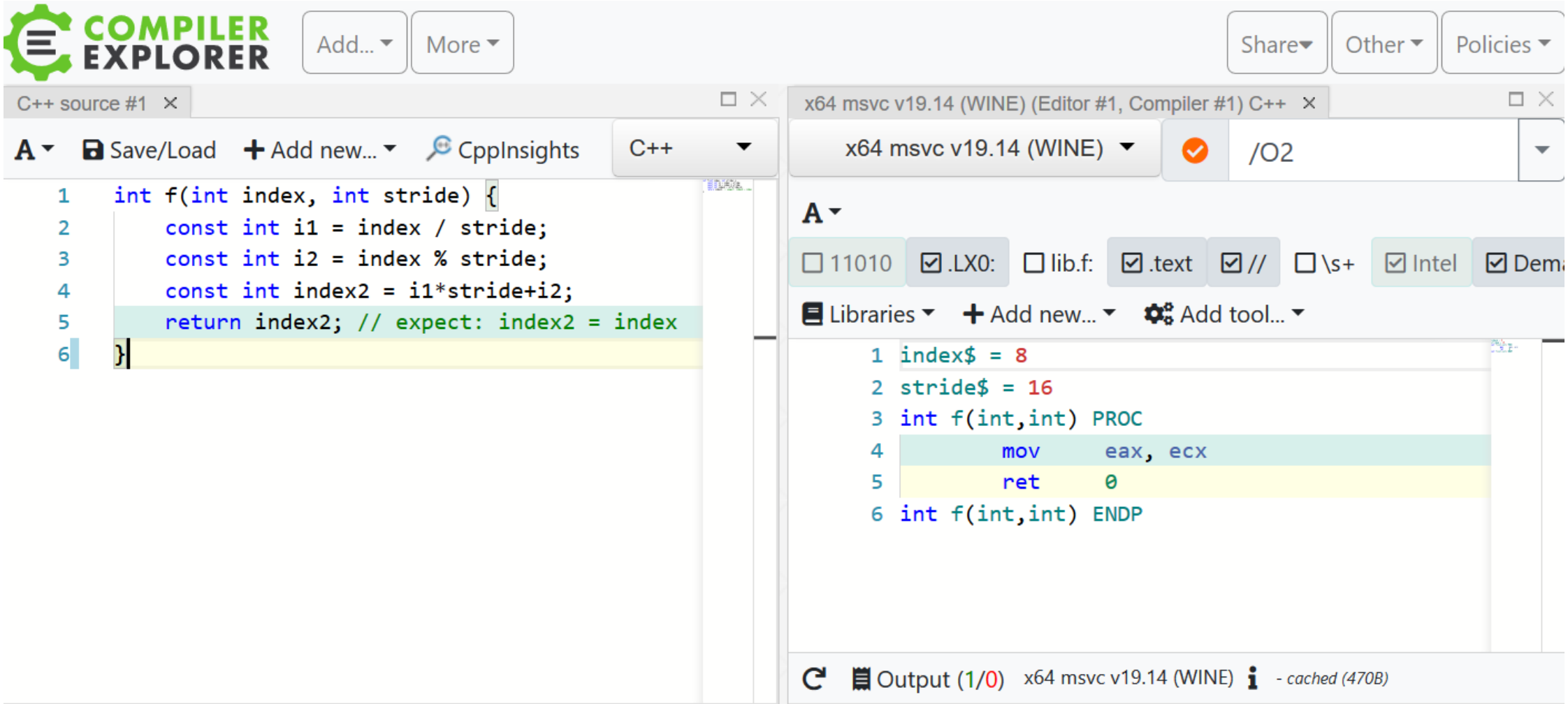


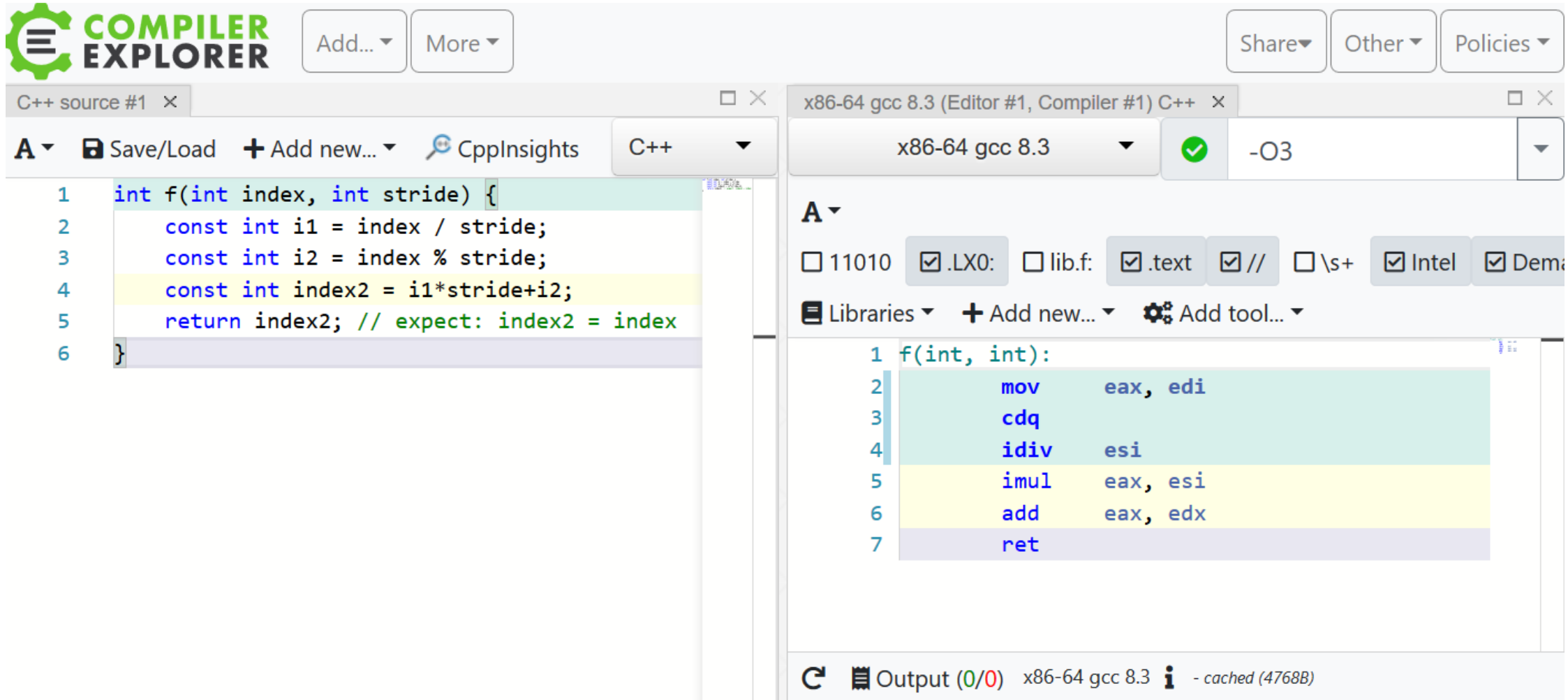
```
for (int index = 0; index < n1*n2; ++index)
{
  int i1 = index / n2;
  int i2 = index % n2;
  res[i1*n2+i2] = a[i1*n2+i2] - b[i1*n2+i2];
}
```

With n1=100,000 and n2=3

	penalty w.r.t. one flattened loop
gcc 9.1 -O3	+383%
clang 8.0 -O3	**+0%**


COMPILER EXPLORER
Add...
More
Share
Other
Policies
C++ source #1
Save/Load
Add new...
CppInsights
C++
1 int f(int index, int stride) {
2 const int i1 = index / stride;
3 const int i2 = index % stride;
4 const int index2 = i1*stride+i2;
5 return index2; // expect: index2 = index
6 }
x86-64 clang 8.0.0 (Editor #1, Compiler #1) C++
x86-64 clang 8.0.0
-O3
11010
.LX0:
lib.f:
.text
//
\s+
Intel
Libraries
Add new...
Add tool...
1 f(int, int):
2 mov eax, edi
3 ret
Output (0/0)
x86-64 clang 8.0.0
- cached (11449B)


COMPILER EXPLORER
Add...
More
Share
Other
Policies
C++ source #1
Save/Load
Add new...
CppInsights
C++
1 int f(int index, int stride) {
2 const int i1 = index / stride;
3 const int i2 = index % stride;
4 const int index2 = i1*stride+i2;
5 return index2; // expect: index2 = index
6 }
x64 msvc v19.14 (WINE) (Editor #1, Compiler #1) C++
x64 msvc v19.14 (WINE)
/O2
11010
.LX0:
lib.f:
.text
//
\s+
Intel
Libraries
Add new...
Add tool...
1 index$ = 8
2 stride$ = 16
3 int f(int,int) PROC
4 mov eax, ecx
5 ret 0
6 int f(int,int) ENDP
Output (1/0)
x64 msvc v19.14 (WINE)
- cached (470B)


COMPILER EXPLORER
Add...
More
Share
Other
Policies
C++ source #1
Save/Load
Add new...
CppInsights
C++
1 int f(int index, int stride) {
2 const int i1 = index / stride;
3 const int i2 = index % stride;
4 const int index2 = i1*stride+i2;
5 return index2; // expect: index2 = index
6 }
x86-64 gcc 8.3 (Editor #1, Compiler #1) C++
x86-64 gcc 8.3
-O3
11010
.LX0:
lib.f:
.text
//
\s+
Intel
Libraries
Add new...
Add tool...
1 f(int, int):
2 mov eax, edi
3 cdq
4 idiv esi
5 imul eax, esi
6 add eax, edx
7 ret
Output (0/0)
x86-64 gcc 8.3
- cached (4768B)

Bug 89518 - missed optimisation for array address calculations

Arnaud Desitter **2019-02-27 11:41:08 UTC** Description

Considering:

```
int f(int index, int stride) {
    const int i1 = index / stride;
    const int i2 = index % stride;
    const int index2 = i1*stride+i2;
    return index2; // expect: index2 = index
}
```

gcc 8.3 with "-O3" on x84_64 emits:

```
f(int, int):
        mov     eax, edi
        cdq
        idiv    esi
        imul    eax, esi
        add     eax, edx
        ret
```

By contrast, clang 7 with "-O3" emits

```
f(int, int):
        mov     eax, edi
        ret
```

MSVC 2017 with "/O2" emits:

```
int f(int,int)
        mov     eax, ecx
        ret     0
```

Is there a way to persuade gcc to simplify this expression at compile time?

Richard Biener 2019-02-27 11:54:44 UTC Comment 1

We do not have a (a / b) * b + (a % b) simplification rule. The following adds one:

```
Index: gcc/match.pd
===================================================================
--- gcc/match.pd        (revision 269242)
+++ gcc/match.pd        (working copy)
@@ -2729,6 +2729,13 @@ (define_operator_list COND_TERNARY
   (mult (convert1? (exact_div @0 @@1)) (convert2? @1))
   (convert @0))

+/* Simplify (A / B) * B + (A  % B) -> A.  */
+(for div (trunc_div ceil_div floor_div round_div)
+     mod (trunc_mod ceil_mod floor_mod round_mod)
+  (simplify
+   (plus:c (mult:c (div @0 @1) @1) (mod @0 @1))
+   @0))
+
 /* ((X /[ex] A) +- B) * A  -->  X +- A * B.  */
 (for op (plus minus)
  (simplify
```

Richard Biener 2019-02-27 11:54:44 UTC Comment 1

We do not have a (a / b) * b + (a % b) simplification rule. The following adds one:

```
Index: gcc/match.pd
===================================================================
--- gcc/match.pd        (revision 269242)
+++ gcc/match.pd        (working copy)
@@ -2729,6 +2729,13 @@ (define_operator_list COND_TERNARY
   (mult (convert1? (exact_div @0 @@1)) (convert2? @1))
   (convert @0)))

+/* Simplify (A / B) * B + (A % B) -> A.  */
+(for div (trunc_div ceil_div floor_div round_div)
+     mod (trunc_mod ceil_mod floor_mod round_mod)
+  (simplify
+   (plus:c (mult:c (div @0 @1) @1) (mod @0 @1))
+   @0))
+
 /* ((X /[ex] A) +- B) * A  -->  X +- A * B.  */
 (for op (plus minus)
  (simplify
```

13 minutes.

Thank you !

GCC 9.1 Released

- *From*: Jakub Jelinek <jakub at redhat dot com>
- *To*: gcc at gcc dot gnu dot org
- *Date*: Fri, 3 May 2019 13:43:28 +0200
- *Subject*: GCC 9.1 Released
- *Reply-to*: Jakub Jelinek <jakub at redhat dot com>

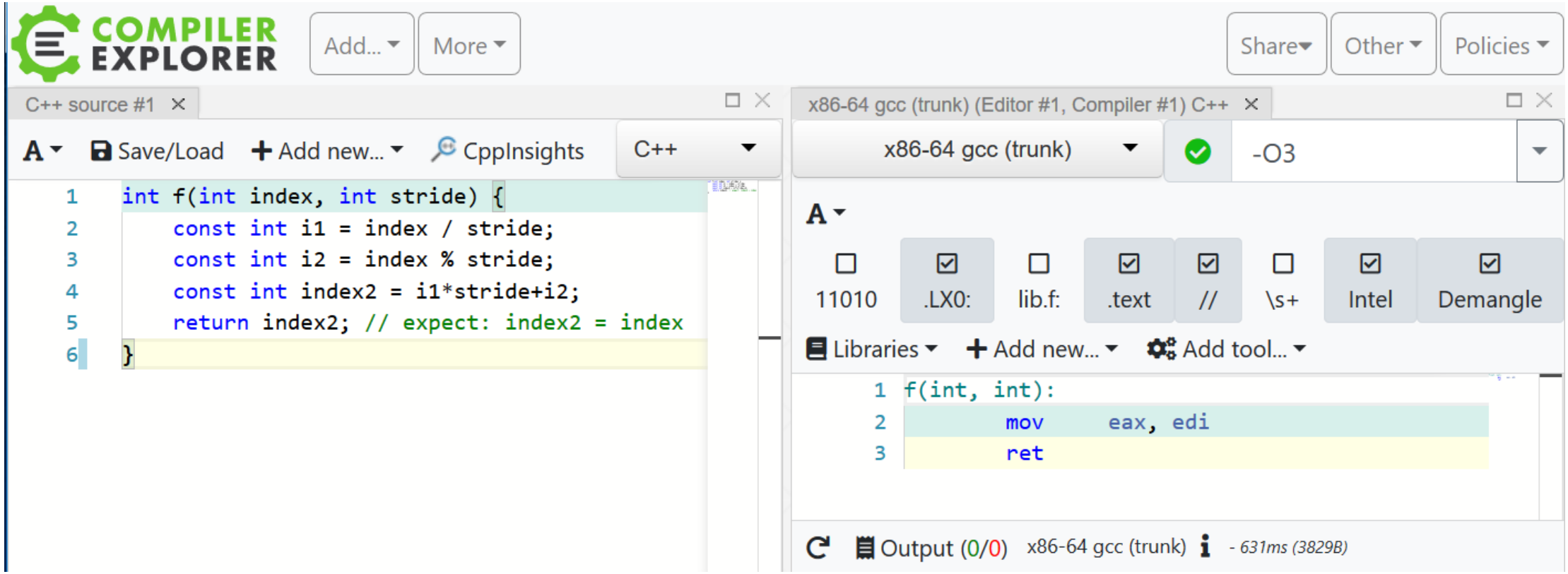
```
We are proud to announce the next, major release of the
GNU Compiler Collection.
```

Richard Biener 2019-05-03 10:46:44 UTC

```
Author: rguenth
Date: Fri May  3 10:46:13 2019
New Revision: 270846

URL: https://gcc.gnu.org/viewcvs?rev=270846&root=gcc&view=rev
Log:
2019-05-03  Richard Biener  <rguenther@suse.de>

        PR middle-end/89518
        * match.pd: Add pattern to optimize (A / B) * B + (A % B) to A.

        * gcc.dg/pr89518.c: New testcase.

Added:
    trunk/gcc/testsuite/gcc.dg/pr89518.c
Modified:
    trunk/gcc/ChangeLog
    trunk/gcc/match.pd
    trunk/gcc/testsuite/ChangeLog
```

Richard Biener 2019-05-03 10:48:52 UTC

Fixed.


COMPILER EXPLORER
Add...
More
Share
Other
Policies
C++ source #1
Save/Load
Add new...
CppInsights
C++
1 int f(int index, int stride) {
2 const int i1 = index / stride;
3 const int i2 = index % stride;
4 const int index2 = i1*stride+i2;
5 return index2; // expect: index2 = index
6 }
x86-64 gcc (trunk) (Editor #1, Compiler #1) C++
x86-64 gcc (trunk)
-O3
11010
.LX0:
lib.f:
.text
//
\s+
Intel
Demangle
Libraries
Add new...
Add tool...
1 f(int, int):
2 mov eax, edi
3 ret
Output (0/0)
x86-64 gcc (trunk)
- 631ms (3829B)

With n1=100,000 and n2=3

	penalty w.r.t. one flattened loop
gcc 9.1 -O3	+383%
gcc trunk -O3	**+0%**
clang 8.0 -O3	**+0%**