Discussion: [RFC] Should We Restrict the Usage of 0-D Vectors in the Vector Dialect?

Today: Inconsistency

```
vector.extract ... : vector<2xf32> from vector<2x2xf32>

vector.extract ... : f32 from vector<2x2xf32>


vector.insert ... : vector<2xf32> into vector<2x2xf32>

vector.insert ... : vector<f32> into vector<2x2xf32>

vector.insert ... : f32 into vector<2x2xf32>
```

Possible ways to get consistency: Option 1

```
vector.extract ... : vector<2xf32> from vector<2x2xf32>
vector.extract ... : vector<f32> into vector<2x2xf32>
vector.extract ... : f32 from vector<2x2xf32>

vector.insert ... : vector<2xf32> into vector<2x2xf32>
vector.insert ... : vector<f32> into vector<2x2xf32>
vector.insert ... : f32 into vector<2x2xf32>
```

Possible ways to get consistency: Option 2

```
vector.extract ... : vector<2xf32> from vector<2x2xf32>

vector.extract ... : f32 from vector<2x2xf32>


vector.insert ... : vector<2xf32> into vector<2x2xf32>

vector.insert ... : vector<f32> into vector<2x2xf32>

vector.insert ... : f32 into vector<2x2xf32>
```

Possible ways to get consistency: Option 1

```
vector.extract ... : vector<2xf32> from vector<2x2xf32>

vector.extract ... : vector<f32> into vector<2x2xf32>

vector.extract ... : f32 from vector<2x2xf32>


vector.insert ... : vector<2xf32> into vector<2x2xf32>

vector.insert ... : vector<f32> into vector<2x2xf32>

vector.insert ... : f32 into vector<2x2xf32>
```

A lot of transformations and patterns currently expect vector.extract to degenerate to the scalar case instead of a 0D vector : very hard to unwind

Possible ways to get consistency: Option 2

```
vector.extract ... : vector<2xf32> from vector<2x2xf32>

vector.extract ... : f32 from vector<2x2xf32>


vector.insert ... : vector<2xf32> into vector<2x2xf32>

vector.insert ... : vector<f32> into vector<2x2xf32>

vector.insert ... : f32 into vector<2x2xf32>
```

Not a lot of operations today produce 0D vectors (pending work) and maintaining 0D vector insertion adds a lot of extra effort.

Short Term

Option 2 is good, it adds consistency with less work and brings us to a good starting place to start doing things correctly.

~~`vector.insert ... : vector<f32> into vector<2x2xf32>`~~

Long Term

Neither Option

Why?

```
vector.extract/vector.insert behavior

vector.extract ... : vector<2xf32> from vector<2x2xf32> // slicing
vector.extract ... : f32 from vector<2x2xf32> // extraction
```

Ok when we didn’t have 0D vectors

Ambiguous when we have 0D vectors

Tensor Dialect: Proper 0D support

```
// slicing
tensor.extract_slice ... : tensor<2xf32> from tensor<2x2xf32>
tensor.extract_slice ... : tensor<f32> from tensor<2x2xf32>

// extraction
tensor.extract ... : f32 from tensor<2x2xf32>
```

My Proposal (In RFC Comments)

Split vector.extract/vector.insert:

```
vector.extract_slice // slicing
vector.extract // extraction

vector.insert_slice // slicing
vector.insert // insertion
```

Overall

Both proposals are complementary

Short Term: Andrej's RFC
Long Term: My Proposal (in comments of RFC)