```
instance (Eq a) => Eq (Foo a) where {
   {-# SPECIALIZE instance Eq (Foo [(Int, Bar)]) #-}
   ... usual stuff ...
 }
```

The pragma must occur inside the `where` part of the instance declaration.

7.22.11 UNPACK pragma

The `UNPACK` indicates to the compiler that it should unpack the contents of a constructor field into the constructor itself, removing a level of indirection. For example:

```
data T = T {-# UNPACK #-} !Float
           {-# UNPACK #-} !Float
```

will create a constructor `T` containing two unboxed floats. This may not always be an optimisation: if the `T` constructor is scrutinised and the floats passed to a non-strict function for example, they will have to be reboxed (this is done automatically by the compiler).

Unpacking constructor fields should only be used in conjunction with `-O`[1], in order to expose unfoldings to the compiler so the reboxing can be removed as often as possible. For example:

```
f :: T -> Float
f (T f1 f2) = f1 + f2
```

The compiler will avoid reboxing `f1` and `f2` by inlining `+` on floats, but only when `-O` is on.

Any single-constructor data is eligible for unpacking; for example

```
data T = T {-# UNPACK #-} !(Int,Int)
```

will store the two `Int`s directly in the `T` constructor, by flattening the pair. Multi-level unpacking is also supported:

```
data T = T {-# UNPACK #-} !S
data S = S {-# UNPACK #-} !Int {-# UNPACK #-} !Int
```

will store two unboxed `Int#`s directly in the `T` constructor. The unpacker can see through newtypes, too.

See also the `-funbox-strict-fields` flag, which essentially has the effect of adding `{-# UNPACK #-}` to every strict constructor field.

7.22.12 NOUNPACK pragma

The `NOUNPACK` pragma indicates to the compiler that it should not unpack the contents of a constructor field. Example:

```
data T = T {-# NOUNPACK #-} !(Int,Int)
```

Even with the flags `-funbox-strict-fields` and `-O`, the field of the constructor `T` is not unpacked.

7.22.13 SOURCE pragma

The `{-#SOURCE #-}` pragma is used only in `import` declarations, to break a module loop. It is described in detail in Section 4.7.9.

[1] in fact, UNPACK has no effect without `-O`, for technical reasons (see tick 5252)