A struct, enum, or union with the `repr(transparent)` representation hint contains a zero-sized field that requires non-trivial alignment.

Erroneous code example:

```
#![feature(repr_align)]

#[repr(align(32))]
struct ForceAlign32;

#[repr(transparent)]
struct Wrapper(f32, ForceAlign32); // error: zero-sized field in transparent
                                   //        struct has alignment larger than 1
```

A transparent struct, enum, or union is supposed to be represented exactly like the piece of data it contains. Zero-sized fields with different alignment requirements potentially conflict with this property. In the example above, `Wrapper` would have to be aligned to 32 bytes even though `f32` has a smaller alignment requirement.

Consider removing the over-aligned zero-sized field:

```
#[repr(transparent)]
struct Wrapper(f32);
```

Alternatively, `PhantomData<T>` has alignment 1 for all `T`, so you can use it if you need to keep the field for some reason:

```
#![feature(repr_align)]

use std::marker::PhantomData;

#[repr(align(32))]
struct ForceAlign32;

#[repr(transparent)]
struct Wrapper(f32, PhantomData<ForceAlign32>);
```

Note that empty arrays `[T; 0]` have the same alignment requirement as the element type `T`. Also note that the error is conservatively reported even when the alignment of the zero-sized type is less than or equal to the data field's alignment.