#pragma omp [begin] assume[s]

Johannes Doerfert, Argonne National Laboratory
#pragma omp [begin] assume[s]

Or, how to talk to your compiler(s).
Motivation

How to convey information to the compiler beyond types?

Compilers have non-portable “built-in assumes”:

MSVC: __assume(<expression>)

GCC: __builtin_assume_aligned(<expression>, <alignment>, ...)  

Clang: __builtin_assume_aligned(<expression>, <alignment>, ...)  
       __builtin_assume(<expression>)
The OpenMP assume directive

Different “kinds” of properties can be expressed:

1) Provide a portable low-level “built-in assume”:
   #pragma omp assume holds(<expression>)

2) Provide high-level “assumptions” for better code generation:
   #pragma omp assume no_openmp absent(<directive>) ...

3) Provide a portable hook for extensions:
   #pragma omp assume ext_IMPLEMENTATION_DEFINED
The OpenMP assume directive

Several forms/spelling, always affects code transitively:

1) **Global scope**, affect “everything”:
   
   `#pragma omp assume <clause>`

2) **Begin-end range scope** for declarations:
   
   `#pragma omp begin assume <clause>`
   `#pragma omp end assumes`

3) **Structured-block scope** inside functions:
   
   `#pragma omp assume <clause>`
   `<structured-block>`
for (int i = 0; i < N; ++i) {
    A[i] += B[i];
}

Use Cases [1/3]
// To indicate vectorization is legal
#pragma omp simd
for (int i = 0; i < N; ++i) {
    A[i] += B[i];
}
Use Cases [1/3]

// To indicate no remainder loop is necessary (for VF=8)
#pragma omp assume holds(N % 8 == 0 && N > 0)

// To indicate vectorization is legal
#pragma omp simd
for (int i = 0; i < N; ++i) {
    A[i] += B[i];
}
Use Cases [2/3]                      [implementation specific]

#pragma omp target teams
{
    // some C++ code with function calls that “could” throw.
}
Use Cases [2/3] [implementation specific]

// Tell the compiler the C++ code on the target will not
// throw exceptions. No need to proof it, it can be assumed.
// Will not affect host compilation (unlike -fno-exceptions).
#pragma omp assumes ext_target_no_exceptions

#pragma omp target teams
{
    // some C++ code with function calls that “could” throw.
}

NOTE: This is an illustrative example and not implemented (yet).
Use Cases [3/3]

#pragma omp target teams
foo();
Use Cases [3/3]

```c
#pragma omp target teams
foo();

void foo() {
    #pragma omp parallel for
    { ... }
}
```
Use Cases [3/3]                      [implementation specific]

#pragma omp target teams
foo();

void foo() {
    #pragma omp parallel for
    { ... }
}

>> clang -Rpass=openmp -fopenmp -fopenmp-targets=nvptx64 foo.c
foo.c:5:3: Parallel region could be called from external target region, will not rewrite the state machine use. Employ `static` linkage for `foo` or add `#pragma omp assume ext_no_external_target_region_caller`.

NOTE: This is an illustrative example and not implemented (yet).
LLVM/Clang Implementation

LLVM/Clang will support `assume` in version 12, development branch before already.

Try the `llvm-openmp-advisor` for code analyses and `assume` suggestions.

Visit [http://openmp.llvm.org/docs/](http://openmp.llvm.org/docs/) for more information, implementation status, extensions, use cases, remarks, ...

Reach out if this sounds interesting and you want to know more or collaborate!

Johannes Doerfert <johannesdoerfert@gmail.com>
openmp.org  OpenMP API specs, forum, reference guides, and more

link.openmp.org/sc20  Videos and PDFs of OpenMP SC’20 presentations