

---

# IREE Python API

*Release snapshot*

**IREE Authors**

**Sep 23, 2023**



## API DOCS

<b>1</b>	<b>Runtime API</b>	<b>1</b>
<b>2</b>	<b>Compiler API</b>	<b>11</b>
2.1	In-Process Compiler API . . . . .	11
2.2	Invoking Command Line Tools . . . . .	11
2.3	IREE Dialect . . . . .	17
2.4	IREE Python DataModel Dialect . . . . .	17
2.5	MLIR Core Dialects . . . . .	17
2.6	TensorFlow Dialects . . . . .	72
2.7	Embedded MLIR API . . . . .	72
<b>3</b>	<b>Indices and tables</b>	<b>95</b>
	<b>Python Module Index</b>	<b>97</b>
	<b>Index</b>	<b>99</b>



## RUNTIME API

Module init for the python bindings.

```
class iree.runtime.BufferCompatibility
```

Members:

NONE

ALLOCATABLE

IMPORTABLE

EXPORTABLE

QUEUE\_TRANSFER

QUEUE\_DISPATCH

**ALLOCATABLE** = <BufferCompatibility.ALLOCATABLE: 1>

**EXPORTABLE** = <BufferCompatibility.EXPORTABLE: 4>

**IMPORTABLE** = <BufferCompatibility.IMPORTABLE: 2>

**NONE** = <BufferCompatibility.NONE: 0>

**QUEUE\_DISPATCH** = <BufferCompatibility.QUEUE\_DISPATCH: 2048>

**QUEUE\_TRANSFER** = <BufferCompatibility.QUEUE\_TRANSFER: 1024>

property name

property value

```
class iree.runtime.BufferUsage
```

Members:

NONE

TRANSFER\_SOURCE

TRANSFER\_TARGET

TRANSFER

DISPATCH\_INDIRECT\_PARAMS

DISPATCH\_UNIFORM\_READ

DISPATCH\_STORAGE\_READ

DISPATCH\_STORAGE\_WRITE

DISPATCH\_STORAGE

```

DISPATCH_IMAGE_READ
DISPATCH_IMAGE_WRITE
DISPATCH_IMAGE
SHARING_EXPORT
SHARING_REPLICATE
SHARING_CONCURRENT
SHARING_IMMUTABLE
MAPPING_SCOPED
MAPPING_PERSISTENT
MAPPING_OPTIONAL
MAPPING_ACCESS_RANDOM
MAPPING_ACCESS_SEQUENTIAL_WRITE
MAPPING
DEFAULT
DEFAULT = <BufferUsage.DEFAULT: 3075>
DISPATCH_IMAGE = <BufferUsage.DISPATCH_IMAGE: 12288>
DISPATCH_IMAGE_READ = <BufferUsage.DISPATCH_IMAGE_READ: 4096>
DISPATCH_IMAGE_WRITE = <BufferUsage.DISPATCH_IMAGE_WRITE: 8192>
DISPATCH_INDIRECT_PARAMS = <BufferUsage.DISPATCH_INDIRECT_PARAMS: 256>
DISPATCH_STORAGE = <BufferUsage.DISPATCH_STORAGE: 3072>
DISPATCH_STORAGE_READ = <BufferUsage.DISPATCH_STORAGE_READ: 1024>
DISPATCH_STORAGE_WRITE = <BufferUsage.DISPATCH_STORAGE_WRITE: 2048>
DISPATCH_UNIFORM_READ = <BufferUsage.DISPATCH_UNIFORM_READ: 512>
MAPPING = <BufferUsage.MAPPING: 150994944>
MAPPING_ACCESS_RANDOM = <BufferUsage.MAPPING_ACCESS_RANDOM: 134217728>
MAPPING_ACCESS_SEQUENTIAL_WRITE = <BufferUsage.MAPPING_ACCESS_SEQUENTIAL_WRITE:
268435456>
MAPPING_OPTIONAL = <BufferUsage.MAPPING_OPTIONAL: 67108864>
MAPPING_PERSISTENT = <BufferUsage.MAPPING_PERSISTENT: 33554432>
MAPPING_SCOPED = <BufferUsage.MAPPING_SCOPED: 16777216>
NONE = <BufferUsage.NONE: 0>
SHARING_CONCURRENT = <BufferUsage.SHARING_CONCURRENT: 262144>
SHARING_EXPORT = <BufferUsage.SHARING_EXPORT: 65536>
SHARING_IMMUTABLE = <BufferUsage.SHARING_IMMUTABLE: 524288>
SHARING_REPLICATE = <BufferUsage.SHARING_REPLICATE: 131072>
TRANSFER = <BufferUsage.TRANSFER: 3>

```

**TRANSFER\_SOURCE** = <BufferUsage.TRANSFER\_SOURCE: 1>

**TRANSFER\_TARGET** = <BufferUsage.TRANSFER\_TARGET: 2>

**property name**

**property value**

```
class iree.runtime.Config(driver_name: Optional[str] = None, *, device: Optional[iree._runtime.HalDevice]
                        = None, tracer: Optional[iree.runtime.tracing.Tracer] = None)
```

System configuration.

**default\_vm\_modules:** Tuple[iree.\_runtime.VmModule, ...]

**device:** iree.\_runtime.HalDevice

**tracer:** Optional[iree.runtime.tracing.Tracer]

**vm\_instance:** iree.\_runtime.VmInstance

```
class iree.runtime.DeviceArray(device: iree._runtime.HalDevice, buffer_view:
                              iree._runtime.HalBufferView, implicit_host_transfer: bool = False,
                              override_dtype=None)
```

An IREE device array.

**Device arrays can be in one of two states:**

1. Host accessible: The array will be backed by host accessible memory and can have the usual things done with it that one expects to be able to do with an ndarray.
2. Device resident: The array is just a handle to a device resident Buffer (and BufferView wrapper). Metadata about the array are accessible (shape and dtype) but anything that touches the data cannot be accessed in this state.

How a device array comes into existence controls how it can transition between these states:

- A user can create a DeviceArray explicitly with a device allocator. Such an array will not be implicitly convertible to host accessible, although accessors exist to do so.
- When created by the platform with a synchronization policy, then implicit transfer back to the host will trigger appropriate waits and be performed automatically (this is the common case for function return values if not otherwise configured, as an example).

**astype**(dtype, casting='unsafe', copy=True)

**property dtype**

**property is\_host\_accessible**

Whether this array is currently host accessible.

**reshape**(\*args)

**property shape**

**to\_host**() → numpy.ndarray

```
class iree.runtime.FunctionInvoker(vm_context: iree._runtime.VmContext, device:
                                  iree._runtime.HalDevice, vm_function: iree._runtime.VmFunction,
                                  tracer: Optional[iree.runtime.tracing.ContextTracer])
```

Wraps a VmFunction, enabling invocations against it.

**property vm\_function:** iree.\_runtime.VmFunction

```
class iree.runtime.HalAllocator
```

**allocate\_buffer**(*self*: iree.\_runtime.HalAllocator, *memory\_type*: int, *allowed\_usage*: int, *allocation\_size*: int) → iree::python::HalBuffer

Allocates a new buffer with requested characteristics (does not initialize with specific data).

**allocate\_buffer\_copy**(*self*: iree.\_runtime.HalAllocator, *memory\_type*: int, *allowed\_usage*: int, *buffer*: object, *element\_type*: Optional[iree.\_runtime.HalElementType] = None) → object

Allocates a new buffer and initializes it from a Python buffer object. If an element type is specified, wraps in a BufferView matching the characteristics of the Python buffer. The format is requested as ND/C-Contiguous, which may incur copies if not already in that format.

**property formatted\_statistics**

**property has\_statistics**

**query\_buffer\_compatibility**(*self*: iree.\_runtime.HalAllocator, *memory\_type*: int, *allowed\_usage*: int, *intended\_usage*: int, *allocation\_size*: int) → int

**property statistics**

**trim**(*self*: iree.\_runtime.HalAllocator) → None

**class iree.runtime.HalBuffer**

**create\_view**(*self*: iree.\_runtime.HalBuffer, *shape*: iree::python::HalShape, *element\_size*: int) → iree::python::HalBufferView

**fill\_zero**(*self*: iree.\_runtime.HalBuffer, *byte\_offset*: int, *byte\_length*: int) → None

**class iree.runtime.HalBufferView**

**property element\_type**

**map**(*self*: iree.\_runtime.HalBufferView) → iree::python::HalMappedMemory

**property ref**

**property shape**

**class iree.runtime.HalDevice**

**property allocator**

**begin\_profiling**(*self*: iree.\_runtime.HalDevice, *\*\*kwargs*) → None

**end\_profiling**(*self*: iree.\_runtime.HalDevice) → None

**class iree.runtime.HalDriver**

**create\_default\_device**(*self*: iree.\_runtime.HalDriver, *\*\*kwargs*) → iree.\_runtime.HalDevice

**create\_device**(\*args, *\*\*kwargs*)

Overloaded function.

1. **create\_device**(*self*: iree.\_runtime.HalDriver, *arg0*: int, *\*\*kwargs*) → iree.\_runtime.HalDevice

2. **create\_device**(*self*: iree.\_runtime.HalDriver, *arg0*: dict, *\*\*kwargs*) → iree.\_runtime.HalDevice

**create\_device\_by\_uri**(*self*: iree.\_runtime.HalDriver, *arg0*: str, *\*\*kwargs*) → iree.\_runtime.HalDevice

**static query**() → List[str]

**query\_available\_devices**(*self*: iree.\_runtime.HalDriver) → list



```

class iree.runtime.HalElementType
    Members:
    NONE
    OPAQUE_8
    OPAQUE_16
    OPAQUE_32
    OPAQUE_64
    BOOL_8
    INT_4
    INT_8
    INT_16
    INT_32
    INT_64
    SINT_4
    SINT_8
    SINT_16
    SINT_32
    SINT_64
    UINT_4
    UINT_8
    UINT_16
    UINT_32
    UINT_64
    FLOAT_16
    FLOAT_32
    FLOAT_64
    BFLOAT_16
    COMPLEX_64
    COMPLEX_128
    BFLOAT_16 = <HalElementType.BFLOAT_16: 570425360>
    BOOL_8 = <HalElementType.BOOL_8: 318767112>
    COMPLEX_128 = <HalElementType.COMPLEX_128: 587202688>
    COMPLEX_64 = <HalElementType.COMPLEX_64: 587202624>
    FLOAT_16 = <HalElementType.FLOAT_16: 553648144>
    FLOAT_32 = <HalElementType.FLOAT_32: 553648160>
    FLOAT_64 = <HalElementType.FLOAT_64: 553648192>

```

```

INT_16 = <HalElementType.INT_16: 268435472>
INT_32 = <HalElementType.INT_32: 268435488>
INT_4 = <HalElementType.INT_4: 268435460>
INT_64 = <HalElementType.INT_64: 268435520>
INT_8 = <HalElementType.INT_8: 268435464>
NONE = <HalElementType.NONE: 0>
OPAQUE_16 = <HalElementType.OPAQUE_16: 16>
OPAQUE_32 = <HalElementType.OPAQUE_32: 32>
OPAQUE_64 = <HalElementType.OPAQUE_64: 64>
OPAQUE_8 = <HalElementType.OPAQUE_8: 8>
SINT_16 = <HalElementType.SINT_16: 285212688>
SINT_32 = <HalElementType.SINT_32: 285212704>
SINT_4 = <HalElementType.SINT_4: 285212676>
SINT_64 = <HalElementType.SINT_64: 285212736>
SINT_8 = <HalElementType.SINT_8: 285212680>
UINT_16 = <HalElementType.UINT_16: 301989904>
UINT_32 = <HalElementType.UINT_32: 301989920>
UINT_4 = <HalElementType.UINT_4: 301989892>
UINT_64 = <HalElementType.UINT_64: 301989952>
UINT_8 = <HalElementType.UINT_8: 301989896>
static map_to_dtype(arg0: int) → object
property name
property value

```

```

class iree.runtime.Linkage
  Members:
    INTERNAL
    IMPORT
    IMPORT_OPTIONAL
    EXPORT
    EXPORT = <Linkage.EXPORT: 2>
    IMPORT = <Linkage.IMPORT: 1>
    IMPORT_OPTIONAL = <Linkage.IMPORT_OPTIONAL: 3>
    INTERNAL = <Linkage.INTERNAL: 0>
    property name
    property value

```

```
class iree.runtime.MemoryAccess
```

Members:

NONE

READ

WRITE

DISCARD

DISCARD\_WRITE

ALL

ALL = <MemoryAccess.ALL: 7>

DISCARD = <MemoryAccess.DISCARD: 4>

DISCARD\_WRITE = <MemoryAccess.DISCARD\_WRITE: 6>

NONE = <MemoryAccess.NONE: 0>

READ = <MemoryAccess.READ: 1>

WRITE = <MemoryAccess.WRITE: 2>

property name

property value

```
class iree.runtime.MemoryType
```

Members:

NONE

OPTIMAL

HOST\_VISIBLE

HOST\_COHERENT

HOST\_CACHED

HOST\_LOCAL

DEVICE\_VISIBLE

DEVICE\_LOCAL

DEVICE\_LOCAL = <MemoryType.DEVICE\_LOCAL: 48>

DEVICE\_VISIBLE = <MemoryType.DEVICE\_VISIBLE: 16>

HOST\_CACHED = <MemoryType.HOST\_CACHED: 8>

HOST\_COHERENT = <MemoryType.HOST\_COHERENT: 4>

HOST\_LOCAL = <MemoryType.HOST\_LOCAL: 70>

HOST\_VISIBLE = <MemoryType.HOST\_VISIBLE: 2>

NONE = <MemoryType.NONE: 0>

OPTIMAL = <MemoryType.OPTIMAL: 1>

property name

property value

```

class iree.runtime.PyModuleInterface

    create(self: iree._runtime.PyModuleInterface) → iree::python::VmModule
    property destroyed
    export(self: iree._runtime.PyModuleInterface, name: str, cconv: str, callable: object) → None
    property initialized

class iree.runtime.Shape

class iree.runtime.SystemContext(vm_modules=None, config: Optional[iree.runtime.system_api.Config] =
                                None)
    Global system.
    add_vm_module(vm_module)
    add_vm_modules(vm_modules)
    property config: iree.runtime.system_api.Config
    property instance: iree._runtime.VmInstance
    property is_dynamic: bool
    property modules: iree.runtime.system_api.BoundModules
    property vm_context: iree._runtime.VmContext

class iree.runtime.Tracer(trace_path: str)
    Object for tracing calls made into the runtime.
    get_unique_name(local_name: str) → str
    persist_vm_module(vm_module: iree._runtime.VmModule) → iree.runtime.tracing.TracedModule

class iree.runtime.VmBuffer

    property ref

class iree.runtime.VmContext

    property context_id
    invoke(self: iree._runtime.VmContext, arg0: iree._runtime.VmFunction, arg1:
           iree._runtime.VmVariantList, arg2: iree._runtime.VmVariantList) → None
    register_modules(self: iree._runtime.VmContext, arg0: List[iree::python::VmModule]) → None

class iree.runtime.VmFunction

    property linkage
    property module_name
    property name
    property ordinal
    property reflection

class iree.runtime.VmInstance

```

```
class iree.runtime.VmModule
```

```
    static from_flatbuffer(arg0: iree._runtime.VmInstance, arg1: object) → iree._runtime.VmModule
```

```
    property function_names
```

```
    lookup_function(self: iree._runtime.VmModule, name: str, linkage: iree._runtime.Linkage =
                    <Linkage.EXPORT: 2>) → Optional[iree._runtime.VmFunction]
```

```
    property name
```

```
    property stashed_flatbuffer_blob
```

```
    property version
```

```
class iree.runtime.VmVariantList
```

```
    get_as_list(self: iree._runtime.VmVariantList, arg0: int) → object
```

```
    get_as_object(self: iree._runtime.VmVariantList, arg0: int, arg1: object) → object
```

```
    get_as_ref(self: iree._runtime.VmVariantList, arg0: int) → object
```

```
    get_serialized_trace_value(self: iree._runtime.VmVariantList, arg0: int) → object
```

```
    get_variant(self: iree._runtime.VmVariantList, arg0: int) → object
```

```
    push_float(self: iree._runtime.VmVariantList, arg0: float) → None
```

```
    push_int(self: iree._runtime.VmVariantList, arg0: int) → None
```

```
    push_list(self: iree._runtime.VmVariantList, arg0: iree._runtime.VmVariantList) → None
```

```
    push_ref(self: iree._runtime.VmVariantList, arg0: handle) → None
```

```
    property ref
```

```
    property size
```

```
iree.runtime.asdevicearray(device: iree._runtime.HalDevice, a, dtype=None, *, implicit_host_transfer: bool
                           = False, memory_type=<MemoryType.DEVICE_LOCAL: 48>,
                           allowed_usage=<BufferUsage.???: 150998019>, element_type:
                           Optional[iree._runtime.HalElementType] = None) →
                           iree.runtime.array_interop.DeviceArray
```

Helper to create a DeviceArray from an arbitrary array like.

This is similar in purpose and usage to `np.asarray`, except that it takes a device as the first argument. This may not be the best mechanism for getting a DeviceArray, depending on your use case, but it is reliable and simple. This function may make a defensive copy or cause implicit transfers to satisfy the request. If this is important to you, then a lower level API is likely more appropriate.

Note that additional flags `memory_type`, `allowed_usage` and `element_type` are only hints if creating a new DeviceArray. If `a` is already a DeviceArray, they are ignored.

```
iree.runtime.benchmark_exe()
```

```
iree.runtime.benchmark_module(module, entry_function=None, inputs=[], **kwargs)
```

```
iree.runtime.create_hal_module(arg0: iree::python::VmInstance, arg1: iree::python::HalDevice) →
                             iree::python::VmModule
```

```
iree.runtime.get_default_tracer() → Optional[iree.runtime.tracing.Tracer]
```

Gets a default call tracer based on environment variables.

`iree.runtime.get_device(device_uri: str, cache: bool = True) → iree._runtime.HalDevice`

Gets a cached device by URI.

**Parameters**

- **device\_uri** – The URI of the device, either just a driver name for the default or a fully qualified “driver://path?params”.
- **cache** – Whether to cache the device (default True).

**Returns** A HalDevice.

`iree.runtime.get_driver(device_uri: str) → iree._runtime.HalDriver`

Returns a HAL driver by device\_uri (or driver name).

`iree.runtime.get_first_device(device_uris: Optional[Sequence[str]] = None, cache: bool = True) → iree._runtime.HalDevice`

Gets the first valid (cached) device for a prioritized list of names.

If no driver\_names are given, and an environment variable of IREE\_DEFAULT\_DEVICE is available, then it is treated as a comma delimited list of driver names to try.

This is meant to be used for default/automatic startup and is not suitable for any kind of multi-device setup.

**Parameters**

- **device\_uris** – Explicit list of device URIs to try.
- **cache** – Whether to cache the device (default True).

**Returns** A HalDevice instance.

`iree.runtime.load_vm_flatbuffer(vm_flatbuffer: bytes, *, driver: Optional[str] = None, backend: Optional[str] = None) → iree.runtime.system_api.BoundModule`

Loads a VM Flatbuffer into a callable module.

Either ‘driver’ or ‘backend’ must be specified.

`iree.runtime.load_vm_flatbuffer_file(path: str, *, driver: Optional[str] = None, backend: Optional[str] = None) → iree.runtime.system_api.BoundModule`

Loads a file containing a VM Flatbuffer into a callable module.

Either ‘driver’ or ‘backend’ must be specified.

`iree.runtime.load_vm_module(vm_module, config: Optional[iree.runtime.system_api.Config] = None)`

Loads a VmModule into a new SystemContext and returns it.

`iree.runtime.load_vm_modules(*vm_modules, config: Optional[iree.runtime.system_api.Config] = None)`

Loads VmModules into a new SystemContext and returns them.

`iree.runtime.normalize_value(value: Any) → Optional[Union[numpy.ndarray, List[Any], Tuple[Any]]]`

Normalizes the given value for input to (or comparison with) IREE.

`iree.runtime.query_available_drivers() → Collection[str]`

Returns a collection of driver names that are available.

## COMPILER API

### 2.1 In-Process Compiler API

IREE provides access to its MLIR-based compiler via a dedicated set of APIs presented here.

### 2.2 Invoking Command Line Tools

As with many compilers, IREE's compiler consists of many command line tools, some of which are designed for compiler devs and are only accessible via source builds. User level tools are distributed via the Python packages and are also accessible via dedicated Python APIs, documented here.

#### 2.2.1 Core Compiler (*iree-compile*)

This module contains Python wrappers for various IREE command-line tools.

This top-level API provides access to the *iree-compile* tool, which compiles MLIR ASM via IREE's compiler to a supported output format (i.e. VM FlatBuffer, C source code, etc).

#### Example

```
import iree.compiler.tools

SIMPLE_MUL_ASM = """
func.func @simple_mul(%arg0: tensor<4xf32>, %arg1: tensor<4xf32>) -> tensor<4xf32> {
    %0 = "tosa.mul"(%arg0, %arg1) {shift = 0 : i32} : (tensor<4xf32>, tensor<4xf32>) ->
    ↪ tensor<4xf32>
    return %0 : tensor<4xf32>
}
"""

# Also see compile_file()
# There are many keyword options available.
# See iree.compiler.CompilerOptions
binary = iree.compiler.tools.compile_str(
    SIMPLE_MUL_ASM, input_type="tosa", target_backends=["llvm-cpu"])
```

```
iree.compiler.tools.compile_file(input_file: str, **kwargs)
    Invokes the IREE compiler on an input file.
```

**Parameters**

- **input\_file** – File containing MLIR assembly to compile.
- **\*\*kwargs** – Keyword arguments corresponding to `CompilerOptions`.

**Returns** Either a byte buffer of the compiled content or `None` if `output_file` was specified in the options.

```
iree.compiler.tools.compile_str(input_str: Union[str, bytes], **kwargs)
```

Invokes the IREE compiler with an input string.

**Parameters**

- **input\_str** – MLIR assembly to parse/compile (str or bytes).
- **\*\*kwargs** – Keyword arguments corresponding to `CompilerOptions`.

**Returns** Either a byte buffer of the compiled content or `None` if `output_file` was specified in the options.

```
class iree.compiler.tools.CompilerOptions(output_file: Optional[str] = None, target_backends:
    Sequence[str] = (), input_type:
    Union[iree.compiler.tools.core.InputType, str] =
    InputType.NONE, output_format:
    Union[iree.compiler.tools.core.OutputFormat, str] =
    OutputFormat.FLATBUFFER_BINARY, extra_args:
    Sequence[str] = (), optimize: bool = True,
    output_mlir_debuginfo: bool = True, output_generic_mlir:
    bool = False, extended_diagnostics: bool = False,
    strip_debug_ops: bool = False, strip_source_map: bool =
    False, crash_reproducer_path: Optional[str] = None,
    enable_tfLite_bindings: bool = False, enable_benchmark:
    bool = False)
```

Options to the compiler backend.

**Parameters**

- **output\_file** – Optionally save the compiled binary to a file instead of returning it.
- **target\_backends** – List of str names of target backends to compile into the binary. The resulting binary will run on targets that match one or more of the compiled backends.
- **input\_type** – The type of input legalization to perform prior to full compilation. Values can either be an `InputType` enum value or a case-insensitive name. Defaults to `InputType.NONE`.
- **output\_format** – Override the output format. See the `OutputFormat` enum. Values can either be an enum value or a case-insensitive name of the option. Typically used for debugging Defaults to `OutputFormat.FLATBUFFER_BINARY`.
- **extra\_args** – Optional list of additional arguments to pass to the compiler. Example: `["-mlir-print-ir-after-all", "-some-other-arg"]`. Individual arguments must be separate items in the list.
- **optimize** – Whether to apply some default high level optimizations (default `True`).
- **output\_mlir\_debuginfo** – Include debuginfo (including paths) in any saved or returned MLIR.
- **output\_generic\_mlir** – Use the generic (and more portable) MLIR formatting for any saved or returned MLIR instead of the per-dialect custom assembly.



- **extended\_diagnostics** – Outputs extended information on diagnostics, potentially outputting very verbosely (defaults to False).
- **strip\_debug\_ops** – Whether to strip high level operations used to aid debugging.
- **strip\_source\_map** – Whether to strip source map information (used to generate better errors).
- **crash\_reproducer\_path** – File name to output an MLIR crash dump to if there is a compiler failure.
- **enable\_tflite\_bindings** – Support the IREE TFLite runtime bindings API shim.
- **enable\_benchmark** – Whether to generate instrumented binaries suitable for benchmarking.

**enum** iree.compiler.tools.**InputType**(value)

Enumeration of allowable input types to the compiler.

An instance of this enum or the string form can be passed to *CompilerOptions.input\_type*.

Valid values are as follows:

**NONE**

**MHLO**

**TOSA**

**TM\_TENSOR**

**XLA**

**enum** iree.compiler.tools.**OutputFormat**(value)

The output format of the compiler.

Valid values are as follows:

**FLATBUFFER\_BINARY**

**FLATBUFFER\_TEXT**

**MLIR\_TEXT**

## 2.2.2 Debugging

A number of optional arguments to the compiler can be useful for debugging:

- *extended\_diagnostics=True* - Outputs verbose attached operations to diagnostics. Can output a large volume of information.
- *crash\_reproducer\_path=... some .mlir file path...* - On a crash or error, a reproducer will be output at the listed path.
- *extra\_args=[...]* - Passes extra arguments to the compiler. Useful for various standard features of MLIR based compilers like *-mlir-print-ir-after-all*.

In addition, the core compiler and frontend compiler APIs have a unified mechanism for saving their temporary files, which are often useful for post mortem debugging. Since the need for this is often as part of a larger system, it is exposed both via an environment variable and an API.

In order to save all temporaries and reproducers, set the *IREE\_SAVE\_TEMPS* environment variable to a directory in which to dump artifacts. For complex programs that invoke the compiler many times, it will typically be necessary to further qualify the path, and there are a few placeholders that will be expanded:

- *{id}* - A per-process monotonically increasing number for each compiler invocation. Can be overridden by the API if a better symbolic name is available (i.e. test case, etc).
- *{pid}* - Process ID of the current process.
- *{main}* - Basename of *sys.argv[0]*, which is typically the name of the Python main file.

For interactive use, the following (on a Unix-like system) should provide value:

```
export IREE_SAVE_TEMPS="/tmp/ireedumps/{main}/{id}"
```

For the context manager based API, refer to the *iree.compiler.tools.debugging.TempFileSaver* class.

```
class iree.compiler.tools.debugging.TempFileSaver(temp_path_pattern: Optional[str] = None, *,
                                                invocation_id: Optional[str] = None)
```

Manages the saving of temp files resulting from tool invocations.

The TempFileSaver is a thread-local context bound object. An attempt to create a new one will return the most recent instance created and entered as a context manager. This allows up-stack callers to establish the policy for saving temporaries and deep implementations will inherit it.

Proper usage from users wishing to establish a saver context:

```
with TempFileSaver():
    # Do things with temp files.
```

Proper usage for implementors wishing to use an established saver context or set up a new one:

```
with TempFileSaver.implicit() as tfs:
    # Do things with temp files.
```

The outer-most creator can customize it with explicit arguments to `__init__` but these will be ignored if an instance is already thread bound.

**TEMP\_PATH\_ENV\_KEY = 'IREE\_SAVE\_TEMPS'**

```
alloc_optional(file_name: str, *, export_as: Optional[str] = None) → Optional[str]
```

Allocates an optional temporary file.

When in non-retained mode, the return value is 'export\_as', meaning that the file is just some user specified output file.

When in retained mode, the output file will be an index-mangled variant of 'file\_name' under the temp\_path. In addition, a mapping will be added so that upon finalization, the file is also exported to 'export\_as' if specified.

Returns None if neither a user-specified 'export\_as' is specified nor in retained mode.

The distinction between retained temporaries and exports is to help in cases for when the caller has requested that an artifact be written to a specific place (i.e. an output file) but for debuggability, we also want to save it as a temporary. In this case, we save it to the temporary location and then conclude by moving artifacts to their final location once the saver goes out of scope.

**static current()**

**static implicit()**

### 2.2.3 TFLite Importer (*iree-import-tflite*)

Using the API below to access *iree-import-tflite* presumes that the tool itself is installed via the appropriate PIP package.

Imports TFLite binaries via the *iree-import-tflite* tool.

`iree.compiler.tools.tflite.compile_file(fb_path: str, **kwargs)`

Compiles a TFLite FlatBuffer file to an IREE binary.

#### Parameters

- **fb\_path** – Path to the FlatBuffer.
- **\*\*kwargs** – Keyword args corresponding to ImportOptions or CompilerOptions.

**Returns** A bytes-like object with the compiled output or None if output\_file= was specified.

`iree.compiler.tools.tflite.compile_str(input_bytes: bytes, **kwargs)`

Compiles in-memory TFLite FlatBuffer to an IREE binary.

#### Parameters

- **input\_bytes** – Flatbuffer content as bytes or IR string.
- **\*\*kwargs** – Keyword args corresponding to ImportOptions or CompilerOptions.

**Returns** A bytes-like object with the compiled output or None if output\_file= was specified.

```
class iree.compiler.tools.tflite.ImportOptions(output_file: Optional[str] = None, target_backends:
    Sequence[str] = (), input_type: Optional[str] = 'tosa',
    output_format: Union[OutputFormat, str] =
    OutputFormat.FLATBUFFER_BINARY, extra_args:
    Sequence[str] = (), optimize: bool = True,
    output_mlir_debuginfo: bool = True,
    output_generic_mlir: bool = False,
    extended_diagnostics: bool = False, strip_debug_ops:
    bool = False, strip_source_map: bool = False,
    crash_reproducer_path: Optional[str] = None,
    enable_tflite_bindings: bool = False,
    enable_benchmark: bool = False, input_arrays:
    Sequence[str] = (), output_arrays: Sequence[str] = (),
    import_only: bool = False, import_extra_args:
    Sequence[str] = (), save_temp_tfl_input: Optional[str]
    = None, save_temp_iree_input: Optional[str] = None)
```

Import options layer on top of the backend compiler options.

#### Parameters

- **input\_arrays** – Sequence of input array node names (if different from default).
- **output\_arrays** – Sequence of output array node names (if different from default).
- **import\_only** – Only import the module. If True, the result will be textual MLIR that can be further fed to the IREE compiler. If False (default), the result will be the fully compiled IREE binary. In both cases, bytes-like output is returned. Note that if the output\_file= is specified and import\_only=True, then the MLIR form will be written to the output file.
- **import\_extra\_args** – Extra arguments to pass to the iree-import-tf tool.
- **save\_temp\_tfl\_input** – Optionally save the IR that results from importing the flatbuffer (prior to any further transformations).

- **save\_temp\_iree\_input** – Optionally save the IR that is the result of the import (ready to be passed to IREE).

## 2.2.4 TensorFlow Importer (*iree-import-tf*)

Using the API below to access *iree-import-tf* presumes that the tool itself is installed via the appropriate PIP package.

Imports TensorFlow artifacts via the `iree-import-tf` tool.

`iree.compiler.tools.tf.compile_module(module, saved_model_dir: Optional[str] = None, **kwargs)`  
Compiles a `tf.Module` to an IREE binary (by saving to disk).

### Parameters

- **module** – The `tf.Module` instance to convert to MLIR
- **saved\_model\_dir** – Optional path to save the `tf.Module` to. The module will not be persisted on disk outside of this call if this is not provided.
- **\*\*kwargs** – Keyword args corresponding to `ImportOptions` or `CompilerOptions`.

**Returns** Same as `compile_saved_model()`.

`iree.compiler.tools.tf.compile_saved_model(saved_model_dir: str, **kwargs)`  
Compiles an on-disk saved model to an IREE binary.

### Parameters

- **saved\_model\_dir** – Path to directory where the model was saved.
- **\*\*kwargs** – Keyword args corresponding to `ImportOptions` or `CompilerOptions`.

**Returns** A bytes-like object with the compiled output or `None` if `output_file=` was specified.

```
class iree.compiler.tools.tf.ImportOptions(output_file: Optional[str] = None, target_backends:
    Sequence[str] = (), input_type: Union[InputType, str] =
    InputType.XLA, output_format: Union[OutputFormat, str] =
    OutputFormat.FLATBUFFER_BINARY, extra_args:
    Sequence[str] = (), optimize: bool = True,
    output_mlir_debuginfo: bool = True, output_generic_mlir:
    bool = False, extended_diagnostics: bool = False,
    strip_debug_ops: bool = False, strip_source_map: bool =
    False, crash_reproducer_path: Optional[str] = None,
    enable_tflite_bindings: bool = False, enable_benchmark:
    bool = False, exported_names: Sequence[str] = (),
    import_only: bool = False, import_type: Union[ImportType,
    str] = ImportType.OBJECT_GRAPH, saved_model_tags:
    Set[str] = <factory>, import_extra_args: Sequence[str] =
    (), save_temp_tf_input: Optional[str] = None,
    save_temp_mid_level_input: Optional[str] = None,
    save_temp_iree_input: Optional[str] = None, use_tosa: bool
    = False)
```

Import options layer on top of the backend compiler options.

### Parameters

- **exported\_names** – Optional sequence representing the exported names to keep (object graph/v2 models only).
- **import\_only** – Only import the module. If `True`, the result will be textual MLIR that can be further fed to the IREE compiler. If `False` (default), the result will be the fully compiled

IREE binary. In both cases, bytes-like output is returned. Note that if the `output_file=` is specified and `import_only=True`, then the MLIR form will be written to the output file.

- **import\_type** – Type of import to perform. See `ImportType` enum.
- **saved\_model\_tags** – Set of tags to export (signature def/v1 saved models only).
- **import\_extra\_args** – Extra arguments to pass to the `iree-import-tf` tool.
- **save\_temp\_tf\_input** – Optionally save the IR that is input to the TensorFlow pipeline.
- **save\_temp\_mid\_level\_input** – Optionally save the IR that is input to the mid level IR.
- **save\_temp\_iree\_input** – Optionally save the IR that is the result of the import (ready to be passed to IREE).

```
enum iree.compiler.tools.tf.ImportType(value)
    Import type of the model.

    Valid values are as follows:

    OBJECT_GRAPH
    SIGNATURE_DEF
```

## 2.3 IREE Dialect

This dialect contains frontend-oriented ops and types which are intended to be used as input to IREE’s compiler (in addition to various MLIR core dialects).

## 2.4 IREE Python DataModel Dialect

This dialect defines a data model for representing a Python program, suitable for compilation to IREE and for the construction of Python-based DSLs.

### 2.4.1 Importer

### 2.4.2 Operations and Types

## 2.5 MLIR Core Dialects

### 2.5.1 *builtin* dialect

```
class iree.compiler.dialects.builtin.ModuleOp(*, loc=None, ip=None)
class iree.compiler.dialects.builtin.UnrealizedConversionCastOp(outputs, inputs, *, loc=None,
                                                                ip=None)

    OPERATION_NAME = 'builtin.unrealized_conversion_cast'
    property inputs
    property outputs
```

## 2.5.2 *linalg* dialect

```
class iree.compiler.dialects.linalg.AffineBuildState(*, global_state: Op-  
                                                    tional[iree.compiler.dialects.linalg.opdsl.lang.affine.AffineBuildSt  
                                                    = None, allow_new_symbols: bool = True,  
                                                    allow_new_dims: bool = True)
```

Internal state for the AffineExprDef.\_create impls.

Note that a “local” AffineBuildState can be created relative to a “global” AffineBuildState. In that case, any affine expressions built will inherit symbol and dim bindings from the global state and will update both as new ones are discovered. This allows for building expressions across contexts which share a common symbol and dim space.

```
property dim_count: int
```

```
get_dim(dimname: str) → int
```

Gets the dim position given a name.

```
get_symbol(symname: str) → int
```

Get a symbol position given a name.

```
property local_dim_count: int
```

```
property local_symbol_count: int
```

```
property symbol_count: int
```

```
class iree.compiler.dialects.linalg.AffineExprDef
```

Base class for an affine expression being defined.

```
build(state: Optional[iree.compiler.dialects.linalg.opdsl.lang.affine.AffineBuildState] = None) →  
       iree.compiler._mlir_libs._mlir.ir.AffineExpr
```

Builds the corresponding \_ir.AffineExpr from the definitions.

```
static coerce_from(py_value)
```

```
visit_affine_exprs(callback)
```

Visits all AffineExprDefs including self.

```
class iree.compiler.dialects.linalg.BatchMatmulOp(inputs, outputs=(), results=(), loc=None, ip=None)
```

```
class iree.compiler.dialects.linalg.BatchMatmulTransposeBOp(inputs, outputs=(), results=(),  
                                                             loc=None, ip=None)
```

```
class iree.compiler.dialects.linalg.BatchMatvecOp(inputs, outputs=(), results=(), loc=None, ip=None)
```

```
class iree.compiler.dialects.linalg.BatchReduceMatmulOp(inputs, outputs=(), results=(), loc=None,  
                                                         ip=None)
```

```
class iree.compiler.dialects.linalg.BinaryFn
```

Binary function namespace.

As the integer types are signless, signedness is implement by different functions that treat integers as signed or unsigned values.

Examples: - max -> *arith.MaxSIOp* - max\_unsigned -> *arith.MaxUIOp*

```
add = add
```

```
max_signed = max_signed
```

```
max_unsigned = max_unsigned
```

```
min_signed = min_signed
```

```

    min_unsigned = min_unsigned
    mul = mul
    sub = sub

class iree.compiler.dialects.linalg.BinaryFnAttrDef(default:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.BinaryFnTy
    Binary function attribute definition.

    Binary function attributes provide a way to make the arithmetic computation parametrizable. Every attribute
    specifies a default binary function that may be overwritten at operation instantiation time.

class iree.compiler.dialects.linalg.BinaryFnType(fn_name: str)
    Binary function.

    A binary function takes two tensor expressions and returns the function evaluation result.

class iree.compiler.dialects.linalg.BroadcastOp

    OPERATION_NAME = 'linalg.broadcast'
    property init
    property input
    property region
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

class iree.compiler.dialects.linalg.Comprehension(*bindings: Tu-
    ple[iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorUse
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorExpress

    Represents a single comprehension.

    property all_reduction_dims:
    Set[Tuple[iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef, ...]]
        Gets the reduction dims for the comprehension or None.

class iree.compiler.dialects.linalg.Conv1DNcwFcwOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv1DNwcWcfOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv1DOp(inputs, outputs=(), results=(), loc=None, ip=None)

class iree.compiler.dialects.linalg.Conv2DNchwFchwOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv2DNgchwFgchwOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv2DNhwcFhwcOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv2DNhwcHwcFOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv2DNhwcHwcFQOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.Conv2DOp(inputs, outputs=(), results=(), loc=None, ip=None)

```

```

class iree.compiler.dialects.linalg.Conv3DNdhwcDhwcF0p(inputs, outputs=(), results=(), loc=None,
                                                         ip=None)

class iree.compiler.dialects.linalg.Conv3DNdhwcDhwcFQ0p(inputs, outputs=(), results=(), loc=None,
                                                         ip=None)

class iree.compiler.dialects.linalg.Conv3D0p(inputs, outputs=(), results=(), loc=None, ip=None)

class iree.compiler.dialects.linalg.CopyOp(inputs, outputs=(), results=(), loc=None, ip=None)

class iree.compiler.dialects.linalg.DefinedOpCallable(op_name: str, op_def:
                                                         iree.compiler.dialects.linalg.opdsl.lang.comprehension.LinalgOp
    Callable that wraps any defined op function.

class iree.compiler.dialects.linalg.DepthwiseConv1DNwcWcOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv1DNwcWcmOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv2DNchwChwOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv2DNhwcHwcOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv2DNhwcHwcQOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv2DNhwcHwcmOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv2DNhwcHwcmQOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv3DNdhwcDhwcOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DepthwiseConv3DNdhwcDhwcmOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.DimDef(dimname: str)
    Represents a named dimension.

    ALL_DIMS = {'__delete__': Dim(__delete__), '__get__': Dim(__get__),
                '__qualname__': Dim(__qualname__), '__set__': Dim(__set__), '__sphinx_mock__':
                Dim(__sphinx_mock__), 'b': Dim(b), 'c': Dim(c), 'cm': Dim(cm), 'f': Dim(f),
                'fg': Dim(fg), 'g': Dim(g), 'ic': Dim(ic), 'k': Dim(k), 'k0': Dim(k0), 'kd':
                Dim(kd), 'kh': Dim(kh), 'kw': Dim(kw), 'm': Dim(m), 'm0': Dim(m0), 'n': Dim(n),
                'n0': Dim(n0), 'od': Dim(od), 'oh': Dim(oh), 'ow': Dim(ow)}

    classmethod create_expando()
        Create an expando class that creates unique symbols based on attr access.

class iree.compiler.dialects.linalg.DotOp(inputs, outputs=(), results=(), loc=None, ip=None)

class iree.compiler.dialects.linalg.ElementwiseBinaryOp(inputs, outputs=(), results=(), loc=None,
                                                         ip=None)

class iree.compiler.dialects.linalg.ElementwiseUnaryOp(inputs, outputs=(), results=(), loc=None,
                                                         ip=None)

```



**enum** iree.compiler.dialects.linalg.**Enum**(value)

Generic enumeration.

Derive from this class to define new enumerations.

The [Enum](#) and its members have the following methods:

**name**

The name of the Enum member.

**value**

The value of the Enum member.

**class** iree.compiler.dialects.linalg.**FillOp**(inputs, outputs=(), results=(), loc=None, ip=None)

**class** iree.compiler.dialects.linalg.**FillRng2D0p**(inputs, outputs=(), results=(), loc=None, ip=None)

**enum** iree.compiler.dialects.linalg.**FunctionKind**(value)

An enumeration.

Valid values are as follows:

**UNARY**

**BINARY**

**TYPE**

**class** iree.compiler.dialects.linalg.**GenericOp**(result\_tensors, inputs, outputs, indexing\_maps, iterator\_types, \*, doc=None, library\_call=None, loc=None, ip=None)

**OPERATION\_NAME** = 'linalg.generic'

**property** doc

**property** inputs

**property** library\_call

**property** outputs

**property** region

**property** result\_tensors

**class** iree.compiler.dialects.linalg.**IndexAttrDef**(\*sizes: [iree.compiler.dialects.linalg.opdsl.lang.affine.SymbolDef](#), default: [Sequence\[int\]](#))

Index attribute definition.

Index attributes provide a way to define and set symbols that can be used in indexing expressions. Every attribute specifies a tuple of symbols that at compile-time are replaced by integer values as well as their default values.

**class** iree.compiler.dialects.linalg.**IndexOp**(dim, \*, loc=None, ip=None)

**OPERATION\_NAME** = 'linalg.index'

**property** dim

**property** result

Shortcut to get an op result if it has only one (throws an error otherwise).

```
class iree.compiler.dialects.linalg.LinalgOpConfig(metadata:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.OpMetadataL
    *, structured_op: Op-
    tional[iree.compiler.dialects.linalg.opdsl.lang.config.LinalgStructure
    = None)
```

Container for any supported linalg op type.

This includes the concrete type by name for ease of parsing by systems that ignore tags.

```
static from_linalg_op_def(op_def:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.LinalgOpDef,
    context:
    Optional[iree.compiler._mlir_libs._site_initialize.<locals>.Context] =
    None) →
    Sequence[iree.compiler.dialects.linalg.opdsl.lang.config.LinalgOpConfig]
```

Expands a LinalgOpDef into corresponding Linalg configured ops.

```
to_yaml_custom_dict()
```

```
yaml_tag = '!LinalgOpConfig'
```

```
class iree.compiler.dialects.linalg.LinalgOpDef(name: str, cpp_class_name: Optional[str] = None,
    doc: Optional[str] = None)
```

Definition of a linalg op.

```
add_operand(name: str, operand: iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandDef)
    Registers an operand.
```

```
class iree.compiler.dialects.linalg.LinalgStructuredOpConfig(comprehension:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.
    domain: Se-
    quence[iree.compiler.dialects.linalg.opdsl.lang.affine.D
    registered_operands: Se-
    quence[iree.compiler.dialects.linalg.opdsl.lang.compre
    context: Op-
    tional[iree.compiler._mlir_libs._site_initialize.<locals>
    = None)
```

Configuration for metadata sufficient to construct a linalg named op.

```
add_indexed_operand(operand_def: iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandDef)
```

```
add_operand(operand_def: iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandDef)
```

```
add_tensor_use(tensor_use: iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorUse)
```

```
property indexing_maps: Sequence[iree.compiler._mlir_libs._mlir.ir.AffineMap]
```

```
property iterator_types: Sequence[str]
```

```
property ordered_dims: Sequence[Tuple[str, int]]
```

Gets the ordered list of dim bindings (symbolic name, position).

TODO: The original parser relies on parse ordering to arrive at the iterator types, but that ordering is not defined on the Python side, so this may be ambiguous.

```
property ordered_operands:
```

```
Sequence[iree.compiler.dialects.linalg.opdsl.lang.config.OperandDefConfig]
```

```
to_yaml_custom_dict()
```

```
yaml_tag = '!LinalgStructuredOpConfig'
```

```

class iree.compiler.dialects.linalg.MapOp(result, inputs, init, *, loc=None, ip=None)

    OPERATION_NAME = 'linalg.map'
    property init
    property inputs
    property mapper
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.linalg.MatmulOp(inputs, outputs=(), results=(), loc=None, ip=None)
class iree.compiler.dialects.linalg.MatmulTransposeBOp(inputs, outputs=(), results=(), loc=None,
    ip=None)
class iree.compiler.dialects.linalg.MatmulUnsignedOp(inputs, outputs=(), results=(), loc=None,
    ip=None)
class iree.compiler.dialects.linalg.MatvecOp(inputs, outputs=(), results=(), loc=None, ip=None)
class iree.compiler.dialects.linalg.Mmt4DOp(inputs, outputs=(), results=(), loc=None, ip=None)
class iree.compiler.dialects.linalg.OpDefinitionDef(def_name: str)
    A method that an op implements.
class iree.compiler.dialects.linalg.OpInterfaceDef(cpp_name: str)
    An interface that an op implements.
class iree.compiler.dialects.linalg.OpMetadataDef(name: str, cpp_class_name: Optional[str], doc:
    Optional[str])
    Metadata about the op (generally not behavior impacting).
    to_yaml_custom_dict()
    yaml_tag = '!LinalgOpMetadata'
class iree.compiler.dialects.linalg.OperandDef(kind:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandKind,
    type_var: Op-
    tional[iree.compiler.dialects.linalg.opdsl.lang.types.TypeVar]
    = None, size_exprs: Op-
    tional[Sequence[iree.compiler.dialects.linalg.opdsl.lang.affine.AffineExpr]]
    = None, index_dims: Op-
    tional[Sequence[iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef]]
    = None, default_indices: Optional[Sequence[int]] =
    None, default_fn: Optional[str] = None)
    Definition of an operand passed to an operation.
    Keep the meta information of Tensor, Scalar, and Attribute operands and provide the shared registration func-
    tionality.
    attach(index: int, name: str, owner: iree.compiler.dialects.linalg.opdsl.lang.comprehension.LinalgOpDef)
    is_attribute() → bool
    is_input() → bool
    is_tensor() → bool
    
```

```

class iree.compiler.dialects.linalg.OperandDefConfig(operand_def:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandDefConfig,
    shape_map: Optional[iree.compiler._mlir_libs._mlir.ir.AffineMap] = None,
    index_attr_map: Optional[iree.compiler._mlir_libs._mlir.ir.AffineMap] = None)
    Wrapper containing an operand definition with additional state.

    property kind: iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandKind
    property name: str
    to_yaml_custom_dict()
    property type_var: iree.compiler.dialects.linalg.opdsl.lang.types.TypeVar
    yaml_tag = '!LinalgOperandDefConfig'

enum iree.compiler.dialects.linalg.OperandKind(value)
    An enumeration.

    Valid values are as follows:

    INPUT_TENSOR
    SCALAR
    OUTPUT_TENSOR
    INDEX_ATTR
    UNARY_FN_ATTR
    BINARY_FN_ATTR
    TYPE_FN_ATTR

class iree.compiler.dialects.linalg.PoolingNchwMaxOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNchwSumOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNcwMaxOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNcwSumOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNdhwMaxOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNdhwMinOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNdhwSumOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNhwcMaxOp(inputs, outputs=(), results=(), loc=None,
    ip=None)

class iree.compiler.dialects.linalg.PoolingNhwcMaxUnsignedOp(inputs, outputs=(), results=(),
    loc=None, ip=None)

```

```

class iree.compiler.dialects.linalg.PoolingNhwcMinOp(inputs, outputs=(), results=(), loc=None,
                                                    ip=None)

class iree.compiler.dialects.linalg.PoolingNhwcMinUnsignedOp(inputs, outputs=(), results=(),
                                                             loc=None, ip=None)

class iree.compiler.dialects.linalg.PoolingNhwcSumOp(inputs, outputs=(), results=(), loc=None,
                                                      ip=None)

class iree.compiler.dialects.linalg.PoolingNwcMaxOp(inputs, outputs=(), results=(), loc=None,
                                                     ip=None)

class iree.compiler.dialects.linalg.PoolingNwcMaxUnsignedOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.PoolingNwcMinOp(inputs, outputs=(), results=(), loc=None,
                                                     ip=None)

class iree.compiler.dialects.linalg.PoolingNwcMinUnsignedOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.PoolingNwcSumOp(inputs, outputs=(), results=(), loc=None,
                                                     ip=None)

class iree.compiler.dialects.linalg.QuantizedBatchMatmulOp(inputs, outputs=(), results=(),
                                                            loc=None, ip=None)

class iree.compiler.dialects.linalg.QuantizedMatmulOp(inputs, outputs=(), results=(), loc=None,
                                                       ip=None)

class iree.compiler.dialects.linalg.ReduceFn

    add = reduce_add
    max_signed = reduce_max_signed
    max_unsigned = reduce_max_unsigned
    min_signed = reduce_min_signed
    min_unsigned = reduce_min_unsigned
    mul = reduce_mul

class iree.compiler.dialects.linalg.ReduceFnType(binary_fn:
                                                  iree.compiler.dialects.linalg.opdsl.lang.comprehension.BinaryFnType)
    Reduction function.
    A binary function that reduces its RHS into its LHS.

class iree.compiler.dialects.linalg.ReduceFnUse(binary_fn: Op-
                                                  tional[iree.compiler.dialects.linalg.opdsl.lang.comprehension.BinaryFnT
                                                  binary_attr: Op-
                                                  tional[iree.compiler.dialects.linalg.opdsl.lang.comprehension.BinaryFnA
                                                  *reduce_dims:
                                                  iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef)
    Reduction function use.
    A reduction use specifies the reduction function and dimensions.

class iree.compiler.dialects.linalg.ReduceOp(result, inputs, inits, dimensions, *, loc=None, ip=None)

    OPERATION_NAME = 'linalg.reduce'

```

**property combiner**

**property inits**

**property inputs**

**class** iree.compiler.dialects.linalg.**ScalarArg**(arg: *str*)

A type of ScalarExpression that references a named argument.

**expr()** → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**class** iree.compiler.dialects.linalg.**ScalarAssign**(arg: *str*, value:

*iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*)

An assignment to a named argument (LHS of a comprehension).

**to\_yaml\_custom\_dict()**

**yaml\_tag** = '!ScalarAssign'

**class** iree.compiler.dialects.linalg.**ScalarConst**(value: *str*)

A type of ScalarExpression representing a constant.

**expr()** → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**class** iree.compiler.dialects.linalg.**ScalarDef**(type\_var:

*iree.compiler.dialects.linalg.opdsl.lang.types.TypeVar*)

Scalar operand definition.

Scalar operands are forwarded to the body of the structured op as they are. A unique name identifies the scalars and an index determines their position in the operation's parameter list.

**property scalar\_name:** *str*

**to\_scalar\_expression()** → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**class** iree.compiler.dialects.linalg.**ScalarExpression**(scalar\_fn: *Op-*

*tional*[*iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarFn*]

*= None*, scalar\_arg: *Op-*

*tional*[*iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarArg*]

*= None*, scalar\_const: *Op-*

*tional*[*iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarConst*]

*= None*, scalar\_index: *Op-*

*tional*[*iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarIndex*]

*= None*)

An expression on scalar values.

**Can be one of:**

- ScalarFn
- ScalarArg
- ScalarConst
- ScalarIndex

**to\_yaml\_custom\_dict()**

**yaml\_tag** = '!ScalarExpression'

**class** iree.compiler.dialects.linalg.**ScalarFn**(kind: *FunctionKind*, fn\_name: *Optional*[*str*], attr\_name:

*Optional*[*str*], type\_var: *Optional*[*TypeVar*], operands:

*Sequence*[*ScalarExpression*])

A type of ScalarExpression that applies a function.

`expr()` → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.SymbolExpression*

**class** `iree.compiler.dialects.linalg.ScalarIndex(dim: int)`

A type of `ScalarExpression` accessing an iteration index.

`expr()` → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.SymbolExpression*

**class** `iree.compiler.dialects.linalg.SymbolDef(symname: str)`

Represents a named symbol.

```
>>> s1 = SymbolDef("s1")
>>> s1
Symbol(s1)
>>> s2 = SymbolDef("s2")
>>> s1 is s2
False
>>> s1 is SymbolDef("s1")
True
```

```
ALL_SYMBOLS = {'Batch': Symbol(Batch), 'C': Symbol(C), 'CM': Symbol(CM), 'DD':
Symbol(DD), 'DH': Symbol(DH), 'DW': Symbol(DW), 'F': Symbol(F), 'FG': Symbol(FG),
'G': Symbol(G), 'IC': Symbol(IC), 'K': Symbol(K), 'K0': Symbol(K0), 'KD':
Symbol(KD), 'KH': Symbol(KH), 'KW': Symbol(KW), 'M': Symbol(M), 'M0': Symbol(M0),
'N': Symbol(N), 'N0': Symbol(N0), 'OD': Symbol(OD), 'OH': Symbol(OH), 'OW':
Symbol(OW), 'SD': Symbol(SD), 'SH': Symbol(SH), 'SW': Symbol(SW), '__delete__':
Symbol(__delete__), '__get__': Symbol(__get__), '__qualname__':
Symbol(__qualname__), '__set__': Symbol(__set__), '__sphinx_mock__':
Symbol(__sphinx_mock__)}
```

**classmethod** `create_expando()`

Create an expando class that creates unique symbols based on attr access.

**class** `iree.compiler.dialects.linalg.TensorDef(type_var:`

*iree.compiler.dialects.linalg.opdsl.lang.types.TypeVar,*

*\*shape:*

*iree.compiler.dialects.linalg.opdsl.lang.affine.AffineExprDef,*

*index\_dims: Op-*

*tional[Sequence[iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef]]*

*= None, output: bool = False)*

Tensor operand definition.

Tensor operands are indexed using the associated `indexing_map` when forwarded to the body of the structured op. A unique name identifies the tensor operands and an index determines their position in the operation's parameter list. A tensor definition takes type, a shape, and an optional flag to mark output tensors. Additionally, a tuple of index dimensions may be used to map the tensor to the loop dimensions of the operation. This mapping is needed to compute the indexing map of shape-only tensors that have no uses.

**class** `iree.compiler.dialects.linalg.TensorExpression`

An expression that can appear on the RHS of a comprehension.

**collect\_dim\_uses**(*uses: Set[iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef]*)

Collects all `DimDefs` reachable through this expression.

**collect\_indices**(*indices: Set[iree.compiler.dialects.linalg.opdsl.lang.comprehension.index]*)

Collects all index accesses reachable through this expression.

**collect\_scalar\_uses**(*uses: Set[iree.compiler.dialects.linalg.opdsl.lang.comprehension.ScalarDef]*)

Collects all `ScalarDefs` reachable through this expression.

**collect\_tensor\_uses**(*uses: Set[iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorUse]*)

Collects all TensorUses reachable through this expression.

**to\_scalar\_expression**() → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**visit\_tensor\_exprs**(*callback:*

*Callable[[iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorExpression],*  
*None]*)

Visits all tensor expression reachable by the expression.

**class** *iree.compiler.dialects.linalg.TensorFn*(*kind:*

*iree.compiler.dialects.linalg.opdsl.lang.comprehension.FunctionKind,*  
*name: Optional[str], operand\_def: Op-*  
*tional[iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandDef],*  
*type\_var: Op-*  
*tional[iree.compiler.dialects.linalg.opdsl.lang.types.TypeVar],*  
*args: Se-*  
*quence[iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorExpres*

Application of a tensor function.

**to\_scalar\_expression**() → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**visit\_tensor\_exprs**(*callback:*

*Callable[[iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorExpression],*  
*None]*)

Visits all tensor expression reachable by the expression.

**class** *iree.compiler.dialects.linalg.TensorReduceFn*(*reduce\_use:*

*iree.compiler.dialects.linalg.opdsl.lang.comprehension.ReduceFnUse*  
*args: Se-*  
*quence[iree.compiler.dialects.linalg.opdsl.lang.comprehension.Tenso*

Application of a reduction function.

This captures the lhs (initial value) separately from the rhs.

**to\_scalar\_expression**() → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**visit\_tensor\_exprs**(*callback:*

*Callable[[iree.compiler.dialects.linalg.opdsl.lang.comprehension.TensorExpression],*  
*None]*)

Visits all tensor expression reachable by the expression.

**class** *iree.compiler.dialects.linalg.TensorUse*(*operand\_def:*

*iree.compiler.dialects.linalg.opdsl.lang.comprehension.OperandDef,*  
*indices: Se-*  
*quence[iree.compiler.dialects.linalg.opdsl.lang.affine.AffineExprDef])*

A used tensor represented by its (tensor\_name, indices).

Note that forming a comprehension via direct assignment is performed through `__setitem__` on the `TensorDef` level. However, performing a reduction with compound ops (`+=`, `*=`, etc) is done by doing a:

`TensorDef.__getitem__ TensorUse.__iadd__ TensorDef.__setitem__`

**property** *tensor\_name: str*

**to\_scalar\_expression**() → *iree.compiler.dialects.linalg.opdsl.lang.scalar\_expr.ScalarExpression*

**class** *iree.compiler.dialects.linalg.TransposeOp*

**OPERATION\_NAME** = 'linalg.transpose'



**property init**

**property input**

**property region**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**class** iree.compiler.dialects.linalg.**TypeFn**

Type conversion function namespace.

As the integer types are signless, signedness is implement by different cast functions that treat integers as signed (*cast\_signed*) or unsigned (*cast\_unsigned*) values.

Examples: - *cast\_signed*(I32 -> I64) -> *arith.ExtSIOp* - *cast\_unsigned*(I32 -> I64) -> *arith.ExtUIOp*

**cast\_signed** = **cast\_signed**

**cast\_unsigned** = **cast\_unsigned**

**class** iree.compiler.dialects.linalg.**TypeFnAttrDef**(*default:*

iree.compiler.dialects.linalg.opdsl.lang.comprehension.TypeFnType)

Type conversion function attribute definition.

Type conversion function attributes provide a way to make type conversions parameterizable. Every attribute specifies a default type conversion function that may be overwritten at operation instantiation time.

**class** iree.compiler.dialects.linalg.**TypeFnType**(*fn\_name: str*)

Type conversion function.

A type conversion function takes a target type and a tensor expression and returns the casted tensor expression.

**class** iree.compiler.dialects.linalg.**TypeVar**(*name: str*)

A replaceable type variable.

Type variables are unique by name.

```
ALL_TYPEVARS = {'AccumType': TypeVar(AccumType), 'F32': TypeVar(F32), 'F64':
TypeVar(F64), 'I32': TypeVar(I32), 'I64': TypeVar(I64), 'LhsType':
TypeVar(LhsType), 'RhsType': TypeVar(RhsType), 'T': TypeVar(T), 'T1': TypeVar(T1),
'T2': TypeVar(T2), 'U': TypeVar(U), 'V': TypeVar(V), '__delete__':
TypeVar(__delete__), '__get__': TypeVar(__get__), '__qualname__':
TypeVar(__qualname__), '__set__': TypeVar(__set__), '__sphinx_mock__':
TypeVar(__sphinx_mock__)}
```

**classmethod** **create\_expando**()

Create an expando class that creates unique type vars on attr access.

**class** iree.compiler.dialects.linalg.**UnaryFn**

Unary function namespace.

**abs** = **abs**

**ceil** = **ceil**

**exp** = **exp**

**floor** = **floor**

**log** = **log**

**negf** = **negf**

**class** iree.compiler.dialects.linalg.**UnaryFnAttrDef**(*default:*  
[iree.compiler.dialects.linalg.opdsl.lang.comprehension.UnaryFnType](#))

Unary function attribute definition.

Unary function attributes provide a way to make the arithmetic computation parametrizable. Every attribute specifies a default unary function that may be overwritten at operation instantiation time.

**class** iree.compiler.dialects.linalg.**UnaryFnType**(*fn\_name: str*)  
 Unary function.

A unary function takes one tensor expression and returns the function evaluation result.

**class** iree.compiler.dialects.linalg.**VecmatOp**(*inputs, outputs=(), results=(), loc=None, ip=None*)

**class** iree.compiler.dialects.linalg.**YAMLObject**

**as\_linalg\_yaml**()

**classmethod to\_yaml**(*dumper, self*)  
 Default to a custom dictionary mapping.

**to\_yaml\_custom\_dict**()

**class** iree.compiler.dialects.linalg.**YieldOp**(*values, \*, loc=None, ip=None*)

**OPERATION\_NAME** = 'linalg.yield'

**property values**

iree.compiler.dialects.linalg.**bind\_op\_def**(*op\_def:*  
[iree.compiler.dialects.linalg.opdsl.lang.comprehension.LinalgOpDef](#))

**class** iree.compiler.dialects.linalg.**const**(*value: Any*)  
 Returns the given constant floating point or integer value.

**to\_scalar\_expression**() → [iree.compiler.dialects.linalg.opdsl.lang.scalar\\_expr.ScalarExpression](#)

iree.compiler.dialects.linalg.**contextmanager**(*func*)  
 @contextmanager decorator.

Typical usage:

@contextmanager def some\_generator(<arguments>):

<setup> try:

yield <value>

**finally:** <cleanup>

This makes this:

**with some\_generator(<arguments>) as <variable>:** <body>

equivalent to this:

<setup> try:

<variable> = <value> <body>

**finally:** <cleanup>

```

iree.compiler.dialects.linalg.current_op_def() →
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.LinalgOpDef

iree.compiler.dialects.linalg.defines(*definitions:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.OpDefinitionDef)

iree.compiler.dialects.linalg.domain(*dimensions:
    iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef)

iree.compiler.dialects.linalg.emit_generic_structured_op(op_config:
    iree.compiler.dialects.linalg.opdsl.lang.config.LinalgStructuredOpConfig,
    *ins:
    iree.compiler._mlir_libs._mlir.ir.Value,
    outs:
    Union[Sequence[iree.compiler._mlir_libs._mlir.ir.Value],
    iree.compiler._mlir_libs._mlir.ir.OpResultList],
    **attrs: Sequence[int])

iree.compiler.dialects.linalg.emit_named_structured_op(op_config:
    iree.compiler.dialects.linalg.opdsl.lang.config.LinalgStructuredOpConfig,
    op_name: str, op_class_name: str, *ins:
    iree.compiler._mlir_libs._mlir.ir.Value, outs:
    Union[Sequence[iree.compiler._mlir_libs._mlir.ir.Value],
    iree.compiler._mlir_libs._mlir.ir.OpResultList],
    **attrs: Sequence[int])

iree.compiler.dialects.linalg.fill_builtin_region(op: MlirOperation) → None
    Fill the region for op, which is assumed to be a builtin named Linalg op.

iree.compiler.dialects.linalg.implements(*interfaces:
    iree.compiler.dialects.linalg.opdsl.lang.comprehension.OpInterfaceDef)

class iree.compiler.dialects.linalg.index(dim: iree.compiler.dialects.linalg.opdsl.lang.affine.DimDef)
    Returns the iteration index for a given dimension name.

    Resolves the given dimension name to obtain its position in the iteration domain of the operation.

    resolve_dimension_name(affine_state: iree.compiler.dialects.linalg.opdsl.lang.affine.AffineBuildState)
    to_scalar_expression() → iree.compiler.dialects.linalg.opdsl.lang.scalar_expr.ScalarExpression

iree.compiler.dialects.linalg.linalg_structured_op(dsl_func=None, *, op_name=None,
    op_class_name=None) →
    iree.compiler.dialects.linalg.opdsl.lang.dsl.DefinedOpCallable

iree.compiler.dialects.linalg.yaml_dump(data, sort_keys=False, **kwargs)

iree.compiler.dialects.linalg.yaml_dump_all(data, sort_keys=False, explicit_start=True, **kwargs)

```

### 2.5.3 math dialect

```

class iree.compiler.dialects.math.AbsFOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.absf'

    property operand

    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

```

```
class iree.compiler.dialects.math.AbsIOp(operand, *, loc=None, ip=None)

    OPERATION_NAME = 'math.absi'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.Atan2Op(lhs, rhs, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.atan2'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.math.AtanOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.atan'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.CbrtOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.cbrt'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.CeilOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.ceil'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.CopySignOp(lhs, rhs, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.copysign'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.math.CosOp(operand, *, fastmath=None, loc=None, ip=None)
```

```

OPERATION_NAME = 'math.cos'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.CountLeadingZerosOp(operand, *, loc=None, ip=None)

OPERATION_NAME = 'math.ctlz'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.CountTrailingZerosOp(operand, *, loc=None, ip=None)

OPERATION_NAME = 'math.cttz'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.CtPopOp(operand, *, loc=None, ip=None)

OPERATION_NAME = 'math.ctpop'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.ErfOp(operand, *, fastmath=None, loc=None, ip=None)

OPERATION_NAME = 'math.erf'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.Exp2Op(operand, *, fastmath=None, loc=None, ip=None)

OPERATION_NAME = 'math.exp2'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.ExpM1Op(operand, *, fastmath=None, loc=None, ip=None)

OPERATION_NAME = 'math.expm1'
property operand
property result
    Shortcut to get an op result if it has only one (throws an error otherwise).

```

```
class iree.compiler.dialects.math.ExpOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.exp'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.FPowIOp(lhs, rhs, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.fpowi'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.math.FloorOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.floor'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.FmaOp(a, b, c, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.fma'
    property a
    property b
    property c
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.IPowIOp(lhs, rhs, *, loc=None, ip=None)

    OPERATION_NAME = 'math.ipowi'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.math.Log10Op(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.log10'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```

class iree.compiler.dialects.math.Log1pOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.log1p'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.Log2Op(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.log2'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.LogOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.log'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.PowFOp(lhs, rhs, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.powf'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.math.RoundEvenOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.roundeven'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.RoundOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.round'
    property operand
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.math.RsqrtOp(operand, *, fastmath=None, loc=None, ip=None)

    OPERATION_NAME = 'math.rsqrt'

```

**property operand**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**class** iree.compiler.dialects.math.**SinOp**(operand, \*, fastmath=None, loc=None, ip=None)

**OPERATION\_NAME** = 'math.sin'

**property operand**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**class** iree.compiler.dialects.math.**SqrtOp**(operand, \*, fastmath=None, loc=None, ip=None)

**OPERATION\_NAME** = 'math.sqrt'

**property operand**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**class** iree.compiler.dialects.math.**TanOp**(operand, \*, fastmath=None, loc=None, ip=None)

**OPERATION\_NAME** = 'math.tan'

**property operand**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**class** iree.compiler.dialects.math.**TanhOp**(operand, \*, fastmath=None, loc=None, ip=None)

**OPERATION\_NAME** = 'math.tanh'

**property operand**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**class** iree.compiler.dialects.math.**TruncOp**(operand, \*, fastmath=None, loc=None, ip=None)

**OPERATION\_NAME** = 'math.trunc'

**property operand**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).



## 2.5.4 memref dialect

```
class iree.compiler.dialects.memref.AllocOp(memref, dynamicSizes, symbolOperands, *,
                                           alignment=None, loc=None, ip=None)

    OPERATION_NAME = 'memref.alloc'
    property alignment
    property dynamicSizes
    property memref
    property symbolOperands

class iree.compiler.dialects.memref.AllocaOp(memref, dynamicSizes, symbolOperands, *,
                                              alignment=None, loc=None, ip=None)

    OPERATION_NAME = 'memref.alloca'
    property alignment
    property dynamicSizes
    property memref
    property symbolOperands

class iree.compiler.dialects.memref.AllocaScopeOp(results_, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.alloca_scope'
    property bodyRegion
    property results_

class iree.compiler.dialects.memref.AllocaScopeReturnOp(results_, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.alloca_scope.return'
    property results_

class iree.compiler.dialects.memref.AssumeAlignmentOp(memref, alignment, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.assume_alignment'
    property alignment
    property memref

class iree.compiler.dialects.memref.AtomicRMWOp(kind, value, memref, indices, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.atomic_rmw'
    property indices
    property memref
    property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
    property value
```

```

class iree.compiler.dialects.memref.AtomicYieldOp(result, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.atomic_yield'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.memref.CastOp(dest, source, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.cast'
    property dest
    property source
class iree.compiler.dialects.memref.CollapseShapeOp(result, src, reassociation, *, loc=None,
                                                    ip=None)

    OPERATION_NAME = 'memref.collapse_shape'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property src
class iree.compiler.dialects.memref.CopyOp(source, target, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.copy'
    property source
    property target
class iree.compiler.dialects.memref.DeallocOp(memref, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.dealloc'
    property memref
class iree.compiler.dialects.memref.DimOp(source, index, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.dim'
    property index
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source
class iree.compiler.dialects.memref.DmaStartOp(operands_, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.dma_start'
    property operands_
class iree.compiler.dialects.memref.DmaWaitOp(tagMemRef, tagIndices, numElements, *, loc=None,
                                              ip=None)

```

```

    OPERATION_NAME = 'memref.dma_wait'
    property numElements
    property tagIndices
    property tagMemRef
class iree.compiler.dialects.memref.ExpandShapeOp(result, src, reassociation, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.expand_shape'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property src
class iree.compiler.dialects.memref.ExtractAlignedPointerAsIndexOp(source, *, loc=None,
                                                                    ip=None)

    OPERATION_NAME = 'memref.extract_aligned_pointer_as_index'
    property aligned_pointer
    property source
class iree.compiler.dialects.memref.ExtractStridedMetadataOp(source, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.extract_strided_metadata'
    property base_buffer
    property offset
    property sizes
    property source
    property strides
class iree.compiler.dialects.memref.GenericAtomicRMWOp

    OPERATION_NAME = 'memref.generic_atomic_rmw'
    property atomic_body
    property indices
    property memref
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.memref.GetGlobalOp(result, name, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.get_global'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

```

```

class iree.compiler.dialects.memref.GlobalOp(sym_name, type_, *, sym_visibility=None,
                                             initial_value=None, constant=None, alignment=None,
                                             loc=None, ip=None)

    OPERATION_NAME = 'memref.global'
    property alignment
    property constant
    property initial_value
    property sym_name
    property sym_visibility

class iree.compiler.dialects.memref.LoadOp(memref: Union[iree.compiler._mlir_libs._mlir.ir.Operation,
                                                         iree.compiler._mlir_libs._mlir.ir.OpView,
                                                         iree.compiler._mlir_libs._mlir.ir.Value], indices:
                                             Optional[Union[iree.compiler._mlir_libs._mlir.ir.Operation,
                                                         iree.compiler._mlir_libs._mlir.ir.OpView,
                                                         Sequence[iree.compiler._mlir_libs._mlir.ir.Value]]] = None,
                                             *, loc=None, ip=None)

class iree.compiler.dialects.memref.MemorySpaceCastOp(dest, source, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.memory_space_cast'
    property dest
    property source

class iree.compiler.dialects.memref.PrefetchOp(memref, indices, isWrite, localityHint, isDataCache, *,
                                                loc=None, ip=None)

    OPERATION_NAME = 'memref.prefetch'
    property indices
    property isDataCache
    property isWrite
    property localityHint
    property memref

class iree.compiler.dialects.memref.RankOp(memref, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.rank'
    property memref

class iree.compiler.dialects.memref.ReallocOp(result, source, *, dynamicResultSize=None,
                                              alignment=None, loc=None, ip=None)

    OPERATION_NAME = 'memref.realloc'
    property alignment
    property dynamicResultSize

```

**property source**

```
class iree.compiler.dialects.memref.ReinterpretCastOp(result, source, offsets, sizes, strides,
                                                    static_offsets, static_sizes, static_strides, *,
                                                    loc=None, ip=None)
```

**OPERATION\_NAME = 'memref.reinterpret\_cast'**

**property offsets**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**property sizes**

**property source**

**property strides**

```
class iree.compiler.dialects.memref.ReshapeOp(result, source, shape, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'memref.reshape'**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**property shape**

**property source**

```
class iree.compiler.dialects.memref.StoreOp(value, memref, indices, *, nontemporal=None, loc=None,
                                             ip=None)
```

**OPERATION\_NAME = 'memref.store'**

**property indices**

**property memref**

**property nontemporal**

**property value**

```
class iree.compiler.dialects.memref.SubViewOp(result, source, offsets, sizes, strides, static_offsets,
                                              static_sizes, static_strides, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'memref.subview'**

**property offsets**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**property sizes**

**property source**

**property strides**

```
class iree.compiler.dialects.memref.TensorStoreOp(tensor, memref, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'memref.tensor\_store'**

```

    property memref
    property tensor
class iree.compiler.dialects.memref.TransposeOp(result, in_, permutation, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.transpose'
    property in_
class iree.compiler.dialects.memref.ViewOp(result, source, byte_shift, sizes, *, loc=None, ip=None)

    OPERATION_NAME = 'memref.view'
    property byte_shift
    property sizes
    property source

```

### 2.5.5 *shape* dialect

```

class iree.compiler.dialects.shape.AddOp(lhs, rhs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.add'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.shape.AnyOp(result, inputs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.any'
    property inputs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.shape.AssumingAllOp(inputs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.assuming_all'
    property inputs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.shape.AssumingOp(results_, witness, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.assuming'
    property doRegion
    property results_
    property witness

```

```

class iree.compiler.dialects.shape.AssumingYieldOp(operands_, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.assuming_yield'
    property operands_

class iree.compiler.dialects.shape.BroadcastOp(result, shapes, *, error=None, loc=None, ip=None)

    OPERATION_NAME = 'shape.broadcast'
    property error
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property shapes

class iree.compiler.dialects.shape.ConcatOp(result, lhs, rhs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.concat'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs

class iree.compiler.dialects.shape.ConstShapeOp(shape, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.const_shape'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property shape

class iree.compiler.dialects.shape.ConstSizeOp(value, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.const_size'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property value

class iree.compiler.dialects.shape.ConstWitnessOp(passing, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.const_witness'
    property passing
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

class iree.compiler.dialects.shape.CstrBroadcastableOp(shapes, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.cstr_broadcastable'

```

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**property shapes**

```
class iree.compiler.dialects.shape.CstrEqOp(shapes, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'shape.cstr\_eq'**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**property shapes**

```
class iree.compiler.dialects.shape.CstrRequireOp(pred, msg, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'shape.cstr\_require'**

**property msg**

**property pred**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

```
class iree.compiler.dialects.shape.DebugPrintOp(output, input, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'shape.debug\_print'**

**property input**

**property output**

```
class iree.compiler.dialects.shape.DimOp(value, index, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'shape.dim'**

**property extent**

**property index**

**property value**

```
class iree.compiler.dialects.shape.DivOp(lhs, rhs, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'shape.div'**

**property lhs**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).

**property rhs**

```
class iree.compiler.dialects.shape.FromExtentTensorOp(input, *, loc=None, ip=None)
```

**OPERATION\_NAME = 'shape.from\_extent\_tensor'**

**property input**

**property result**

Shortcut to get an op result if it has only one (throws an error otherwise).



```

class iree.compiler.dialects.shape.FromExtentsOp(extents, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.from_extents'
    property extents
    property shape

class iree.compiler.dialects.shape.FuncOp(sym_name, function_type, *, arg_attrs=None,
                                           res_attrs=None, sym_visibility=None, loc=None, ip=None)

    OPERATION_NAME = 'shape.func'
    property body
    property sym_name
    property sym_visibility

class iree.compiler.dialects.shape.FunctionLibraryOp

    OPERATION_NAME = 'shape.function_library'
    property body
    property sym_name
    property sym_visibility

class iree.compiler.dialects.shape.GetExtentOp(shape, dim, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.get_extent'
    property dim
    property extent
    property shape

class iree.compiler.dialects.shape.IndexToSizeOp(arg, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.index_to_size'
    property arg
    property result
    Shortcut to get an op result if it has only one (throws an error otherwise).

class iree.compiler.dialects.shape.IsBroadcastableOp(shapes, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.is_broadcastable'
    property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
    property shapes

class iree.compiler.dialects.shape.MaxOp(lhs, rhs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.max'

```

```
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.shape.MeetOp(arg0, arg1, *, error=None, loc=None, ip=None)

    OPERATION_NAME = 'shape.meet'
    property arg0
    property arg1
    property error
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.shape.MinOp(lhs, rhs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.min'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.shape.MulOp(lhs, rhs, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.mul'
    property lhs
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property rhs
class iree.compiler.dialects.shape.NumElementsOp(shape, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.num_elements'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property shape
class iree.compiler.dialects.shape.RankOp(shape, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.rank'
    property rank
    property shape
class iree.compiler.dialects.shape.ReduceOp(result, shape, initVals, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.reduce'
```

```

    property initVals
    property region
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property shape
class iree.compiler.dialects.shape.ReturnOp(operands_, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.return'
    property operands_
class iree.compiler.dialects.shape.ShapeEqOp(shapes, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.shape_eq'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property shapes
class iree.compiler.dialects.shape.ShapeOfOp(arg, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.shape_of'
    property arg
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.shape.SizeToIndexOp(arg, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.size_to_index'
    property arg
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.shape.SplitAtOp(head, tail, operand, index, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.split_at'
    property head
    property index
    property operand
    property tail
class iree.compiler.dialects.shape.ToExtentTensorOp(result, input, *, loc=None, ip=None)

    OPERATION_NAME = 'shape.to_extent_tensor'
    property input
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

```

```
class iree.compiler.dialects.shape.ValueAsShapeOp(result, arg, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'shape.value_as_shape'
```

```
    property arg
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
class iree.compiler.dialects.shape.ValueOfOp(result, arg, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'shape.value_of'
```

```
    property arg
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
class iree.compiler.dialects.shape.WithOp(operand, shape, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'shape.with_shape'
```

```
    property operand
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
    property shape
```

```
class iree.compiler.dialects.shape.YieldOp(operands_, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'shape.yield'
```

```
    property operands_
```

## 2.5.6 std dialect

## 2.5.7 tensor dialect

```
class iree.compiler.dialects.tensor.CastOp(dest, source, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tensor.cast'
```

```
    property dest
```

```
    property source
```

```
class iree.compiler.dialects.tensor.CollapseShapeOp(result, src, reassociation, *, loc=None,
                                                    ip=None)
```

```
    OPERATION_NAME = 'tensor.collapse_shape'
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
    property src
```

```

class iree.compiler.dialects.tensor.DimOp(source, index, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.dim'
    property index
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source

class iree.compiler.dialects.tensor.EmptyOp(sizes: Sequence[Union[int,
    iree.compiler._mlir_libs._mlir.ir.Value]], element_type:
    iree.compiler._mlir_libs._mlir.ir.Type, *, loc=None,
    ip=None)

class iree.compiler.dialects.tensor.ExpandShapeOp(result, src, reassociation, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.expand_shape'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property src

class iree.compiler.dialects.tensor.ExtractOp(tensor, indices, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.extract'
    property indices
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property tensor

class iree.compiler.dialects.tensor.ExtractSliceOp(result, source, offsets, sizes, strides, static_offsets,
    static_sizes, static_strides, *, loc=None,
    ip=None)

    OPERATION_NAME = 'tensor.extract_slice'
    property offsets
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property sizes
    property source
    property strides

class iree.compiler.dialects.tensor.FromElementsOp

    OPERATION_NAME = 'tensor.from_elements'
    property elements
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

```

```
class iree.compiler.dialects.tensor.GatherOp(result, source, indices, gather_dims, *, unique=None,
                                             loc=None, ip=None)
```

```
    OPERATION_NAME = 'tensor.gather'
```

```
    property indices
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
    property source
```

```
    property unique
```

```
class iree.compiler.dialects.tensor.GenerateOp(result, dynamicExtents, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tensor.generate'
```

```
    property body
```

```
    property dynamicExtents
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
class iree.compiler.dialects.tensor.InsertOp(scalar, dest, indices, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tensor.insert'
```

```
    property dest
```

```
    property indices
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
    property scalar
```

```
class iree.compiler.dialects.tensor.InsertSliceOp(source, dest, offsets, sizes, strides, static_offsets,
                                                  static_sizes, static_strides, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tensor.insert_slice'
```

```
    property dest
```

```
    property offsets
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
    property sizes
```

```
    property source
```

```
    property strides
```

```
class iree.compiler.dialects.tensor.PackOp(source, dest, inner_dims_pos, inner_tiles, static_inner_tiles,
                                             *, padding_value=None, outer_dims_perm=None,
                                             loc=None, ip=None)
```

```
    OPERATION_NAME = 'tensor.pack'
```

```

    property dest
    property inner_tiles
    property padding_value
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source
class iree.compiler.dialects.tensor.PadOp(result, source, low, high, static_low, static_high, *,
                                          nofold=None, loc=None, ip=None)

    OPERATION_NAME = 'tensor.pad'
    property high
    property low
    property nofold
    property region
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source
class iree.compiler.dialects.tensor.ParallelInsertSliceOp(source, dest, offsets, sizes, strides,
                                                           static_offsets, static_sizes, static_strides,
                                                           *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.parallel_insert_slice'
    property dest
    property offsets
    property sizes
    property source
    property strides
class iree.compiler.dialects.tensor.RankOp(tensor, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.rank'
    property tensor
class iree.compiler.dialects.tensor.ReshapeOp(result, source, shape, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.reshape'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property shape
    property source

```

```

class iree.compiler.dialects.tensor.ScatterOp(result, source, dest, indices, scatter_dims, *,
                                              unique=None, loc=None, ip=None)

    OPERATION_NAME = 'tensor.scatter'
    property dest
    property indices
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source
    property unique

class iree.compiler.dialects.tensor.SplatOp(aggregate, input, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.splat'
    property aggregate
    property input

class iree.compiler.dialects.tensor.UnPackOp(source, dest, inner_dims_pos, inner_tiles,
                                              static_inner_tiles, *, outer_dims_perm=None, loc=None,
                                              ip=None)

    OPERATION_NAME = 'tensor.unpack'
    property dest
    property inner_tiles
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source

class iree.compiler.dialects.tensor.YieldOp(value, *, loc=None, ip=None)

    OPERATION_NAME = 'tensor.yield'
    property value

```

### 2.5.8 tosa dialect

```

class iree.compiler.dialects.tosa.AbsOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.abs'
    property input1
    property output

class iree.compiler.dialects.tosa.AddOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.add'

```



```

    property input1
    property input2
    property output
class iree.compiler.dialects.tosa.ApplyScaleOp(output, value, multiplier, shift, double_round, *,
                                              loc=None, ip=None)

    OPERATION_NAME = 'tosa.apply_scale'
    property double_round
    property multiplier
    property output
    property shift
    property value
class iree.compiler.dialects.tosa.ArgMaxOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.argmax'
    property axis
    property input
    property output
class iree.compiler.dialects.tosa.ArithmeticRightShiftOp(output, input1, input2, round, *,
                                                          loc=None, ip=None)

    OPERATION_NAME = 'tosa.arithmetic_right_shift'
    property input1
    property input2
    property output
    property round
class iree.compiler.dialects.tosa.AvgPool2dOp(output, input, kernel, stride, pad, *,
                                              quantization_info=None, loc=None, ip=None)

    OPERATION_NAME = 'tosa.avg_pool2d'
    property input
    property output
class iree.compiler.dialects.tosa.BitwiseAndOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.bitwise_and'
    property input1
    property input2
    property output

```

```
class iree.compiler.dialects.tosa.BitwiseNotOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.bitwise_not'
    property input1
    property output

class iree.compiler.dialects.tosa.BitwiseOrOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.bitwise_or'
    property input1
    property input2
    property output

class iree.compiler.dialects.tosa.BitwiseXorOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.bitwise_xor'
    property input1
    property input2
    property output

class iree.compiler.dialects.tosa.CastOp(output, input, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.cast'
    property input
    property output

class iree.compiler.dialects.tosa.CeilOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.ceil'
    property input1
    property output

class iree.compiler.dialects.tosa.ClampOp(output, input, min_int, max_int, min_fp, max_fp, *, loc=None,
                                          ip=None)

    OPERATION_NAME = 'tosa.clamp'
    property input
    property max_fp
    property max_int
    property min_fp
    property min_int
    property output

class iree.compiler.dialects.tosa.ClzOp(output, input1, *, loc=None, ip=None)
```

```

    OPERATION_NAME = 'tosa.clz'
    property input1
    property output
class iree.compiler.dialects.tosa.ConcatOp(input1, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.concat'
    property axis
    property input1
    property output
class iree.compiler.dialects.tosa.ConstOp(value, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.const'
    property output
class iree.compiler.dialects.tosa.Conv2D0p(output, input, weight, bias, pad, stride, dilation, *,
                                           quantization_info=None, loc=None, ip=None)

    OPERATION_NAME = 'tosa.conv2d'
    property bias
    property input
    property output
    property weight
class iree.compiler.dialects.tosa.Conv3D0p(output, input, weight, bias, pad, stride, dilation, *,
                                           quantization_info=None, loc=None, ip=None)

    OPERATION_NAME = 'tosa.conv3d'
    property bias
    property input
    property output
    property weight
class iree.compiler.dialects.tosa.CustomOp(outputs, identifier, config, implementation_attrs, inputs, *,
                                           loc=None, ip=None)

    OPERATION_NAME = 'tosa.custom'
    property config
    property identifier
    property implementation_attrs
    property inputs
    property outputs

```

```
class iree.compiler.dialects.tosa.DepthwiseConv2D0p(output, input, weight, bias, pad, stride, dilation,
                                                    *, quantization_info=None, loc=None,
                                                    ip=None)
```

```
    OPERATION_NAME = 'tosa.depthwise_conv2d'
```

```
    property bias
```

```
    property input
```

```
    property output
```

```
    property weight
```

```
class iree.compiler.dialects.tosa.Div0p(output, input1, input2, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.div'
```

```
    property input1
```

```
    property input2
```

```
    property output
```

```
class iree.compiler.dialects.tosa.EqualOp(input1, input2, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.equal'
```

```
    property input1
```

```
    property input2
```

```
    property output
```

```
class iree.compiler.dialects.tosa.Exp0p(output, input1, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.exp'
```

```
    property input1
```

```
    property output
```

```
class iree.compiler.dialects.tosa.FFT2d0p(output_real, output_imag, input_real, input_imag, inverse, *,
                                           loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.fft2d'
```

```
    property input_imag
```

```
    property input_real
```

```
    property inverse
```

```
    property output_imag
```

```
    property output_real
```

```
class iree.compiler.dialects.tosa.FloorOp(output, input1, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.floor'
```

```
    property input1
```

```

    property output

class iree.compiler.dialects.tosa.FullyConnectedOp(output, input, weight, bias, *,
                                                    quantization_info=None, loc=None, ip=None)

    OPERATION_NAME = 'tosa.fully_connected'
    property bias
    property input
    property output
    property weight

class iree.compiler.dialects.tosa.GatherOp(output, values, indices, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.gather'
    property indices
    property output
    property values

class iree.compiler.dialects.tosa.GreaterEqualOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.greater_equal'
    property input1
    property input2
    property output

class iree.compiler.dialects.tosa.GreaterOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.greater'
    property input1
    property input2
    property output

class iree.compiler.dialects.tosa.IdentityOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.identity'
    property input1
    property output

class iree.compiler.dialects.tosa.IfOp(output, cond, inputs, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.cond_if'
    property cond
    property else_branch
    property inputs

```

```
    property output
    property then_branch
class iree.compiler.dialects.tosa.LogOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.log'
    property input1
    property output
class iree.compiler.dialects.tosa.LogicalAndOp(z, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.logical_and'
    property input1
    property input2
    property z
class iree.compiler.dialects.tosa.LogicalLeftShiftOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.logical_left_shift'
    property input1
    property input2
    property output
class iree.compiler.dialects.tosa.LogicalNotOp(input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.logical_not'
    property input1
    property output
class iree.compiler.dialects.tosa.LogicalOrOp(z, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.logical_or'
    property input1
    property input2
    property z
class iree.compiler.dialects.tosa.LogicalRightShiftOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.logical_right_shift'
    property input1
    property input2
    property output
class iree.compiler.dialects.tosa.LogicalXorOp(z, input1, input2, *, loc=None, ip=None)
```

```

    OPERATION_NAME = 'tosa.logical_xor'
    property input1
    property input2
    property z
class iree.compiler.dialects.tosa.MatMulOp(c, a, b, *, quantization_info=None, loc=None, ip=None)

    OPERATION_NAME = 'tosa.matmul'
    property a
    property b
    property c
class iree.compiler.dialects.tosa.MaxPool2dOp(output, input, kernel, stride, pad, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.max_pool2d'
    property input
    property output
class iree.compiler.dialects.tosa.MaximumOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.maximum'
    property input1
    property input2
    property output
class iree.compiler.dialects.tosa.MinimumOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.minimum'
    property input1
    property input2
    property output
class iree.compiler.dialects.tosa.MulOp(output, input1, input2, shift, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.mul'
    property input1
    property input2
    property output
    property shift
class iree.compiler.dialects.tosa.NegateOp(output, input1, *, quantization_info=None, loc=None,
                                           ip=None)

    OPERATION_NAME = 'tosa.negate'

```

```

    property input1
    property output
class iree.compiler.dialects.tosa.PadOp(output, input1, padding, *, pad_const=None,
                                       quantization_info=None, loc=None, ip=None)

    OPERATION_NAME = 'tosa.pad'
    property input1
    property output
    property pad_const
    property padding
class iree.compiler.dialects.tosa.PowOp(z, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.pow'
    property input1
    property input2
    property z
class iree.compiler.dialects.tosa.RFFT2dOp(output_real, output_imag, input, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.rfft2d'
    property input
    property output_imag
    property output_real
class iree.compiler.dialects.tosa.ReciprocalOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reciprocal'
    property input1
    property output
class iree.compiler.dialects.tosa.ReduceAllOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reduce_all'
    property axis
    property input
    property output
class iree.compiler.dialects.tosa.ReduceAnyOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reduce_any'
    property axis
    property input

```



```

    property output

class iree.compiler.dialects.tosa.ReduceMaxOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reduce_max'
    property axis
    property input
    property output

class iree.compiler.dialects.tosa.ReduceMinOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reduce_min'
    property axis
    property input
    property output

class iree.compiler.dialects.tosa.ReduceProdOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reduce_prod'
    property axis
    property input
    property output

class iree.compiler.dialects.tosa.ReduceSumOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reduce_sum'
    property axis
    property input
    property output

class iree.compiler.dialects.tosa.RescaleOp(output, input, input_zp, output_zp, multiplier, shift, scale32,
                                             double_round, per_channel, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.rescale'
    property double_round
    property input
    property input_zp
    property output
    property output_zp
    property per_channel
    property scale32

class iree.compiler.dialects.tosa.ReshapeOp(output, input1, new_shape, *, loc=None, ip=None)

```

```

    OPERATION_NAME = 'tosa.reshape'
    property input1
    property output
class iree.compiler.dialects.tosa.ResizeOp(output, input, scale, offset, border, mode, *, loc=None,
                                           ip=None)

    OPERATION_NAME = 'tosa.resize'
    property input
    property mode
    property output
class iree.compiler.dialects.tosa.ReverseOp(output, input, axis, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.reverse'
    property axis
    property input
    property output
class iree.compiler.dialects.tosa.RsqrtOp(output, input1, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.rsqrt'
    property input1
    property output
class iree.compiler.dialects.tosa.ScatterOp(values_out, values_in, indices, input, *, loc=None,
                                           ip=None)

    OPERATION_NAME = 'tosa.scatter'
    property indices
    property input
    property values_in
    property values_out
class iree.compiler.dialects.tosa.SelectOp(output, pred, on_true, on_false, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.select'
    property on_false
    property on_true
    property output
    property pred
class iree.compiler.dialects.tosa.SigmoidOp(output, input, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.sigmoid'

```

```

    property input
    property output
class iree.compiler.dialects.tosa.SliceOp(output, input, start, size, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.slice'
    property input
    property output
class iree.compiler.dialects.tosa.SubOp(output, input1, input2, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.sub'
    property input1
    property input2
    property output
class iree.compiler.dialects.tosa.TableOp(output, input, table, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.table'
    property input
    property output
    property table
class iree.compiler.dialects.tosa.TanhOp(output, input, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.tanh'
    property input
    property output
class iree.compiler.dialects.tosa.TileOp(output, input1, multiples, *, loc=None, ip=None)

    OPERATION_NAME = 'tosa.tile'
    property input1
    property output
class iree.compiler.dialects.tosa.TransposeConv2DOp(output, input, filter, bias, out_pad, stride,
                                                    out_shape, *, quantization_info=None,
                                                    loc=None, ip=None)

    OPERATION_NAME = 'tosa.transpose_conv2d'
    property bias
    property filter
    property input
    property output

```

```
class iree.compiler.dialects.tosa.TransposeOp(output, input1, perms, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.transpose'
```

```
    property input1
```

```
    property output
```

```
    property perms
```

```
class iree.compiler.dialects.tosa.WhileOp(output, inputs, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.while_loop'
```

```
    property body
```

```
    property cond
```

```
    property inputs
```

```
    property output
```

```
class iree.compiler.dialects.tosa.YieldOp(inputs, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'tosa.yield'
```

```
    property inputs
```

## 2.5.9 vector dialect

```
class iree.compiler.dialects.vector.BitCastOp(result, source, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'vector.bitcast'
```

```
    property result
```

```
        Shortcut to get an op result if it has only one (throws an error otherwise).
```

```
    property source
```

```
class iree.compiler.dialects.vector.BroadcastOp(vector, source, *, loc=None, ip=None)
```

```
    OPERATION_NAME = 'vector.broadcast'
```

```
    property source
```

```
    property vector
```

```
class iree.compiler.dialects.vector.CompressStoreOp(base, indices, mask, valueToStore, *, loc=None,
                                                         ip=None)
```

```
    OPERATION_NAME = 'vector.compressstore'
```

```
    property base
```

```
    property indices
```

```
    property mask
```

```
    property valueToStore
```

```

class iree.compiler.dialects.vector.ConstantMaskOp(result, mask_dim_sizes, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.constant_mask'
class iree.compiler.dialects.vector.ContractionOp(result, lhs, rhs, acc, masks, indexing_maps,
                                                    iterator_types, *, kind=None, loc=None, ip=None)

    OPERATION_NAME = 'vector.contract'
    property acc
    property lhs
    property masks
    property rhs
class iree.compiler.dialects.vector.CreateMaskOp(result, operands_, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.create_mask'
    property operands_
class iree.compiler.dialects.vector.ExpandLoadOp(result, base, indices, mask, pass_thru, *, loc=None,
                                                    ip=None)

    OPERATION_NAME = 'vector.expandload'
    property base
    property indices
    property mask
    property pass_thru
    property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.vector.ExtractElementOp(vector, *, position=None, loc=None, ip=None)

    OPERATION_NAME = 'vector.extractelement'
    property position
    property result
    Shortcut to get an op result if it has only one (throws an error otherwise).
    property vector
class iree.compiler.dialects.vector.ExtractOp(vector, position, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.extract'
    property vector
class iree.compiler.dialects.vector.ExtractStridedSliceOp(result, vector, offsets, sizes, strides, *,
                                                            loc=None, ip=None)

    OPERATION_NAME = 'vector.extract_strided_slice'

```

property vector

```
class iree.compiler.dialects.vector.FMAOp(lhs, rhs, acc, *, loc=None, ip=None)
```

OPERATION\_NAME = 'vector.fma'

property acc

property lhs

property result

Shortcut to get an op result if it has only one (throws an error otherwise).

property rhs

```
class iree.compiler.dialects.vector.FlatTransposeOp(res, matrix, rows, columns, *, loc=None,
                                                    ip=None)
```

OPERATION\_NAME = 'vector.flat\_transpose'

property columns

property matrix

property res

property rows

```
class iree.compiler.dialects.vector.GatherOp(result, base, indices, index_vec, mask, pass_thru, *,
                                             loc=None, ip=None)
```

OPERATION\_NAME = 'vector.gather'

property base

property index\_vec

property indices

property mask

property pass\_thru

property result

Shortcut to get an op result if it has only one (throws an error otherwise).

```
class iree.compiler.dialects.vector.InsertElementOp(source, dest, *, position=None, loc=None,
                                                    ip=None)
```

OPERATION\_NAME = 'vector.insertelement'

property dest

property position

property result

Shortcut to get an op result if it has only one (throws an error otherwise).

property source

```
class iree.compiler.dialects.vector.InsertOp(source, dest, position, *, loc=None, ip=None)
```

OPERATION\_NAME = 'vector.insert'

```

    property dest
    property res
    property source
class iree.compiler.dialects.vector.InsertStridedSliceOp(source, dest, offsets, strides, *, loc=None,
                                                         ip=None)

    OPERATION_NAME = 'vector.insert_strided_slice'
    property dest
    property res
    property source
class iree.compiler.dialects.vector.LoadOp(result, base, indices, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.load'
    property base
    property indices
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.vector.MaskOp

    OPERATION_NAME = 'vector.mask'
    property mask
    property maskRegion
    property passthru
    property results_
class iree.compiler.dialects.vector.MaskedLoadOp(result, base, indices, mask, pass_thru, *, loc=None,
                                                  ip=None)

    OPERATION_NAME = 'vector.maskedload'
    property base
    property indices
    property mask
    property pass_thru
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
class iree.compiler.dialects.vector.MaskedStoreOp(base, indices, mask, valueToStore, *, loc=None,
                                                  ip=None)

    OPERATION_NAME = 'vector.maskedstore'
    property base

```

```
    property indices
    property mask
    property valueToStore
class iree.compiler.dialects.vector.MatmulOp(res, lhs, rhs, lhs_rows, lhs_columns, rhs_columns, *,
                                             loc=None, ip=None)

    OPERATION_NAME = 'vector.matrix_multiply'
    property lhs
    property lhs_columns
    property lhs_rows
    property res
    property rhs
    property rhs_columns
class iree.compiler.dialects.vector.MultiDimReductionOp(kind, source, acc, reduction_dims, *,
                                                         loc=None, ip=None)

    OPERATION_NAME = 'vector.multi_reduction'
    property acc
    property dest
    property source
class iree.compiler.dialects.vector.OuterProductOp(result, lhs, rhs, acc, *, kind=None, loc=None,
                                                    ip=None)

    OPERATION_NAME = 'vector.outerproduct'
    property acc
    property lhs
    property rhs
class iree.compiler.dialects.vector.PrintOp(source, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.print'
    property source
class iree.compiler.dialects.vector.ReductionOp(dest, kind, vector, *, acc=None, loc=None, ip=None)

    OPERATION_NAME = 'vector.reduction'
    property acc
    property dest
    property vector
```



```

class iree.compiler.dialects.vector.ReshapeOp(result, vector, input_shape, output_shape,
                                              fixed_vector_sizes, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.reshape'
    property input_shape
    property output_shape
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property vector

class iree.compiler.dialects.vector.ScalableExtractOp(res, source, pos, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.scalable.extract'
    property pos
    property res
    property source

class iree.compiler.dialects.vector.ScalableInsertOp(source, dest, pos, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.scalable.insert'
    property dest
    property pos
    property res
    property source

class iree.compiler.dialects.vector.ScanOp(kind, source, initial_value, reduction_dim, inclusive, *,
                                           loc=None, ip=None)

    OPERATION_NAME = 'vector.scan'
    property accumulated_value
    property dest
    property inclusive
    property initial_value
    property reduction_dim
    property source

class iree.compiler.dialects.vector.ScatterOp(base, indices, index_vec, mask, valueToStore, *,
                                              loc=None, ip=None)

    OPERATION_NAME = 'vector.scatter'
    property base
    property index_vec
    property indices

```

```
    property mask
    property valueToStore
class iree.compiler.dialects.vector.ShapeCastOp(result, source, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.shape_cast'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source
class iree.compiler.dialects.vector.ShuffleOp(v1, v2, mask, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.shuffle'
    property v1
    property v2
    property vector
class iree.compiler.dialects.vector.SplatOp(aggregate, input, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.splat'
    property aggregate
    property input
class iree.compiler.dialects.vector.StoreOp(valueToStore, base, indices, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.store'
    property base
    property indices
    property valueToStore
class iree.compiler.dialects.vector.TransferReadOp(vector, source, indices, permutation_map,
                                                    padding, *, mask=None, in_bounds=None,
                                                    loc=None, ip=None)

    OPERATION_NAME = 'vector.transfer_read'
    property indices
    property mask
    property padding
    property source
    property vector
class iree.compiler.dialects.vector.TransferWriteOp(result, vector, source, indices, permutation_map,
                                                    *, mask=None, in_bounds=None, loc=None,
                                                    ip=None)

    OPERATION_NAME = 'vector.transfer_write'
```

```

    property indices
    property mask
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property source
    property vector

class iree.compiler.dialects.vector.TransposeOp(result, vector, transp, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.transpose'
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).
    property vector

class iree.compiler.dialects.vector.TypeCastOp(result, memref, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.type_cast'
    property memref
    property result
        Shortcut to get an op result if it has only one (throws an error otherwise).

class iree.compiler.dialects.vector.VectorScaleOp(*, loc=None, ip=None)

    OPERATION_NAME = 'vector.vscale'
    property res

class iree.compiler.dialects.vector.WarpExecuteOnLane0Op

    OPERATION_NAME = 'vector.warp_execute_on_lane_0'
    property args
    property laneid
    property results_
    property warpRegion
    property warp_size

class iree.compiler.dialects.vector.YieldOp(operands_, *, loc=None, ip=None)

    OPERATION_NAME = 'vector.yield'
    property operands_

```

## 2.6 TensorFlow Dialects

IREE’s compiler is capable of ingesting selected TensorFlow dialects directly. Those that are integrated into the in-process compiler API are presented here.

### 2.6.1 *chlo* dialect

### 2.6.2 *mhlo* dialect

## 2.7 Embedded MLIR API

### 2.7.1 *iree.compiler.ir* module

```
class iree.compiler.ir.AffineAddExpr

    static get(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1:
        iree.compiler._mlir_libs._mlir.ir.AffineExpr) → iree.compiler._mlir_libs._mlir.ir.AffineAddExpr
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
class iree.compiler.ir.AffineBinaryExpr

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
    property lhs
    property rhs
class iree.compiler.ir.AffineCeilDivExpr

    static get(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1:
        iree.compiler._mlir_libs._mlir.ir.AffineExpr) →
        iree.compiler._mlir_libs._mlir.ir.AffineCeilDivExpr
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
class iree.compiler.ir.AffineConstantExpr

    static get(value: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.AffineConstantExpr
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
    property value
class iree.compiler.ir.AffineDimExpr

    static get(position: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.AffineDimExpr
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
    property position
```

```
class iree.compiler.ir.AffineExpr
```

```
compose(self: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg0: mlir::python::PyAffineMap) →  
    iree.compiler._mlir_libs._mlir.ir.AffineExpr
```

**property context**

```
dump(self: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → None  
    Dumps a debug representation of the object to stderr.
```

```
static get_add(*args, **kwargs)  
    Overloaded function.
```

1. get\_add(arg0: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr, arg1: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineAddExpr

Gets an affine expression containing a sum of two expressions.

2. get\_add(arg0: int, arg1: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineAddExpr

Gets an affine expression containing a sum of a constant and another expression.

3. get\_add(arg0: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr, arg1: int) -> (anonymous namespace)::PyAffineAddExpr

Gets an affine expression containing a sum of an expression and a constant.

```
static get_ceil_div(*args, **kwargs)  
    Overloaded function.
```

1. get\_ceil\_div(arg0: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr, arg1: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineCeilDivExpr

Gets an affine expression containing the rounded-up result of dividing one expression by another.

2. get\_ceil\_div(arg0: int, arg1: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineCeilDivExpr

Gets a semi-affine expression containing the rounded-up result of dividing a constant by an expression.

3. get\_ceil\_div(arg0: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr, arg1: int) -> (anonymous namespace)::PyAffineCeilDivExpr

Gets an affine expression containing the rounded-up result of dividing an expression by a constant.

```
static get_constant(value: int, context: mlir.ir.Context = None) → (anonymous  
    namespace)::PyAffineConstantExpr
```

Gets a constant affine expression with the given value.

```
static get_dim(position: int, context: mlir.ir.Context = None) → (anonymous  
    namespace)::PyAffineDimExpr
```

Gets an affine expression of a dimension at the given position.

```
static get_floor_div(*args, **kwargs)  
    Overloaded function.
```

1. get\_floor\_div(arg0: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr, arg1: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineFloorDivExpr

Gets an affine expression containing the rounded-down result of dividing one expression by another.

2. get\_floor\_div(arg0: int, arg1: iree.compiler.\_mlir\_libs.\_mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineFloorDivExpr

Gets a semi-affine expression containing the rounded-down result of dividing a constant by an expression.

3. `get_floor_div(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1: int) -> (anonymous namespace)::PyAffineFloorDivExpr`

Gets an affine expression containing the rounded-down result of dividing an expression by a constant.

**static** `get_mod(*args, **kwargs)`

Overloaded function.

1. `get_mod(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1: iree.compiler._mlir_libs._mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineModExpr`

Gets an affine expression containing the modulo of dividing one expression by another.

2. `get_mod(arg0: int, arg1: iree.compiler._mlir_libs._mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineModExpr`

Gets a semi-affine expression containing the modulo of dividing a constant by an expression.

3. `get_mod(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1: int) -> (anonymous namespace)::PyAffineModExpr`

Gets an affine expression containing the module of dividing an expression by a constant.

**static** `get_mul(*args, **kwargs)`

Overloaded function.

1. `get_mul(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1: iree.compiler._mlir_libs._mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineMulExpr`

Gets an affine expression containing a product of two expressions.

2. `get_mul(arg0: int, arg1: iree.compiler._mlir_libs._mlir.ir.AffineExpr) -> (anonymous namespace)::PyAffineMulExpr`

Gets an affine expression containing a product of a constant and another expression.

3. `get_mul(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1: int) -> (anonymous namespace)::PyAffineMulExpr`

Gets an affine expression containing a product of an expression and a constant.

**static** `get_symbol(position: int, context: mlir.ir.Context = None) -> (anonymous namespace)::PyAffineSymbolExpr`

Gets an affine expression of a symbol at the given position.

**class** `iree.compiler.ir.AffineExprList`

**class** `iree.compiler.ir.AffineFloorDivExpr`

**static** `get(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1: iree.compiler._mlir_libs._mlir.ir.AffineExpr) -> iree.compiler._mlir_libs._mlir.ir.AffineFloorDivExpr`

**static** `isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) -> bool`

**class** `iree.compiler.ir.AffineMap`

**static** `compress_unused_symbols(arg0: list, arg1: mlir.ir.Context) -> List[iree.compiler._mlir_libs._mlir.ir.AffineMap]`

**property** `context`

Context that owns the Affine Map

```

dump(self: iree.compiler._mlir_libs._mlir.ir.AffineMap) → None
    Dumps a debug representation of the object to stderr.

static get(dim_count: int, symbol_count: int, exprs: list, context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap
    Gets a map with the given expressions as results.

static get_constant(value: int, context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap
    Gets an affine map with a single constant result

static get_empty(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.AffineMap
    Gets an empty affine map.

static get_identity(n_dims: int, context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap
    Gets an identity map with the given number of dimensions.

get_major_submap(self: iree.compiler._mlir_libs._mlir.ir.AffineMap, n_results: int) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap

static get_minor_identity(n_dims: int, n_results: int, context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap
    Gets a minor identity map with the given number of dimensions and results.

get_minor_submap(self: iree.compiler._mlir_libs._mlir.ir.AffineMap, n_results: int) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap

static get_permutation(permutation: List[int], context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap
    Gets an affine map that permutes its inputs.

get_submap(self: iree.compiler._mlir_libs._mlir.ir.AffineMap, result_positions: List[int]) →
    iree.compiler._mlir_libs._mlir.ir.AffineMap

property is_permutation
property is_projected_permutation
property n_dims
property n_inputs
property n_symbols

replace(self: iree.compiler._mlir_libs._mlir.ir.AffineMap, expr: iree.compiler._mlir_libs._mlir.ir.AffineExpr,
    replacement: iree.compiler._mlir_libs._mlir.ir.AffineExpr, n_result_dims: int, n_result_syms: int)
    → iree.compiler._mlir_libs._mlir.ir.AffineMap

property results

class iree.compiler.ir.AffineMapAttr

    static get(affine_map: iree.compiler._mlir_libs._mlir.ir.AffineMap) →
        iree.compiler._mlir_libs._mlir.ir.AffineMapAttr
        Gets an attribute wrapping an AffineMap.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type

class iree.compiler.ir.AffineModExpr
    
```

```

    static get(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1:
        iree.compiler._mlir_libs._mlir.ir.AffineExpr) → iree.compiler._mlir_libs._mlir.ir.AffineModExpr

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
class iree.compiler.ir.AffineMulExpr

    static get(arg0: iree.compiler._mlir_libs._mlir.ir.AffineExpr, arg1:
        iree.compiler._mlir_libs._mlir.ir.AffineExpr) → iree.compiler._mlir_libs._mlir.ir.AffineMulExpr

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
class iree.compiler.ir.AffineSymbolExpr

    static get(position: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.AffineSymbolExpr

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.AffineExpr) → bool
    property position
class iree.compiler.ir.ArrayAttr

    static get(attributes: list, context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.ArrayAttr
        Gets a uniqued Array attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
    property type
class iree.compiler.ir.ArrayAttributeIterator
class iree.compiler.ir.AttrBuilder

    static contains(arg0: str) → bool

    static get(arg0: str) → function

    static insert(arg0: str, arg1: function) → None
class iree.compiler.ir.Attribute

    property context
        Context that owns the Attribute

    dump(self: iree.compiler._mlir_libs._mlir.ir.Attribute) → None
        Dumps a debug representation of the object to stderr.

    get_named(self: iree.compiler._mlir_libs._mlir.ir.Attribute, arg0: str) → mlir::python::PyNamedAttribute
        Binds a name to the attribute

    static parse(asm: str, context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Attribute
        Parses an attribute from an assembly form. Raises an MLIRError on failure.

    property type
class iree.compiler.ir.BF16Type

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.BF16Type
        Create a bf16 type.

```



```

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.Block

    append(self: iree.compiler._mlir_libs._mlir.ir.Block, operation:
        iree.compiler._mlir_libs._mlir.ir.OperationBase) → None
        Appends an operation to this block. If the operation is currently in another block, it will be moved.

    append_to(self: iree.compiler._mlir_libs._mlir.ir.Block, arg0: iree.compiler._mlir_libs._mlir.ir.Region) →
        None
        Append this block to a region, transferring ownership if necessary

    property arguments
        Returns a list of block arguments.

    create_after(self: iree.compiler._mlir_libs._mlir.ir.Block, *args) → iree.compiler._mlir_libs._mlir.ir.Block
        Creates and returns a new Block after this block (with given argument types).

    static create_at_start(parent: iree.compiler._mlir_libs._mlir.ir.Region, arg_types: list = []) →
        iree.compiler._mlir_libs._mlir.ir.Block
        Creates and returns a new Block at the beginning of the given region (with given argument types).

    create_before(self: iree.compiler._mlir_libs._mlir.ir.Block, *args) →
        iree.compiler._mlir_libs._mlir.ir.Block
        Creates and returns a new Block before this block (with given argument types).

    property operations
        Returns a forward-optimized sequence of operations.

    property owner
        Returns the owning operation of this block.

    property region
        Returns the owning region of this block.
class iree.compiler.ir.BlockArgument

    property arg_number

    static isinstance(other_value: iree.compiler._mlir_libs._mlir.ir.Value) → bool

    property owner

    set_type(self: iree.compiler._mlir_libs._mlir.ir.BlockArgument, type:
        iree.compiler._mlir_libs._mlir.ir.Type) → None
class iree.compiler.ir.BlockArgumentList

    property types

class iree.compiler.ir.BlockIterator

class iree.compiler.ir.BlockList

    append(self: iree.compiler._mlir_libs._mlir.ir.BlockList, *args) → iree.compiler._mlir_libs._mlir.ir.Block
        Appends a new block, with argument types as positional args.

        Returns The created block.

class iree.compiler.ir.BoolAttr

```

**static get**(*value: bool, context: mlir.ir.Context = None*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.BoolAttr*  
 Gets an unqued bool attribute

**static isinstance**(*other: iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute*) → bool

**property type**

**property value**

Returns the value of the bool attribute

**class iree.compiler.ir.ComplexType**

**property element\_type**

Returns element type.

**static get**(*arg0: iree.compiler.\_mlir\_libs.\_mlir.ir.Type*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.ComplexType*  
 Create a complex type

**static isinstance**(*other: iree.compiler.\_mlir\_libs.\_mlir.ir.Type*) → bool

**class iree.compiler.ir.Context**(\*args, \*\*kwargs)

**class iree.compiler.ir.DenseBoolArrayAttr**

**static get**(*values: List[int], context: mlir.ir.Context = None*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.DenseBoolArrayAttr*  
 Gets a unqued dense array attribute

**static isinstance**(*other: iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute*) → bool

**property type**

**class iree.compiler.ir.DenseBoolArrayIterator**

**class iree.compiler.ir.DenseElementsAttr**

**static get**(*array: buffer, signless: bool = True, type: Optional[iree.compiler.\_mlir\_libs.\_mlir.ir.Type] = None, shape: Optional[List[int]] = None, context: mlir.ir.Context = None*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.DenseElementsAttr*  
 Gets a DenseElementsAttr from a Python buffer or array.

When *type* is not provided, then some limited type inferencing is done based on the buffer format. Support presently exists for 8/16/32/64 signed and unsigned integers and float16/float32/float64. DenseElementsAttr of these types can also be converted back to a corresponding buffer.

For conversions outside of these types, a *type=* must be explicitly provided and the buffer contents must be bit-castable to the MLIR internal representation:

- Integer types (except for i1): the buffer must be byte aligned to the next byte boundary.
- Floating point types: Must be bit-castable to the given floating point size.
- i1 (bool): Bit packed into 8bit words where the bit pattern matches a row major ordering. An arbitrary Numpy *bool\_* array can be bit packed to this specification with: *np.packbits(ary, axis=None, bitorder='little')*.

If a single element buffer is passed (or for i1, a single byte with value 0 or 255), then a splat will be created.

#### Parameters

- **array** – The array or buffer to convert.

- **signless** – If inferring an appropriate MLIR type, use signless types for integers (defaults True).
- **type** – Skips inference of the MLIR element type and uses this instead. The storage size must be consistent with the actual contents of the buffer.
- **shape** – Overrides the shape of the buffer when constructing the MLIR shaped type. This is needed when the physical and logical shape differ (as for i1).
- **context** – Explicit context, if not from context manager.

**Returns** DenseElementsAttr on success.

**Raises** **ValueError** – If the type of the buffer or array cannot be matched to an MLIR type or if the buffer does not meet expectations.

```
static get_splat(shaped_type: iree.compiler._mlir_libs._mlir.ir.Type, element_attr:
    iree.compiler._mlir_libs._mlir.ir.Attribute) →
    iree.compiler._mlir_libs._mlir.ir.DenseElementsAttr
```

Gets a DenseElementsAttr where all values are the same

```
get_splat_value(self: iree.compiler._mlir_libs._mlir.ir.DenseElementsAttr) →
    iree.compiler._mlir_libs._mlir.ir.Attribute
```

property is\_splat

```
static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
```

property type

```
class iree.compiler.ir.DenseF32ArrayAttr
```

```
static get(values: List[float], context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.DenseF32ArrayAttr
```

Gets a uniqued dense array attribute

```
static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
```

property type

```
class iree.compiler.ir.DenseF32ArrayIterator
```

```
class iree.compiler.ir.DenseF64ArrayAttr
```

```
static get(values: List[float], context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.DenseF64ArrayAttr
```

Gets a uniqued dense array attribute

```
static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
```

property type

```
class iree.compiler.ir.DenseF64ArrayIterator
```

```
class iree.compiler.ir.DenseFPElementsAttr
```

```
static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
```

property type

```
class iree.compiler.ir.DenseI16ArrayAttr
```

```

    static get(values: List[int], context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.DenseI16ArrayAttr
        Gets a uniqued dense array attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type
class iree.compiler.ir.DenseI16ArrayIterator
class iree.compiler.ir.DenseI32ArrayAttr

    static get(values: List[int], context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.DenseI32ArrayAttr
        Gets a uniqued dense array attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type
class iree.compiler.ir.DenseI32ArrayIterator
class iree.compiler.ir.DenseI64ArrayAttr

    static get(values: List[int], context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.DenseI64ArrayAttr
        Gets a uniqued dense array attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type
class iree.compiler.ir.DenseI64ArrayIterator
class iree.compiler.ir.DenseI8ArrayAttr

    static get(values: List[int], context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.DenseI8ArrayAttr
        Gets a uniqued dense array attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type
class iree.compiler.ir.DenseI8ArrayIterator
class iree.compiler.ir.DenseIntElementsAttr

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type
class iree.compiler.ir.Diagnostic

    property location
    property message
    property notes
    property severity

```

```

class iree.compiler.ir.DiagnosticHandler

    property attached
    detach(self: iree.compiler._mlir_libs._mlir.ir.DiagnosticHandler) → None
    property had_error

class iree.compiler.ir.DiagnosticInfo

    property location
    property message
    property notes
    property severity

class iree.compiler.ir.DiagnosticSeverity
    Members:
    ERROR
    WARNING
    NOTE
    REMARK
    ERROR = <DiagnosticSeverity.ERROR: 0>
    NOTE = <DiagnosticSeverity.NOTE: 2>
    REMARK = <DiagnosticSeverity.REMARK: 3>
    WARNING = <DiagnosticSeverity.WARNING: 1>
    property name
    property value

class iree.compiler.ir.Dialect

    property descriptor

class iree.compiler.ir.DialectDescriptor

    property namespace

class iree.compiler.ir.DialectRegistry
class iree.compiler.ir.Dialects
class iree.compiler.ir.DictAttr

    static get(value: dict = {}, context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.DictAttr
        Gets an uniqued dict attribute
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
    property type

class iree.compiler.ir.F16Type

```

```

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.F16Type
        Create a f16 type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.F32Type

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.F32Type
        Create a f32 type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.F64Type

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.F64Type
        Create a f64 type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.FlatSymbolRefAttr

    static get(value: str, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.FlatSymbolRefAttr
        Gets a uniqued FlatSymbolRef attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
    property type
    property value
        Returns the value of the FlatSymbolRef attribute as a string
class iree.compiler.ir.Float8E4M3B11FNUZType

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Float8E4M3B11FNUZType
        Create a float8_e4m3b11fnuz type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.Float8E4M3FNType

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Float8E4M3FNType
        Create a float8_e4m3fn type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.Float8E4M3FNUZType

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Float8E4M3FNUZType
        Create a float8_e4m3fnuz type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
class iree.compiler.ir.Float8E5M2FNUZType

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Float8E5M2FNUZType
        Create a float8_e5m2fnuz type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

```

```
class iree.compiler.ir.Float8E5M2Type
```

```
    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Float8E5M2Type
        Create a float8_e5m2 type.
```

```
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
```

```
class iree.compiler.ir.FloatAttr
```

```
    static get(type: iree.compiler._mlir_libs._mlir.ir.Type, value: float, loc: mlir.ir.Location = None) →
        iree.compiler._mlir_libs._mlir.ir.FloatAttr
```

```
        Gets an unquied float point attribute associated to a type
```

```
    static get_f32(value: float, context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.FloatAttr
        Gets an unquied float point attribute associated to a f32 type
```

```
    static get_f64(value: float, context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.FloatAttr
        Gets an unquied float point attribute associated to a f64 type
```

```
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool
```

```
    property type
```

```
    property value
```

```
        Returns the value of the float point attribute
```

```
class iree.compiler.ir.FunctionType
```

```
    static get(inputs: List[iree.compiler._mlir_libs._mlir.ir.Type], results:
        List[iree.compiler._mlir_libs._mlir.ir.Type], context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.FunctionType
```

```
        Gets a FunctionType from a list of input and result types
```

```
    property inputs
```

```
        Returns the list of input types in the FunctionType.
```

```
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
```

```
    property results
```

```
        Returns the list of result types in the FunctionType.
```

```
class iree.compiler.ir.IndexType
```

```
    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.IndexType
        Create an index type.
```

```
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
```

```
class iree.compiler.ir.InferTypeOpInterface
```

```
    inferReturnTypes(self: iree.compiler._mlir_libs._mlir.ir.InferTypeOpInterface, operands: Optional[list] =
        None, attributes: Optional[iree.compiler._mlir_libs._mlir.ir.Attribute] = None, regions:
        Optional[List[iree.compiler._mlir_libs._mlir.ir.Region]] = None, context: mlir.ir.Context
        = None, loc: mlir.ir.Location = None) → List[iree.compiler._mlir_libs._mlir.ir.Type]
```

```
        Given the arguments required to build an operation, attempts to infer its return types. Raises ValueError
        on failure.
```

```
    property operation
```

```
        Returns an Operation for which the interface was constructed.
```

**property opview**

Returns an OpView subclass `_instance_` for which the interface was constructed

**class** iree.compiler.ir.InsertionPoint

**static** `at_block_begin(block: iree.compiler._mlir_libs._mlir.ir.Block) →`

`iree.compiler._mlir_libs._mlir.ir.InsertionPoint`

Inserts at the beginning of the block.

**static** `at_block_terminator(block: iree.compiler._mlir_libs._mlir.ir.Block) →`

`iree.compiler._mlir_libs._mlir.ir.InsertionPoint`

Inserts before the block terminator.

**property block**

Returns the block that this InsertionPoint points to.

**property current**

**insert**(*self*: iree.compiler.\_mlir\_libs.\_mlir.ir.InsertionPoint, operation:

`iree.compiler._mlir_libs._mlir.ir.OperationBase`) → `None`

Inserts an operation.

**class** iree.compiler.ir.IntegerAttr

**static** `get(type: iree.compiler._mlir_libs._mlir.ir.Type, value: int) →`

`iree.compiler._mlir_libs._mlir.ir.IntegerAttr`

Gets an unique integer attribute associated to a type

**static** `isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool`

**property type**

**property value**

Returns the value of the integer attribute

**class** iree.compiler.ir.IntegerSet

**property constraints**

**property context**

**dump**(*self*: iree.compiler.\_mlir\_libs.\_mlir.ir.IntegerSet) → `None`

Dumps a debug representation of the object to stderr.

**static** `get(num_dims: int, num_symbols: int, exprs: list, eq_flags: List[bool], context: mlir.ir.Context =`

`None`) → `iree.compiler._mlir_libs._mlir.ir.IntegerSet`

**static** `get_empty(num_dims: int, num_symbols: int, context: mlir.ir.Context = None) →`

`iree.compiler._mlir_libs._mlir.ir.IntegerSet`

**get\_replaced**(*self*: iree.compiler.\_mlir\_libs.\_mlir.ir.IntegerSet, dim\_exprs: list, symbol\_exprs: list,

`num_result_dims: int, num_result_symbols: int`) →

`iree.compiler._mlir_libs._mlir.ir.IntegerSet`

**property is\_canonical\_empty**

**property n\_dims**

**property n\_equalities**

**property n\_inequalities**



```

    property n_inputs
    property n_symbols
class iree.compiler.ir.IntegerSetConstraint

    property expr
    property is_eq
class iree.compiler.ir.IntegerSetConstraintList
class iree.compiler.ir.IntegerType

    static get_signed(width: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.IntegerType
        Create a signed integer type
    static get_signless(width: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.IntegerType
        Create a signless integer type
    static get_unsigned(width: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.IntegerType
        Create an unsigned integer type
    property is_signed
        Returns whether this is a signed integer
    property is_signless
        Returns whether this is a signless integer
    property is_unsigned
        Returns whether this is an unsigned integer
    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool
    property width
        Returns the width of the integer type
class iree.compiler.ir.Location

    property attr
        Get the underlying LocationAttr
    static callsite(callee: iree.compiler._mlir_libs._mlir.ir.Location, frames:
        List[iree.compiler._mlir_libs._mlir.ir.Location], context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.Location
        Gets a Location representing a caller and callsite
    property context
        Context that owns the Location
    property current
    emit_error(self: iree.compiler._mlir_libs._mlir.ir.Location, message: str) → None
        Emits an error at this location
    static file(filename: str, line: int, col: int, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.Location
        Gets a Location representing a file, line and column

```

```

static from_attr(attribute: mlir::python::PyAttribute, context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.Location
    Gets a Location from a LocationAttr

static fused(locations: List[iree.compiler._mlir_libs._mlir.ir.Location], metadata:
    Optional[mlir::python::PyAttribute] = None, context: mlir.ir.Context = None) →
    iree.compiler._mlir_libs._mlir.ir.Location
    Gets a Location representing a fused location with optional metadata

static name(name: str, childLoc: Optional[iree.compiler._mlir_libs._mlir.ir.Location] = None, context:
    mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Location
    Gets a Location representing a named location with optional child location

static unknown(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Location
    Gets a Location representing an unknown location

```

```

exception iree.compiler.ir.MLIRError(message, error_diagnostics)

```

An exception with diagnostic information. Has the following fields: message: str error\_diagnostics: List[ir.DiagnosticInfo]

```

class iree.compiler.ir.MemRefType

```

```

property affine_map

```

The layout of the MemRef type as an affine map.

```

static get(shape: List[int], element_type: iree.compiler._mlir_libs._mlir.ir.Type, layout:
    iree.compiler._mlir_libs._mlir.ir.Attribute = None, memory_space:
    iree.compiler._mlir_libs._mlir.ir.Attribute = None, loc: mlir.ir.Location = None) →
    iree.compiler._mlir_libs._mlir.ir.MemRefType

```

Create a memref type

```

static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

```

```

property layout

```

The layout of the MemRef type.

```

property memory_space

```

Returns the memory space of the given MemRef type.

```

class iree.compiler.ir.Module

```

```

property body

```

Return the block for this module

```

property context

```

Context that created the Module

```

static create(loc: mlir.ir.Location = None) → object

```

Creates an empty module

```

dump(self: iree.compiler._mlir_libs._mlir.ir.Module) → None

```

Dumps a debug representation of the object to stderr.

```

property operation

```

Accesses the module as an operation

```

static parse(asm: str, context: mlir.ir.Context = None) → object

```

Parses a module's assembly format from a string.

Returns a new MlirModule or raises an MLIR error if the parsing fails.

See also: <https://mlir.llvm.org/docs/LangRef/>

**class** iree.compiler.ir.NamedAttribute

**property** attr

The underlying generic attribute of the NamedAttribute binding

**property** name

The name of the NamedAttribute binding

**class** iree.compiler.ir.NoneType

**static** get(context: mlir.ir.Context = None) → iree.compiler.\_mlir\_libs.\_mlir.ir.NoneType

Create a none type.

**static** isinstance(other: iree.compiler.\_mlir\_libs.\_mlir.ir.Type) → bool

**class** iree.compiler.ir.OpAttributeMap

**class** iree.compiler.ir.OpOperand

**property** operand\_number

**property** owner

**class** iree.compiler.ir.OpOperandIterator

**class** iree.compiler.ir.OpOperandList

**class** iree.compiler.ir.OpResult

**static** isinstance(other\_value: iree.compiler.\_mlir\_libs.\_mlir.ir.Value) → bool

**property** owner

**property** result\_number

**class** iree.compiler.ir.OpResultList

**property** types

**class** iree.compiler.ir.OpView

**classmethod** build\_generic()

(cls: object, results: list = None, operands: list = None, attributes: Optional[dict] = None, successors: Optional[List[mlir::python::PyBlock]] = None, regions: Optional[int] = None, loc: mlir.ir.Location = None, ip: object = None) -> object

Builds a specific, generated OpView based on class level attributes.

**property** context

Context that owns the Operation

**property** operation

**classmethod** parse()

(cls: object, source: str, \*, source\_name: str = "", context: mlir.ir.Context = None) -> object

Parses a specific, generated OpView based on class level attributes

```
class iree.compiler.ir.OpaqueAttr
```

**property data**

Returns the data for the Opaqued attributes as a string

**property dialect\_namespace**

Returns the dialect namespace for the Opaque attribute as a string

**static get**(*dialect\_namespace: str, buffer: buffer, type: iree.compiler.\_mlir\_libs.\_mlir.ir.Type, context: mlir.ir.Context = None*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.OpaqueAttr*

Gets an Opaque attribute.

**static isinstance**(*other: iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute*) → bool

**property type**

```
class iree.compiler.ir.OpaqueType
```

**property data**

Returns the data for the Opaque type as a string.

**property dialect\_namespace**

Returns the dialect namespace for the Opaque type as a string.

**static get**(*dialect\_namespace: str, buffer: str, context: mlir.ir.Context = None*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.OpaqueType*

Create an unregistered (opaque) dialect type.

**static isinstance**(*other: iree.compiler.\_mlir\_libs.\_mlir.ir.Type*) → bool

```
class iree.compiler.ir.Operation
```

**clone**(*self: iree.compiler.\_mlir\_libs.\_mlir.ir.Operation, ip: object = None*) → *object*

**property context**

Context that owns the Operation

**static create**(*name: str, results: Optional[List[mlir::python::PyType]] = None, operands: Optional[List[mlir::python::PyValue]] = None, attributes: Optional[dict] = None, successors: Optional[List[mlir::python::PyBlock]] = None, regions: int = 0, loc: mlir.ir.Location = None, ip: object = None*) → *object*

Creates a new operation.

**Parameters**

- **name** – Operation name (e.g. “dialect.operation”).
- **results** – Sequence of Type representing op result types.
- **attributes** – Dict of str:Attribute.
- **successors** – List of Block for the operation’s successors.
- **regions** – Number of regions to create.
- **location** – A Location object (defaults to resolve from context manager).
- **ip** – An InsertionPoint (defaults to resolve from context manager or set to False to disable insertion, even with an insertion point set in the context manager).

**Returns** A new “detached” Operation object. Detached operations can be added to blocks, which causes them to become “attached.”

```

erase(self: iree.compiler._mlir_libs._mlir.ir.Operation) → None

property name
property opview
property parent

static parse(source: str, *, source_name: str = "", context: mlir.ir.Context = None) → object
    Parses an operation. Supports both text assembly format and binary bytecode format.

class iree.compiler.ir.OperationIterator
class iree.compiler.ir.OperationList
class iree.compiler.ir.RankedTensorType

    property encoding

    static get(shape: List[int], element_type: iree.compiler._mlir_libs._mlir.ir.Type, encoding:
        Optional[iree.compiler._mlir_libs._mlir.ir.Attribute] = None, loc: mlir.ir.Location = None) →
        iree.compiler._mlir_libs._mlir.ir.RankedTensorType
        Create a ranked tensor type

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

class iree.compiler.ir.Region

    property blocks
        Returns a forward-optimized sequence of blocks.

    property owner
        Returns the operation owning this region.

class iree.compiler.ir.RegionIterator
class iree.compiler.ir.RegionSequence
class iree.compiler.ir.ShapedType

    property element_type
        Returns the element type of the shaped type.

    get_dim_size(self: iree.compiler._mlir_libs._mlir.ir.ShapedType, dim: int) → int
        Returns the dim-th dimension of the given ranked shaped type.

    static get_dynamic_size() → int
        Returns the value used to indicate dynamic dimensions in shaped types.

    static get_dynamic_stride_or_offset() → int
        Returns the value used to indicate dynamic strides or offsets in shaped types.

    property has_rank
        Returns whether the given shaped type is ranked.

    property has_static_shape
        Returns whether the given shaped type has a static shape.

    is_dynamic_dim(self: iree.compiler._mlir_libs._mlir.ir.ShapedType, dim: int) → bool
        Returns whether the dim-th dimension of the given shaped type is dynamic.

    static is_dynamic_size(dim_size: int) → bool
        Returns whether the given dimension size indicates a dynamic dimension.

```

**is\_dynamic\_stride\_or\_offset**(*self*: iree.compiler.\_mlir\_libs.\_mlir.ir.ShapedType, *dim\_size*: int) → bool  
Returns whether the given value is used as a placeholder for dynamic strides and offsets in shaped types.

**static isinstance**(*other*: iree.compiler.\_mlir\_libs.\_mlir.ir.Type) → bool

**property rank**

Returns the rank of the given ranked shaped type.

**property shape**

Returns the shape of the ranked shaped type as a list of integers.

**class** iree.compiler.ir.StridedLayoutAttr

**static get**(*offset*: int, *strides*: List[int], *context*: mlir.ir.Context = None) →  
*iree.compiler.\_mlir\_libs.\_mlir.ir.StridedLayoutAttr*

Gets a strided layout attribute.

**static get\_fully\_dynamic**(*rank*: int, *context*: mlir.ir.Context = None) →  
*iree.compiler.\_mlir\_libs.\_mlir.ir.StridedLayoutAttr*

Gets a strided layout attribute with dynamic offset and strides of a given rank.

**static isinstance**(*other*: iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute) → bool

**property offset**

Returns the value of the float point attribute

**property strides**

Returns the value of the float point attribute

**property type**

**class** iree.compiler.ir.StringAttr

**static get**(*value*: str, *context*: mlir.ir.Context = None) → *iree.compiler.\_mlir\_libs.\_mlir.ir.StringAttr*  
Gets a uniqued string attribute

**static get\_typed**(*type*: iree.compiler.\_mlir\_libs.\_mlir.ir.Type, *value*: str) →  
*iree.compiler.\_mlir\_libs.\_mlir.ir.StringAttr*

Gets a uniqued string attribute associated to a type

**static isinstance**(*other*: iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute) → bool

**property type**

**property value**

Returns the value of the string attribute

**class** iree.compiler.ir.SymbolTable

**erase**(*self*: iree.compiler.\_mlir\_libs.\_mlir.ir.SymbolTable, *operation*:  
*iree.compiler.\_mlir\_libs.\_mlir.ir.\_OperationBase*) → None

**static get\_symbol\_name**(*symbol*: iree.compiler.\_mlir\_libs.\_mlir.ir.\_OperationBase) →  
*iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute*

**static get\_visibility**(*symbol*: iree.compiler.\_mlir\_libs.\_mlir.ir.\_OperationBase) →  
*iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute*

**insert**(*self*: iree.compiler.\_mlir\_libs.\_mlir.ir.SymbolTable, *operation*:  
*iree.compiler.\_mlir\_libs.\_mlir.ir.\_OperationBase*) → *iree.compiler.\_mlir\_libs.\_mlir.ir.Attribute*

```

static replace_all_symbol_uses(old_symbol: str, new_symbol: str, from_op:
                               iree.compiler._mlir_libs._mlir.ir._OperationBase) → None

static set_symbol_name(symbol: iree.compiler._mlir_libs._mlir.ir._OperationBase, name: str) → None

static set_visibility(symbol: iree.compiler._mlir_libs._mlir.ir._OperationBase, visibility: str) → None

static walk_symbol_tables(from_op: iree.compiler._mlir_libs._mlir.ir._OperationBase,
                          all_sym_uses_visible: bool, callback: object) → None

class iree.compiler.ir.TupleType

    static get_tuple(elements: list, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.TupleType
        Create a tuple type

    get_type(self: iree.compiler._mlir_libs._mlir.ir.TupleType, pos: int) →
        iree.compiler._mlir_libs._mlir.ir.Type
        Returns the pos-th type in the tuple type.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

    property num_types
        Returns the number of types contained in a tuple.

class iree.compiler.ir.Type

    property context
        Context that owns the Type

    dump(self: iree.compiler._mlir_libs._mlir.ir.Type) → None
        Dumps a debug representation of the object to stderr.

    static parse(asm: str, context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.Type
        Parses the assembly form of a type.

        Returns a Type object or raises an MLIR error if the type cannot be parsed.

        See also: https://mlir.llvm.org/docs/LangRef/#type-system

class iree.compiler.ir.TypeAttr

    static get(value: iree.compiler._mlir_libs._mlir.ir.Type, context: mlir.ir.Context = None) →
        iree.compiler._mlir_libs._mlir.ir.TypeAttr
        Gets a unique Type attribute

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type

    property value

class iree.compiler.ir.UnitAttr

    static get(context: mlir.ir.Context = None) → iree.compiler._mlir_libs._mlir.ir.UnitAttr
        Create a Unit attribute.

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Attribute) → bool

    property type

```

```

class iree.compiler.ir.UnrankedMemRefType

    static get(element_type: iree.compiler._mlir_libs._mlir.ir.Type, memory_space:
        iree.compiler._mlir_libs._mlir.ir.Attribute, loc: mlir.ir.Location = None) →
        iree.compiler._mlir_libs._mlir.ir.UnrankedMemRefType
        Create a unranked memref type

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

    property memory_space
        Returns the memory space of the given Unranked MemRef type.

class iree.compiler.ir.UnrankedTensorType

    static get(element_type: iree.compiler._mlir_libs._mlir.ir.Type, loc: mlir.ir.Location = None) →
        iree.compiler._mlir_libs._mlir.ir.UnrankedTensorType
        Create a unranked tensor type

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

class iree.compiler.ir.Value

    property context
        Context in which the value lives.

    dump(self: iree.compiler._mlir_libs._mlir.ir.Value) → None
        Dumps a debug representation of the object to stderr.

    property owner

    property type

    property uses

class iree.compiler.ir.VectorType

    static get(shape: List[int], elementType: iree.compiler._mlir_libs._mlir.ir.Type, loc: mlir.ir.Location =
        None) → iree.compiler._mlir_libs._mlir.ir.VectorType
        Create a vector type

    static isinstance(other: iree.compiler._mlir_libs._mlir.ir.Type) → bool

iree.compiler.ir.register_attribute_builder(kind)

```

## 2.7.2 iree.compiler.passmanager module

```

class iree.compiler.passmanager.PassManager

    add(self: iree.compiler._mlir_libs._mlir.passmanager.PassManager, pipeline: str) → None
        Add textual pipeline elements to the pass manager. Throws a ValueError if the pipeline can't be parsed.

    enable_ir_printing(self: iree.compiler._mlir_libs._mlir.passmanager.PassManager) → None
        Enable mlir-print-ir-after-all.

    enable_verifier(self: iree.compiler._mlir_libs._mlir.passmanager.PassManager, enable: bool) → None
        Enable / disable verify-each.

```



```
static parse(pipeline: str, context: mlir.ir.Context = None) →  

    iree.compiler._mlir_libs._mlir.passmanager.PassManager  

    Parse a textual pass-pipeline and return a top-level PassManager that can be applied on a Module. Throw  

    a ValueError if the pipeline can't be parsed
```

```
run(self: iree.compiler._mlir_libs._mlir.passmanager.PassManager, operation:  

    iree.compiler._mlir_libs._mlir.ir._OperationBase) → None  

    Run the pass manager on the provided operation, raising an MLIRError on failure.
```



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### i

- `iree.compiler.dialects.builtin`, [17](#)
- `iree.compiler.dialects.linalg`, [18](#)
- `iree.compiler.dialects.math`, [31](#)
- `iree.compiler.dialects.memref`, [37](#)
- `iree.compiler.dialects.shape`, [42](#)
- `iree.compiler.dialects.tensor`, [48](#)
- `iree.compiler.dialects.tosa`, [52](#)
- `iree.compiler.dialects.vector`, [64](#)
- `iree.compiler.ir`, [72](#)
- `iree.compiler.passmanager`, [92](#)
- `iree.compiler.tools`, [11](#)
- `iree.compiler.tools.debugging`, [13](#)
- `iree.compiler.tools.tf`, [16](#)
- `iree.compiler.tools.tflite`, [15](#)
- `iree.runtime`, [1](#)



## A

- `a` (*iree.compiler.dialects.math.FmaOp* property), 34
- `a` (*iree.compiler.dialects.tosa.MatMulOp* property), 59
- `abs` (*iree.compiler.dialects.linalg.UnaryFn* attribute), 29
- `AbsFOp` (class in *iree.compiler.dialects.math*), 31
- `AbsIOp` (class in *iree.compiler.dialects.math*), 31
- `AbsOp` (class in *iree.compiler.dialects.tosa*), 52
- `acc` (*iree.compiler.dialects.vector.ContractionOp* property), 65
- `acc` (*iree.compiler.dialects.vector.FMAOp* property), 66
- `acc` (*iree.compiler.dialects.vector.MultiDimReductionOp* property), 68
- `acc` (*iree.compiler.dialects.vector.OuterProductOp* property), 68
- `acc` (*iree.compiler.dialects.vector.ReductionOp* property), 68
- `accumulated_value` (*iree.compiler.dialects.vector.ScanOp* property), 69
- `add` (*iree.compiler.dialects.linalg.BinaryFn* attribute), 18
- `add` (*iree.compiler.dialects.linalg.ReduceFn* attribute), 25
- `add()` (*iree.compiler.passmanager.PassManager* method), 92
- `add_indexed_operand()` (*iree.compiler.dialects.linalg.LinalgStructuredOpConfig* method), 22
- `add_operand()` (*iree.compiler.dialects.linalg.LinalgOpDef* method), 22
- `add_operand()` (*iree.compiler.dialects.linalg.LinalgStructuredOpConfig* method), 22
- `add_tensor_use()` (*iree.compiler.dialects.linalg.LinalgStructuredOpConfig* method), 22
- `add_vm_module()` (*iree.runtime.SystemContext* method), 8
- `add_vm_modules()` (*iree.runtime.SystemContext* method), 8
- `AddOp` (class in *iree.compiler.dialects.shape*), 42
- `AddOp` (class in *iree.compiler.dialects.tosa*), 52
- `affine_map` (*iree.compiler.ir.MemRefType* property), 86
- `AffineAddExpr` (class in *iree.compiler.ir*), 72
- `AffineBinaryExpr` (class in *iree.compiler.ir*), 72
- `AffineBuildState` (class in *iree.compiler.dialects.linalg*), 18
- `AffineCeilDivExpr` (class in *iree.compiler.ir*), 72
- `AffineConstantExpr` (class in *iree.compiler.ir*), 72
- `AffineDimExpr` (class in *iree.compiler.ir*), 72
- `AffineExpr` (class in *iree.compiler.ir*), 72
- `AffineExprDef` (class in *iree.compiler.dialects.linalg*), 18
- `AffineExprList` (class in *iree.compiler.ir*), 74
- `AffineFloorDivExpr` (class in *iree.compiler.ir*), 74
- `AffineMap` (class in *iree.compiler.ir*), 74
- `AffineMapAttr` (class in *iree.compiler.ir*), 75
- `AffineModExpr` (class in *iree.compiler.ir*), 75
- `AffineMulExpr` (class in *iree.compiler.ir*), 76
- `AffineSymbolExpr` (class in *iree.compiler.ir*), 76
- `aggregate` (*iree.compiler.dialects.tensor.SplatOp* property), 52
- `aggregate` (*iree.compiler.dialects.vector.SplatOp* property), 70
- `aligned_pointer` (*iree.compiler.dialects.memref.ExtractAlignedPointerAsIntValueOp* property), 39
- `alignment` (*iree.compiler.dialects.memref.AllocaOp* property), 37
- `alignment` (*iree.compiler.dialects.memref.AllocOp* property), 37
- `alignment` (*iree.compiler.dialects.memref.AssumeAlignmentOp* property), 37
- `alignment` (*iree.compiler.dialects.memref.GlobalOp* property), 40
- `alignment` (*iree.compiler.dialects.memref.ReallocOp* property), 40
- `ALL` (*iree.runtime.MemoryAccess* attribute), 7
- `ALL_DIMS` (*iree.compiler.dialects.linalg.DimDef* attribute), 20
- `all_reduction_dims` (*iree.compiler.dialects.linalg.Comprehension* property), 19
- `ALL_SYMBOLS` (*iree.compiler.dialects.linalg.SymbolDef* attribute), 27
- `ALL_TYPEVARS` (*iree.compiler.dialects.linalg.TypeVar* attribute), 29
- `alloc_optional()` (*iree.compiler.tools.debugging.TempFileSaver* method), 14
- `AllocaOp` (class in *iree.compiler.dialects.memref*), 37
- `AllocaScopeOp` (class in *iree.compiler.dialects.memref*), 37

*iree.compiler.dialects.memref*), 37  
 AllocaScopeReturnOp (class in *iree.compiler.dialects.memref*), 37  
 ALLOCATABLE (*iree.runtime.BufferCompatibility* attribute), 1  
 allocate\_buffer() (*iree.runtime.HalAllocator* method), 3  
 allocate\_buffer\_copy() (*iree.runtime.HalAllocator* method), 4  
 allocator (*iree.runtime.HalDevice* property), 4  
 AllocOp (class in *iree.compiler.dialects.memref*), 37  
 AnyOp (class in *iree.compiler.dialects.shape*), 42  
 append() (*iree.compiler.ir.Block* method), 77  
 append() (*iree.compiler.ir.BlockList* method), 77  
 append\_to() (*iree.compiler.ir.Block* method), 77  
 ApplyScaleOp (class in *iree.compiler.dialects.tosa*), 53  
 arg (*iree.compiler.dialects.shape.IndexToSizeOp* property), 45  
 arg (*iree.compiler.dialects.shape.ShapeOfOp* property), 47  
 arg (*iree.compiler.dialects.shape.SizeToIndexOp* property), 47  
 arg (*iree.compiler.dialects.shape.ValueAsShapeOp* property), 48  
 arg (*iree.compiler.dialects.shape.ValueOfOp* property), 48  
 arg0 (*iree.compiler.dialects.shape.MeetOp* property), 46  
 arg1 (*iree.compiler.dialects.shape.MeetOp* property), 46  
 arg\_number (*iree.compiler.ir.BlockArgument* property), 77  
 ArgMaxOp (class in *iree.compiler.dialects.tosa*), 53  
 args (*iree.compiler.dialects.vector.WarpExecuteOnLane0Op* property), 71  
 arguments (*iree.compiler.ir.Block* property), 77  
 ArithmeticRightShiftOp (class in *iree.compiler.dialects.tosa*), 53  
 ArrayAttr (class in *iree.compiler.ir*), 76  
 ArrayAttributeIterator (class in *iree.compiler.ir*), 76  
 as\_linalg\_yaml() (*iree.compiler.dialects.linalg.YAMLOperation* method), 30  
 asdevicearray() (in module *iree.runtime*), 9  
 AssumeAlignmentOp (class in *iree.compiler.dialects.memref*), 37  
 AssumingAllOp (class in *iree.compiler.dialects.shape*), 42  
 AssumingOp (class in *iree.compiler.dialects.shape*), 42  
 AssumingYieldOp (class in *iree.compiler.dialects.shape*), 42  
 astype() (*iree.runtime.DeviceArray* method), 3  
 at\_block\_begin() (*iree.compiler.ir.InsertionPoint* static method), 84  
 at\_block\_terminator() (*iree.compiler.ir.InsertionPoint* static method), 84  
 Atan2Op (class in *iree.compiler.dialects.math*), 32  
 AtanOp (class in *iree.compiler.dialects.math*), 32  
 atomic\_body (*iree.compiler.dialects.memref.GenericAtomicRMWOp* property), 39  
 AtomicRMWOp (class in *iree.compiler.dialects.memref*), 37  
 AtomicYieldOp (class in *iree.compiler.dialects.memref*), 37  
 attach() (*iree.compiler.dialects.linalg.OperandDef* method), 23  
 attached (*iree.compiler.ir.DiagnosticHandler* property), 81  
 attr (*iree.compiler.ir.Location* property), 85  
 attr (*iree.compiler.ir.NamedAttribute* property), 87  
 AttrBuilder (class in *iree.compiler.ir*), 76  
 Attribute (class in *iree.compiler.ir*), 76  
 AvgPool2dOp (class in *iree.compiler.dialects.tosa*), 53  
 axis (*iree.compiler.dialects.tosa.ArgMaxOp* property), 53  
 axis (*iree.compiler.dialects.tosa.ConcatOp* property), 55  
 axis (*iree.compiler.dialects.tosa.ReduceAllOp* property), 60  
 axis (*iree.compiler.dialects.tosa.ReduceAnyOp* property), 60  
 axis (*iree.compiler.dialects.tosa.ReduceMaxOp* property), 61  
 axis (*iree.compiler.dialects.tosa.ReduceMinOp* property), 61  
 axis (*iree.compiler.dialects.tosa.ReduceProdOp* property), 61  
 axis (*iree.compiler.dialects.tosa.ReduceSumOp* property), 61  
 axis (*iree.compiler.dialects.tosa.ReverseOp* property), 62  
**B**  
 b (*iree.compiler.dialects.math.FmaOp* property), 34  
 b (*iree.compiler.dialects.tosa.MatMulOp* property), 59  
 base (*iree.compiler.dialects.vector.CompressStoreOp* property), 64  
 base (*iree.compiler.dialects.vector.ExpandLoadOp* property), 65  
 base (*iree.compiler.dialects.vector.GatherOp* property), 66  
 base (*iree.compiler.dialects.vector.LoadOp* property), 67  
 base (*iree.compiler.dialects.vector.MaskedLoadOp* property), 67  
 base (*iree.compiler.dialects.vector.MaskedStoreOp* property), 67  
 base (*iree.compiler.dialects.vector.ScatterOp* property), 69  
 base (*iree.compiler.dialects.vector.StoreOp* property), 70



[base\\_buffer](#) (*iree.compiler.dialects.memref.ExtractStridedSliceOp* property), 39  
[BatchMatmulOp](#) (class in *iree.compiler.dialects.linalg*), 18  
[BatchMatmulTransposeBOp](#) (class in *iree.compiler.dialects.linalg*), 18  
[BatchMatvecOp](#) (class in *iree.compiler.dialects.linalg*), 18  
[BatchReduceMatmulOp](#) (class in *iree.compiler.dialects.linalg*), 18  
[begin\\_profiling\(\)](#) (*iree.runtime.HalDevice* method), 4  
[benchmark\\_exe\(\)](#) (in module *iree.runtime*), 9  
[benchmark\\_module\(\)](#) (in module *iree.runtime*), 9  
[BF16Type](#) (class in *iree.compiler.ir*), 76  
[BFLOAT\\_16](#) (*iree.runtime.HalElementType* attribute), 5  
[bias](#) (*iree.compiler.dialects.tosa.Conv2DOp* property), 55  
[bias](#) (*iree.compiler.dialects.tosa.Conv3DOp* property), 55  
[bias](#) (*iree.compiler.dialects.tosa.DepthwiseConv2DOp* property), 56  
[bias](#) (*iree.compiler.dialects.tosa.FullyConnectedOp* property), 57  
[bias](#) (*iree.compiler.dialects.tosa.TransposeConv2DOp* property), 63  
[BINARY](#) (*iree.compiler.dialects.linalg.FunctionKind* attribute), 21  
[BINARY\\_FN\\_ATTR](#) (*iree.compiler.dialects.linalg.OperandKind* attribute), 24  
[BinaryFn](#) (class in *iree.compiler.dialects.linalg*), 18  
[BinaryFnAttrDef](#) (class in *iree.compiler.dialects.linalg*), 19  
[BinaryFnType](#) (class in *iree.compiler.dialects.linalg*), 19  
[bind\\_op\\_def\(\)](#) (in module *iree.compiler.dialects.linalg*), 30  
[BitCastOp](#) (class in *iree.compiler.dialects.vector*), 64  
[BitwiseAndOp](#) (class in *iree.compiler.dialects.tosa*), 53  
[BitwiseNotOp](#) (class in *iree.compiler.dialects.tosa*), 53  
[BitwiseOrOp](#) (class in *iree.compiler.dialects.tosa*), 54  
[BitwiseXorOp](#) (class in *iree.compiler.dialects.tosa*), 54  
[Block](#) (class in *iree.compiler.ir*), 77  
[block](#) (*iree.compiler.ir.InsertionPoint* property), 84  
[BlockArgument](#) (class in *iree.compiler.ir*), 77  
[BlockArgumentList](#) (class in *iree.compiler.ir*), 77  
[BlockIterator](#) (class in *iree.compiler.ir*), 77  
[BlockList](#) (class in *iree.compiler.ir*), 77  
[blocks](#) (*iree.compiler.ir.Region* property), 89  
[body](#) (*iree.compiler.dialects.shape.FuncOp* property), 45  
[body](#) (*iree.compiler.dialects.shape.FunctionLibraryOp* property), 45  
[body](#) (*iree.compiler.dialects.tensor.GenerateOp* property), 50  
[body](#) (*iree.compiler.dialects.tosa.WhileOp* property), 64  
[body](#) (*iree.compiler.ir.Module* property), 86  
[bodyRegion](#) (*iree.compiler.dialects.memref.AllocascopeOp* property), 37  
[BOOL\\_8](#) (*iree.runtime.HalElementType* attribute), 5  
[BoolAttr](#) (class in *iree.compiler.ir*), 77  
[BroadcastOp](#) (class in *iree.compiler.dialects.linalg*), 19  
[BroadcastOp](#) (class in *iree.compiler.dialects.shape*), 43  
[BroadcastOp](#) (class in *iree.compiler.dialects.vector*), 64  
[BufferCompatibility](#) (class in *iree.runtime*), 1  
[BufferUsage](#) (class in *iree.runtime*), 1  
[build\(\)](#) (*iree.compiler.dialects.linalg.AffineExprDef* method), 18  
[build\\_generic\(\)](#) (*iree.compiler.ir.OpView* class method), 87  
[byte\\_shift](#) (*iree.compiler.dialects.memref.ViewOp* property), 42

## C

[c](#) (*iree.compiler.dialects.math.FmaOp* property), 34  
[c](#) (*iree.compiler.dialects.tosa.MatMulOp* property), 59  
[callsite\(\)](#) (*iree.compiler.ir.Location* static method), 85  
[cast\\_signed](#) (*iree.compiler.dialects.linalg.TypeFn* attribute), 29  
[cast\\_unsigned](#) (*iree.compiler.dialects.linalg.TypeFn* attribute), 29  
[CastOp](#) (class in *iree.compiler.dialects.memref*), 38  
[CastOp](#) (class in *iree.compiler.dialects.tensor*), 48  
[CastOp](#) (class in *iree.compiler.dialects.tosa*), 54  
[CbrtOp](#) (class in *iree.compiler.dialects.math*), 32  
[ceil](#) (*iree.compiler.dialects.linalg.UnaryFn* attribute), 29  
[CeilOp](#) (class in *iree.compiler.dialects.math*), 32  
[CeilOp](#) (class in *iree.compiler.dialects.tosa*), 54  
[ClampOp](#) (class in *iree.compiler.dialects.tosa*), 54  
[clone\(\)](#) (*iree.compiler.ir.Operation* method), 88  
[ClzOp](#) (class in *iree.compiler.dialects.tosa*), 54  
[coerce\\_from\(\)](#) (*iree.compiler.dialects.linalg.AffineExprDef* static method), 18  
[CollapseShapeOp](#) (class in *iree.compiler.dialects.memref*), 38  
[CollapseShapeOp](#) (class in *iree.compiler.dialects.tensor*), 48  
[collect\\_dim\\_uses\(\)](#) (*iree.compiler.dialects.linalg.TensorExpression* method), 27  
[collect\\_indices\(\)](#) (*iree.compiler.dialects.linalg.TensorExpression* method), 27  
[collect\\_scalar\\_uses\(\)](#) (*iree.compiler.dialects.linalg.TensorExpression* method), 27  
[collect\\_tensor\\_uses\(\)](#) (*iree.compiler.dialects.linalg.TensorExpression* method), 27  
[columns](#) (*iree.compiler.dialects.vector.FlatTransposeOp* property), 66

combiner (*iree.compiler.dialects.linalg.ReduceOp* property), 25

compile\_file() (in module *iree.compiler.tools*), 11

compile\_file() (in module *iree.compiler.tools.tflite*), 15

compile\_module() (in module *iree.compiler.tools.tf*), 16

compile\_saved\_model() (in module *iree.compiler.tools.tf*), 16

compile\_str() (in module *iree.compiler.tools*), 12

compile\_str() (in module *iree.compiler.tools.tflite*), 15

CompilerOptions (class in *iree.compiler.tools*), 12

COMPLEX\_128 (*iree.runtime.HalElementType* attribute), 5

COMPLEX\_64 (*iree.runtime.HalElementType* attribute), 5

ComplexType (class in *iree.compiler.ir*), 78

compose() (*iree.compiler.ir.AffineExpr* method), 73

Comprehension (class in *iree.compiler.dialects.linalg*), 19

compress\_unused\_symbols() (*iree.compiler.ir.AffineMap* static method), 74

CompressStoreOp (class in *iree.compiler.dialects.vector*), 64

ConcatOp (class in *iree.compiler.dialects.shape*), 43

ConcatOp (class in *iree.compiler.dialects.tosa*), 55

cond (*iree.compiler.dialects.tosa.IfOp* property), 57

cond (*iree.compiler.dialects.tosa.WhileOp* property), 64

Config (class in *iree.runtime*), 3

config (*iree.compiler.dialects.tosa.CustomOp* property), 55

config (*iree.runtime.SystemContext* property), 8

const (class in *iree.compiler.dialects.linalg*), 30

constant (*iree.compiler.dialects.memref.GlobalOp* property), 40

ConstantMaskOp (class in *iree.compiler.dialects.vector*), 64

ConstOp (class in *iree.compiler.dialects.tosa*), 55

constraints (*iree.compiler.ir.IntegerSet* property), 84

ConstShapeOp (class in *iree.compiler.dialects.shape*), 43

ConstSizeOp (class in *iree.compiler.dialects.shape*), 43

ConstWitnessOp (class in *iree.compiler.dialects.shape*), 43

contains() (*iree.compiler.ir.AttrBuilder* static method), 76

Context (class in *iree.compiler.ir*), 78

context (*iree.compiler.ir.AffineExpr* property), 73

context (*iree.compiler.ir.AffineMap* property), 74

context (*iree.compiler.ir.Attribute* property), 76

context (*iree.compiler.ir.IntegerSet* property), 84

context (*iree.compiler.ir.Location* property), 85

context (*iree.compiler.ir.Module* property), 86

context (*iree.compiler.ir.Operation* property), 88

context (*iree.compiler.ir.OpView* property), 87

context (*iree.compiler.ir.Type* property), 91

context (*iree.compiler.ir.Value* property), 92

context\_id (*iree.runtime.VmContext* property), 8

contextmanager() (in module *iree.compiler.dialects.linalg*), 30

ContractionOp (class in *iree.compiler.dialects.vector*), 65

Conv1DNCwFcwOp (class in *iree.compiler.dialects.linalg*), 19

Conv1DNwcWcfOp (class in *iree.compiler.dialects.linalg*), 19

Conv1DOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DNchwFchwOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DNghwFgchwOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DNhwcFhwcOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DNhwcHwcfOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DNhwcHwcfQOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DOp (class in *iree.compiler.dialects.linalg*), 19

Conv2DOp (class in *iree.compiler.dialects.tosa*), 55

Conv3DNdhwcDhwcFOp (class in *iree.compiler.dialects.linalg*), 19

Conv3DNdhwcDhwcFQOp (class in *iree.compiler.dialects.linalg*), 20

Conv3DOp (class in *iree.compiler.dialects.linalg*), 20

Conv3DOp (class in *iree.compiler.dialects.tosa*), 55

CopyOp (class in *iree.compiler.dialects.linalg*), 20

CopyOp (class in *iree.compiler.dialects.memref*), 38

CopySignOp (class in *iree.compiler.dialects.math*), 32

CosOp (class in *iree.compiler.dialects.math*), 32

CountLeadingZerosOp (class in *iree.compiler.dialects.math*), 33

CountTrailingZerosOp (class in *iree.compiler.dialects.math*), 33

create() (*iree.compiler.ir.Module* static method), 86

create() (*iree.compiler.ir.Operation* static method), 88

create() (*iree.runtime.PyModuleInterface* method), 8

create\_after() (*iree.compiler.ir.Block* method), 77

create\_at\_start() (*iree.compiler.ir.Block* static method), 77

create\_before() (*iree.compiler.ir.Block* method), 77

create\_default\_device() (*iree.runtime.HalDriver* method), 4

create\_device() (*iree.runtime.HalDriver* method), 4

create\_device\_by\_uri() (*iree.runtime.HalDriver* method), 4

create\_expando() (*iree.compiler.dialects.linalg.DimDef* class method), 20

create\_expando() (*iree.compiler.dialects.linalg.SymbolDef* class method), 27

create\_expando() (*iree.compiler.dialects.linalg.TypeVar*

[class method](#)), 29  
[create\\_hal\\_module\(\)](#) (in module [iree.runtime](#)), 9  
[create\\_view\(\)](#) ([iree.runtime.HalBuffer](#) method), 4  
[CreateMaskOp](#) (class in [iree.compiler.dialects.vector](#)), 65  
[CstrBroadcastableOp](#) (class in [iree.compiler.dialects.shape](#)), 43  
[CstrEqOp](#) (class in [iree.compiler.dialects.shape](#)), 44  
[CstrRequireOp](#) (class in [iree.compiler.dialects.shape](#)), 44  
[CtPopOp](#) (class in [iree.compiler.dialects.math](#)), 33  
[current](#) ([iree.compiler.ir.InsertionPoint](#) property), 84  
[current](#) ([iree.compiler.ir.Location](#) property), 85  
[current\(\)](#) ([iree.compiler.tools.debugging.TempFileSaver](#) static method), 14  
[current\\_op\\_def\(\)](#) (in module [iree.compiler.dialects.linalg](#)), 30  
[CustomOp](#) (class in [iree.compiler.dialects.tosa](#)), 55

## D

[data](#) ([iree.compiler.ir.OpaqueAttr](#) property), 88  
[data](#) ([iree.compiler.ir.OpaqueType](#) property), 88  
[DeallocOp](#) (class in [iree.compiler.dialects.memref](#)), 38  
[DebugPrintOp](#) (class in [iree.compiler.dialects.shape](#)), 44  
[DEFAULT](#) ([iree.runtime.BufferUsage](#) attribute), 2  
[default\\_vm\\_modules](#) ([iree.runtime.Config](#) attribute), 3  
[DefinedOpCallable](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[defines\(\)](#) (in module [iree.compiler.dialects.linalg](#)), 31  
[DenseBoolArrayAttr](#) (class in [iree.compiler.ir](#)), 78  
[DenseBoolArrayIterator](#) (class in [iree.compiler.ir](#)), 78  
[DenseElementsAttr](#) (class in [iree.compiler.ir](#)), 78  
[DenseF32ArrayAttr](#) (class in [iree.compiler.ir](#)), 79  
[DenseF32ArrayIterator](#) (class in [iree.compiler.ir](#)), 79  
[DenseF64ArrayAttr](#) (class in [iree.compiler.ir](#)), 79  
[DenseF64ArrayIterator](#) (class in [iree.compiler.ir](#)), 79  
[DenseFPElementsAttr](#) (class in [iree.compiler.ir](#)), 79  
[DenseI16ArrayAttr](#) (class in [iree.compiler.ir](#)), 79  
[DenseI16ArrayIterator](#) (class in [iree.compiler.ir](#)), 80  
[DenseI32ArrayAttr](#) (class in [iree.compiler.ir](#)), 80  
[DenseI32ArrayIterator](#) (class in [iree.compiler.ir](#)), 80  
[DenseI64ArrayAttr](#) (class in [iree.compiler.ir](#)), 80  
[DenseI64ArrayIterator](#) (class in [iree.compiler.ir](#)), 80  
[DenseI8ArrayAttr](#) (class in [iree.compiler.ir](#)), 80  
[DenseI8ArrayIterator](#) (class in [iree.compiler.ir](#)), 80  
[DenseIntElementsAttr](#) (class in [iree.compiler.ir](#)), 80  
[DepthwiseConv1DNwcWcmOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv1DNwcWcOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv2DNchwChwOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv2DNhwcHwcOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv2DNhwcHwcMQOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv2DNhwcHwcOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv2DNhwcHwcMQOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv2DOp](#) (class in [iree.compiler.dialects.tosa](#)), 55  
[DepthwiseConv3DNdhwcDhwcOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[DepthwiseConv3DNdhwcDhwcMQOp](#) (class in [iree.compiler.dialects.linalg](#)), 20  
[descriptor](#) ([iree.compiler.ir.Dialect](#) property), 81  
[dest](#) ([iree.compiler.dialects.memref.CastOp](#) property), 38  
[dest](#) ([iree.compiler.dialects.memref.MemorySpaceCastOp](#) property), 40  
[dest](#) ([iree.compiler.dialects.tensor.CastOp](#) property), 48  
[dest](#) ([iree.compiler.dialects.tensor.InsertOp](#) property), 50  
[dest](#) ([iree.compiler.dialects.tensor.InsertSliceOp](#) property), 50  
[dest](#) ([iree.compiler.dialects.tensor.PackOp](#) property), 50  
[dest](#) ([iree.compiler.dialects.tensor.ParallelInsertSliceOp](#) property), 51  
[dest](#) ([iree.compiler.dialects.tensor.ScatterOp](#) property), 52  
[dest](#) ([iree.compiler.dialects.tensor.UnPackOp](#) property), 52  
[dest](#) ([iree.compiler.dialects.vector.InsertElementOp](#) property), 66  
[dest](#) ([iree.compiler.dialects.vector.InsertOp](#) property), 66  
[dest](#) ([iree.compiler.dialects.vector.InsertStridedSliceOp](#) property), 67  
[dest](#) ([iree.compiler.dialects.vector.MultiDimReductionOp](#) property), 68  
[dest](#) ([iree.compiler.dialects.vector.ReductionOp](#) property), 68  
[dest](#) ([iree.compiler.dialects.vector.ScalableInsertOp](#) property), 69  
[dest](#) ([iree.compiler.dialects.vector.ScanOp](#) property), 69  
[destroyed](#) ([iree.runtime.PyModuleInterface](#) property), 8  
[detach\(\)](#) ([iree.compiler.ir.DiagnosticHandler](#) method), 81  
[device](#) ([iree.runtime.Config](#) attribute), 3  
[DEVICE\\_LOCAL](#) ([iree.runtime.MemoryType](#) attribute), 7  
[DEVICE\\_VISIBLE](#) ([iree.runtime.MemoryType](#) attribute), 7  
[DeviceArray](#) (class in [iree.runtime](#)), 3  
[Diagnostic](#) (class in [iree.compiler.ir](#)), 80  
[DiagnosticHandler](#) (class in [iree.compiler.ir](#)), 80

[DiagnosticInfo](#) (class in `iree.compiler.ir`), 81  
[DiagnosticSeverity](#) (class in `iree.compiler.ir`), 81  
[Dialect](#) (class in `iree.compiler.ir`), 81  
[dialect\\_namespace](#) (`iree.compiler.ir.OpaqueAttr` property), 88  
[dialect\\_namespace](#) (`iree.compiler.ir.OpaqueType` property), 88  
[DialectDescriptor](#) (class in `iree.compiler.ir`), 81  
[DialectRegistry](#) (class in `iree.compiler.ir`), 81  
[Dialects](#) (class in `iree.compiler.ir`), 81  
[DictAttr](#) (class in `iree.compiler.ir`), 81  
[dim](#) (`iree.compiler.dialects.linalg.IndexOp` property), 21  
[dim](#) (`iree.compiler.dialects.shape.GetExtentOp` property), 45  
[dim\\_count](#) (`iree.compiler.dialects.linalg.AffineBuildState` property), 18  
[DimDef](#) (class in `iree.compiler.dialects.linalg`), 20  
[DimOp](#) (class in `iree.compiler.dialects.memref`), 38  
[DimOp](#) (class in `iree.compiler.dialects.shape`), 44  
[DimOp](#) (class in `iree.compiler.dialects.tensor`), 48  
[DISCARD](#) (`iree.runtime.MemoryAccess` attribute), 7  
[DISCARD\\_WRITE](#) (`iree.runtime.MemoryAccess` attribute), 7  
[DISPATCH\\_IMAGE](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_IMAGE\\_READ](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_IMAGE\\_WRITE](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_INDIRECT\\_PARAMS](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_STORAGE](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_STORAGE\\_READ](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_STORAGE\\_WRITE](#) (`iree.runtime.BufferUsage` attribute), 2  
[DISPATCH\\_UNIFORM\\_READ](#) (`iree.runtime.BufferUsage` attribute), 2  
[DivOp](#) (class in `iree.compiler.dialects.shape`), 44  
[DivOp](#) (class in `iree.compiler.dialects.tosa`), 56  
[DmaStartOp](#) (class in `iree.compiler.dialects.memref`), 38  
[DmaWaitOp](#) (class in `iree.compiler.dialects.memref`), 38  
[doc](#) (`iree.compiler.dialects.linalg.GenericOp` property), 21  
[domain\(\)](#) (in module `iree.compiler.dialects.linalg`), 31  
[doRegion](#) (`iree.compiler.dialects.shape.AssumingOp` property), 42  
[DotOp](#) (class in `iree.compiler.dialects.linalg`), 20  
[double\\_round](#) (`iree.compiler.dialects.tosa.ApplyScaleOp` property), 53  
[double\\_round](#) (`iree.compiler.dialects.tosa.RescaleOp` property), 61  
[dtype](#) (`iree.runtime.DeviceArray` property), 3  
[dump\(\)](#) (`iree.compiler.ir.AffineExpr` method), 73  
[dump\(\)](#) (`iree.compiler.ir.AffineMap` method), 74  
[dump\(\)](#) (`iree.compiler.ir.Attribute` method), 76  
[dump\(\)](#) (`iree.compiler.ir.IntegerSet` method), 84  
[dump\(\)](#) (`iree.compiler.ir.Module` method), 86  
[dump\(\)](#) (`iree.compiler.ir.Type` method), 91  
[dump\(\)](#) (`iree.compiler.ir.Value` method), 92  
[dynamicExtents](#) (`iree.compiler.dialects.tensor.GenerateOp` property), 50  
[dynamicResultSize](#) (`iree.compiler.dialects.memref.ReallocOp` property), 40  
[dynamicSizes](#) (`iree.compiler.dialects.memref.AllocOp` property), 37  
[dynamicSizes](#) (`iree.compiler.dialects.memref.AllocOp` property), 37

## E

[element\\_type](#) (`iree.compiler.ir.ComplexType` property), 78  
[element\\_type](#) (`iree.compiler.ir.ShapedType` property), 89  
[element\\_type](#) (`iree.runtime.HalBufferView` property), 4  
[elements](#) (`iree.compiler.dialects.tensor.FromElementsOp` property), 49  
[ElemwiseBinaryOp](#) (class in `iree.compiler.dialects.linalg`), 20  
[ElemwiseUnaryOp](#) (class in `iree.compiler.dialects.linalg`), 20  
[else\\_branch](#) (`iree.compiler.dialects.tosa.IfOp` property), 57  
[emit\\_error\(\)](#) (`iree.compiler.ir.Location` method), 85  
[emit\\_generic\\_structured\\_op\(\)](#) (in module `iree.compiler.dialects.linalg`), 31  
[emit\\_named\\_structured\\_op\(\)](#) (in module `iree.compiler.dialects.linalg`), 31  
[EmptyOp](#) (class in `iree.compiler.dialects.tensor`), 49  
[enable\\_ir\\_printing\(\)](#) (`iree.compiler.passmanager.PassManager` method), 92  
[enable\\_verifier\(\)](#) (`iree.compiler.passmanager.PassManager` method), 92  
[encoding](#) (`iree.compiler.ir.RankedTensorType` property), 89  
[end\\_profiling\(\)](#) (`iree.runtime.HalDevice` method), 4  
[EqualOp](#) (class in `iree.compiler.dialects.tosa`), 56  
[erase\(\)](#) (`iree.compiler.ir.Operation` method), 88  
[erase\(\)](#) (`iree.compiler.ir.SymbolTable` method), 90  
[ErfOp](#) (class in `iree.compiler.dialects.math`), 33  
[error](#) (`iree.compiler.dialects.shape.BroadcastOp` property), 43  
[error](#) (`iree.compiler.dialects.shape.MeetOp` property), 46  
[ERROR](#) (`iree.compiler.ir.DiagnosticSeverity` attribute), 81  
[exp](#) (`iree.compiler.dialects.linalg.UnaryFn` attribute), 29  
[Exp2Op](#) (class in `iree.compiler.dialects.math`), 33



ExpandLoadOp (class in *iree.compiler.dialects.vector*), 65

ExpandShapeOp (class in *iree.compiler.dialects.memref*), 39

ExpandShapeOp (class in *iree.compiler.dialects.tensor*), 49

ExpM1Op (class in *iree.compiler.dialects.math*), 33

ExpOp (class in *iree.compiler.dialects.math*), 33

ExpOp (class in *iree.compiler.dialects.tosa*), 56

EXPORT (*iree.runtime.Linkage* attribute), 6

export() (*iree.runtime.PyModuleInterface* method), 8

EXPORTABLE (*iree.runtime.BufferCompatibility* attribute), 1

expr (*iree.compiler.ir.IntegerSetConstraint* property), 85

expr() (*iree.compiler.dialects.linalg.ScalarArg* method), 26

expr() (*iree.compiler.dialects.linalg.ScalarConst* method), 26

expr() (*iree.compiler.dialects.linalg.ScalarFn* method), 26

expr() (*iree.compiler.dialects.linalg.ScalarIndex* method), 27

extent (*iree.compiler.dialects.shape.DimOp* property), 44

extent (*iree.compiler.dialects.shape.GetExtentOp* property), 45

extents (*iree.compiler.dialects.shape.FromExtentsOp* property), 45

ExtractAlignedPointerAsIndexOp (class in *iree.compiler.dialects.memref*), 39

ExtractElementOp (class in *iree.compiler.dialects.vector*), 65

ExtractOp (class in *iree.compiler.dialects.tensor*), 49

ExtractOp (class in *iree.compiler.dialects.vector*), 65

ExtractSliceOp (class in *iree.compiler.dialects.tensor*), 49

ExtractStridedMetadataOp (class in *iree.compiler.dialects.memref*), 39

ExtractStridedSliceOp (class in *iree.compiler.dialects.vector*), 65

**F**

F16Type (class in *iree.compiler.ir*), 81

F32Type (class in *iree.compiler.ir*), 82

F64Type (class in *iree.compiler.ir*), 82

FFT2dOp (class in *iree.compiler.dialects.tosa*), 56

file() (*iree.compiler.ir.Location* static method), 85

fill\_builtin\_region() (in *iree.compiler.dialects.linalg* module), 31

fill\_zero() (*iree.runtime.HalBuffer* method), 4

FillOp (class in *iree.compiler.dialects.linalg*), 21

FillRng2D0p (class in *iree.compiler.dialects.linalg*), 21

filter (*iree.compiler.dialects.tosa.TransposeConv2D0p* property), 63

FLATBUFFER\_BINARY (*iree.compiler.tools.OutputFormat* attribute), 13

FLATBUFFER\_TEXT (*iree.compiler.tools.OutputFormat* attribute), 13

FlatSymbolRefAttr (class in *iree.compiler.ir*), 82

FlatTransposeOp (class in *iree.compiler.dialects.vector*), 66

Float8E4M3B11FNUZType (class in *iree.compiler.ir*), 82

Float8E4M3FNType (class in *iree.compiler.ir*), 82

Float8E4M3FNUZType (class in *iree.compiler.ir*), 82

Float8E5M2FNUZType (class in *iree.compiler.ir*), 82

Float8E5M2Type (class in *iree.compiler.ir*), 82

FLOAT\_16 (*iree.runtime.HalElementType* attribute), 5

FLOAT\_32 (*iree.runtime.HalElementType* attribute), 5

FLOAT\_64 (*iree.runtime.HalElementType* attribute), 5

FloatAttr (class in *iree.compiler.ir*), 83

floor (*iree.compiler.dialects.linalg.UnaryFn* attribute), 29

FloorOp (class in *iree.compiler.dialects.math*), 34

FloorOp (class in *iree.compiler.dialects.tosa*), 56

FmaOp (class in *iree.compiler.dialects.math*), 34

FMAOp (class in *iree.compiler.dialects.vector*), 66

formatted\_statistics (*iree.runtime.HalAllocator* property), 4

FPowI0p (class in *iree.compiler.dialects.math*), 34

from\_attr() (*iree.compiler.ir.Location* static method), 85

from\_flatbuffer() (*iree.runtime.VmModule* static method), 9

from\_linalg\_op\_def() (*iree.compiler.dialects.linalg.LinalgOpConfig* static method), 22

FromElementsOp (class in *iree.compiler.dialects.tensor*), 49

FromExtentsOp (class in *iree.compiler.dialects.shape*), 44

FromExtentTensorOp (class in *iree.compiler.dialects.shape*), 44

FullyConnectedOp (class in *iree.compiler.dialects.tosa*), 57

FuncOp (class in *iree.compiler.dialects.shape*), 45

function\_names (*iree.runtime.VmModule* property), 9

FunctionInvoker (class in *iree.runtime*), 3

FunctionLibraryOp (class in *iree.compiler.dialects.shape*), 45

FunctionType (class in *iree.compiler.ir*), 83

fused() (*iree.compiler.ir.Location* static method), 86

**G**

GatherOp (class in *iree.compiler.dialects.tensor*), 49

GatherOp (class in *iree.compiler.dialects.tosa*), 57

GatherOp (class in *iree.compiler.dialects.vector*), 66

GenerateOp (class in *iree.compiler.dialects.tensor*), 50

GenericAtomicRMWOp (class in iree.compiler.dialects.memref), 39

GenericOp (class in iree.compiler.dialects.linalg), 21

get() (iree.compiler.ir.AffineAddExpr static method), 72

get() (iree.compiler.ir.AffineCeilDivExpr static method), 72

get() (iree.compiler.ir.AffineConstantExpr static method), 72

get() (iree.compiler.ir.AffineDimExpr static method), 72

get() (iree.compiler.ir.AffineFloorDivExpr static method), 74

get() (iree.compiler.ir.AffineMap static method), 75

get() (iree.compiler.ir.AffineMapAttr static method), 75

get() (iree.compiler.ir.AffineModExpr static method), 75

get() (iree.compiler.ir.AffineMulExpr static method), 76

get() (iree.compiler.ir.AffineSymbolExpr static method), 76

get() (iree.compiler.ir.ArrayAttr static method), 76

get() (iree.compiler.ir.AttrBuilder static method), 76

get() (iree.compiler.ir.BF16Type static method), 76

get() (iree.compiler.ir.BoolAttr static method), 77

get() (iree.compiler.ir.ComplexType static method), 78

get() (iree.compiler.ir.DenseBoolArrayAttr static method), 78

get() (iree.compiler.ir.DenseElementsAttr static method), 78

get() (iree.compiler.ir.DenseF32ArrayAttr static method), 79

get() (iree.compiler.ir.DenseF64ArrayAttr static method), 79

get() (iree.compiler.ir.DenseI16ArrayAttr static method), 79

get() (iree.compiler.ir.DenseI32ArrayAttr static method), 80

get() (iree.compiler.ir.DenseI64ArrayAttr static method), 80

get() (iree.compiler.ir.DenseI8ArrayAttr static method), 80

get() (iree.compiler.ir.DictAttr static method), 81

get() (iree.compiler.ir.F16Type static method), 81

get() (iree.compiler.ir.F32Type static method), 82

get() (iree.compiler.ir.F64Type static method), 82

get() (iree.compiler.ir.FlatSymbolRefAttr static method), 82

get() (iree.compiler.ir.Float8E4M3B11FNUZType static method), 82

get() (iree.compiler.ir.Float8E4M3FNType static method), 82

get() (iree.compiler.ir.Float8E4M3FNUZType static method), 82

get() (iree.compiler.ir.Float8E5M2FNUZType static method), 82

get() (iree.compiler.ir.Float8E5M2Type static method), 83

get() (iree.compiler.ir.FloatAttr static method), 83

get() (iree.compiler.ir.FunctionType static method), 83

get() (iree.compiler.ir.IndexType static method), 83

get() (iree.compiler.ir.IntegerAttr static method), 84

get() (iree.compiler.ir.IntegerSet static method), 84

get() (iree.compiler.ir.MemRefType static method), 86

get() (iree.compiler.ir.NoneType static method), 87

get() (iree.compiler.ir.OpaqueAttr static method), 88

get() (iree.compiler.ir.OpaqueType static method), 88

get() (iree.compiler.ir.RankedTensorType static method), 89

get() (iree.compiler.ir.StridedLayoutAttr static method), 90

get() (iree.compiler.ir.StringAttr static method), 90

get() (iree.compiler.ir.TypeAttr static method), 91

get() (iree.compiler.ir.UnitAttr static method), 91

get() (iree.compiler.ir.UnrankedMemRefType static method), 92

get() (iree.compiler.ir.UnrankedTensorType static method), 92

get() (iree.compiler.ir.VectorType static method), 92

get\_add() (iree.compiler.ir.AffineExpr static method), 73

get\_as\_list() (iree.runtime.VmVariantList method), 9

get\_as\_object() (iree.runtime.VmVariantList method), 9

get\_as\_ref() (iree.runtime.VmVariantList method), 9

get\_ceil\_div() (iree.compiler.ir.AffineExpr static method), 73

get\_constant() (iree.compiler.ir.AffineExpr static method), 73

get\_constant() (iree.compiler.ir.AffineMap static method), 75

get\_default\_tracer() (in module iree.runtime), 9

get\_device() (in module iree.runtime), 9

get\_dim() (iree.compiler.dialects.linalg.AffineBuildState method), 18

get\_dim() (iree.compiler.ir.AffineExpr static method), 73

get\_dim\_size() (iree.compiler.ir.ShapedType method), 89

get\_driver() (in module iree.runtime), 10

get\_dynamic\_size() (iree.compiler.ir.ShapedType static method), 89

get\_dynamic\_stride\_or\_offset() (iree.compiler.ir.ShapedType static method), 89

get\_empty() (iree.compiler.ir.AffineMap static method), 75

get\_empty() (iree.compiler.ir.IntegerSet static method), 84

get\_f32() (iree.compiler.ir.FloatAttr static method), 83

get\_f64() (iree.compiler.ir.FloatAttr static method), 83

get\_first\_device() (in module iree.runtime), 10

get\_floor\_div() (iree.compiler.ir.AffineExpr static

method), 73  
 get\_fully\_dynamic() (iree.compiler.ir.StridedLayoutAttr static method), 90  
 get\_identity() (iree.compiler.ir.AffineMap static method), 75  
 get\_major\_submap() (iree.compiler.ir.AffineMap method), 75  
 get\_minor\_identity() (iree.compiler.ir.AffineMap static method), 75  
 get\_minor\_submap() (iree.compiler.ir.AffineMap method), 75  
 get\_mod() (iree.compiler.ir.AffineExpr static method), 74  
 get\_mul() (iree.compiler.ir.AffineExpr static method), 74  
 get\_named() (iree.compiler.ir.Attribute method), 76  
 get\_permutation() (iree.compiler.ir.AffineMap static method), 75  
 get\_replaced() (iree.compiler.ir.IntegerSet method), 84  
 get\_serialized\_trace\_value() (iree.runtime.VmVariantList method), 9  
 get\_signed() (iree.compiler.ir.IntegerType static method), 85  
 get\_signless() (iree.compiler.ir.IntegerType static method), 85  
 get\_splat() (iree.compiler.ir.DenseElementsAttr static method), 79  
 get\_splat\_value() (iree.compiler.ir.DenseElementsAttr method), 79  
 get\_submap() (iree.compiler.ir.AffineMap method), 75  
 get\_symbol() (iree.compiler.dialects.linalg.AffineBuildState method), 18  
 get\_symbol() (iree.compiler.ir.AffineExpr static method), 74  
 get\_symbol\_name() (iree.compiler.ir.SymbolTable static method), 90  
 get\_tuple() (iree.compiler.ir.TupleType static method), 91  
 get\_type() (iree.compiler.ir.TupleType method), 91  
 get\_typed() (iree.compiler.ir.StringAttr static method), 90  
 get\_unique\_name() (iree.runtime.Tracer method), 8  
 get\_unsigned() (iree.compiler.ir.IntegerType static method), 85  
 get\_variant() (iree.runtime.VmVariantList method), 9  
 get\_visibility() (iree.compiler.ir.SymbolTable static method), 90  
 GetExtentOp (class in iree.compiler.dialects.shape), 45  
 GetGlobalOp (class in iree.compiler.dialects.memref), 39  
 GlobalOp (class in iree.compiler.dialects.memref), 39  
 GreaterEqualOp (class in iree.compiler.dialects.tosa), 57  
 GreaterOp (class in iree.compiler.dialects.tosa), 57  
**H**  
 had\_error (iree.compiler.ir.DiagnosticHandler property), 81  
 HalAllocator (class in iree.runtime), 3  
 HalBuffer (class in iree.runtime), 4  
 HalBufferView (class in iree.runtime), 4  
 HalDevice (class in iree.runtime), 4  
 HalDriver (class in iree.runtime), 4  
 HalElementType (class in iree.runtime), 4  
 has\_rank (iree.compiler.ir.ShapedType property), 89  
 has\_static\_shape (iree.compiler.ir.ShapedType property), 89  
 has\_statistics (iree.runtime.HalAllocator property), 4  
 head (iree.compiler.dialects.shape.SplitAtOp property), 47  
 high (iree.compiler.dialects.tensor.PadOp property), 51  
 HOST\_CACHED (iree.runtime.MemoryType attribute), 7  
 HOST\_COHERENT (iree.runtime.MemoryType attribute), 7  
 HOST\_LOCAL (iree.runtime.MemoryType attribute), 7  
 HOST\_VISIBLE (iree.runtime.MemoryType attribute), 7  
**I**  
 identifier (iree.compiler.dialects.tosa.CustomOp property), 55  
 IdentityOp (class in iree.compiler.dialects.tosa), 57  
 IfOp (class in iree.compiler.dialects.tosa), 57  
 implementation\_attrs (iree.compiler.dialects.tosa.CustomOp property), 55  
 implements() (in module iree.compiler.dialects.linalg), 31  
 implicit() (iree.compiler.tools.debugging.TempFileSaver static method), 14  
 IMPORT (iree.runtime.Linkage attribute), 6  
 IMPORT\_OPTIONAL (iree.runtime.Linkage attribute), 6  
 IMPORTABLE (iree.runtime.BufferCompatibility attribute), 1  
 ImportOptions (class in iree.compiler.tools.tf), 16  
 ImportOptions (class in iree.compiler.tools.tflite), 15  
 in\_ (iree.compiler.dialects.memref.TransposeOp property), 42  
 inclusive (iree.compiler.dialects.vector.ScanOp property), 69  
 index (class in iree.compiler.dialects.linalg), 31  
 index (iree.compiler.dialects.memref.DimOp property), 38  
 index (iree.compiler.dialects.shape.DimOp property), 44  
 index (iree.compiler.dialects.shape.SplitAtOp property), 47  
 index (iree.compiler.dialects.tensor.DimOp property), 49

[INDEX\\_ATTR](#) (*iree.compiler.dialects.linalg.OperandKind* attribute), 24  
[index\\_vec](#) (*iree.compiler.dialects.vector.GatherOp* property), 66  
[index\\_vec](#) (*iree.compiler.dialects.vector.ScatterOp* property), 69  
[IndexAttrDef](#) (class in *iree.compiler.dialects.linalg*), 21  
[indexing\\_maps](#) (*iree.compiler.dialects.linalg.LinalgStructureOp* property), 22  
[IndexOp](#) (class in *iree.compiler.dialects.linalg*), 21  
[IndexToSizeOp](#) (class in *iree.compiler.dialects.shape*), 45  
[IndexType](#) (class in *iree.compiler.ir*), 83  
[indices](#) (*iree.compiler.dialects.memref.AtomicRMWOp* property), 37  
[indices](#) (*iree.compiler.dialects.memref.GenericAtomicRMWOp* property), 39  
[indices](#) (*iree.compiler.dialects.memref.PrefetchOp* property), 40  
[indices](#) (*iree.compiler.dialects.memref.StoreOp* property), 41  
[indices](#) (*iree.compiler.dialects.tensor.ExtractOp* property), 49  
[indices](#) (*iree.compiler.dialects.tensor.GatherOp* property), 50  
[indices](#) (*iree.compiler.dialects.tensor.InsertOp* property), 50  
[indices](#) (*iree.compiler.dialects.tensor.ScatterOp* property), 52  
[indices](#) (*iree.compiler.dialects.tosa.GatherOp* property), 57  
[indices](#) (*iree.compiler.dialects.tosa.ScatterOp* property), 62  
[indices](#) (*iree.compiler.dialects.vector.CompressStoreOp* property), 64  
[indices](#) (*iree.compiler.dialects.vector.ExpandLoadOp* property), 65  
[indices](#) (*iree.compiler.dialects.vector.GatherOp* property), 66  
[indices](#) (*iree.compiler.dialects.vector.LoadOp* property), 67  
[indices](#) (*iree.compiler.dialects.vector.MaskedLoadOp* property), 67  
[indices](#) (*iree.compiler.dialects.vector.MaskedStoreOp* property), 67  
[indices](#) (*iree.compiler.dialects.vector.ScatterOp* property), 69  
[indices](#) (*iree.compiler.dialects.vector.StoreOp* property), 70  
[indices](#) (*iree.compiler.dialects.vector.TransferReadOp* property), 70  
[indices](#) (*iree.compiler.dialects.vector.TransferWriteOp* property), 70  
[inferReturnTypes\(\)](#) (*iree.compiler.ir.InferTypeOpInterface* method), 83  
[InferTypeOpInterface](#) (class in *iree.compiler.ir*), 83  
[init](#) (*iree.compiler.dialects.linalg.BroadcastOp* property), 19  
[init](#) (*iree.compiler.dialects.linalg.MapOp* property), 23  
[init](#) (*iree.compiler.dialects.linalg.TransposeOp* property), 28  
[initial\\_value](#) (*iree.compiler.dialects.memref.GlobalOp* property), 40  
[initial\\_value](#) (*iree.compiler.dialects.vector.ScanOp* property), 69  
[initialized](#) (*iree.runtime.PyModuleInterface* property), 8  
[inits](#) (*iree.compiler.dialects.linalg.ReduceOp* property), 26  
[input\\_vals](#) (*iree.compiler.dialects.shape.ReduceOp* property), 46  
[inner\\_tiles](#) (*iree.compiler.dialects.tensor.PackOp* property), 51  
[inner\\_tiles](#) (*iree.compiler.dialects.tensor.UnPackOp* property), 52  
[input](#) (*iree.compiler.dialects.linalg.BroadcastOp* property), 19  
[input](#) (*iree.compiler.dialects.linalg.TransposeOp* property), 29  
[input](#) (*iree.compiler.dialects.shape.DebugPrintOp* property), 44  
[input](#) (*iree.compiler.dialects.shape.FromExtentTensorOp* property), 44  
[input](#) (*iree.compiler.dialects.shape.ToExtentTensorOp* property), 47  
[input](#) (*iree.compiler.dialects.tensor.SplatOp* property), 52  
[input](#) (*iree.compiler.dialects.tosa.ArgMaxOp* property), 53  
[input](#) (*iree.compiler.dialects.tosa.AvgPool2dOp* property), 53  
[input](#) (*iree.compiler.dialects.tosa.CastOp* property), 54  
[input](#) (*iree.compiler.dialects.tosa.ClampOp* property), 54  
[input](#) (*iree.compiler.dialects.tosa.Conv2DOp* property), 55  
[input](#) (*iree.compiler.dialects.tosa.Conv3DOp* property), 55  
[input](#) (*iree.compiler.dialects.tosa.DepthwiseConv2DOp* property), 56  
[input](#) (*iree.compiler.dialects.tosa.FullyConnectedOp* property), 57  
[input](#) (*iree.compiler.dialects.tosa.MaxPool2dOp* property), 59  
[input](#) (*iree.compiler.dialects.tosa.ReduceAllOp* property), 60  
[input](#) (*iree.compiler.dialects.tosa.ReduceAnyOp* property), 60





erty), 58  
input2 (*iree.compiler.dialects.tosa.LogicalRightShiftOp* property), 58  
input2 (*iree.compiler.dialects.tosa.LogicalXorOp* property), 59  
input2 (*iree.compiler.dialects.tosa.MaximumOp* property), 59  
input2 (*iree.compiler.dialects.tosa.MinimumOp* property), 59  
input2 (*iree.compiler.dialects.tosa.MulOp* property), 59  
input2 (*iree.compiler.dialects.tosa.PowOp* property), 60  
input2 (*iree.compiler.dialects.tosa.SubOp* property), 63  
input\_imag (*iree.compiler.dialects.tosa.FFT2dOp* property), 56  
input\_real (*iree.compiler.dialects.tosa.FFT2dOp* property), 56  
input\_shape (*iree.compiler.dialects.vector.ReshapeOp* property), 69  
INPUT\_TENSOR (*iree.compiler.dialects.linalg.OperandKind* attribute), 24  
input\_zp (*iree.compiler.dialects.tosa.RescaleOp* property), 61  
inputs (*iree.compiler.dialects.builtin.UnrealizedConversionCastOp* property), 17  
inputs (*iree.compiler.dialects.linalg.GenericOp* property), 21  
inputs (*iree.compiler.dialects.linalg.MapOp* property), 23  
inputs (*iree.compiler.dialects.linalg.ReduceOp* property), 26  
inputs (*iree.compiler.dialects.shape.AnyOp* property), 42  
inputs (*iree.compiler.dialects.shape.AssumingAllOp* property), 42  
inputs (*iree.compiler.dialects.tosa.CustomOp* property), 55  
inputs (*iree.compiler.dialects.tosa.IfOp* property), 57  
inputs (*iree.compiler.dialects.tosa.WhileOp* property), 64  
inputs (*iree.compiler.dialects.tosa.YieldOp* property), 64  
inputs (*iree.compiler.ir.FunctionType* property), 83  
insert() (*iree.compiler.ir.AttrBuilder* static method), 76  
insert() (*iree.compiler.ir.InsertionPoint* method), 84  
insert() (*iree.compiler.ir.SymbolTable* method), 90  
InsertElementOp (class in *iree.compiler.dialects.vector*), 66  
InsertionPoint (class in *iree.compiler.ir*), 84  
InsertOp (class in *iree.compiler.dialects.tensor*), 50  
InsertOp (class in *iree.compiler.dialects.vector*), 66  
InsertSliceOp (class in *iree.compiler.dialects.tensor*), 50  
InsertStridedSliceOp (class in *iree.compiler.dialects.vector*), 67  
instance (*iree.runtime.SystemContext* property), 8  
INT\_16 (*iree.runtime.HalElementType* attribute), 5  
INT\_32 (*iree.runtime.HalElementType* attribute), 6  
INT\_4 (*iree.runtime.HalElementType* attribute), 6  
INT\_64 (*iree.runtime.HalElementType* attribute), 6  
INT\_8 (*iree.runtime.HalElementType* attribute), 6  
IntegerAttr (class in *iree.compiler.ir*), 84  
IntegerSet (class in *iree.compiler.ir*), 84  
IntegerSetConstraint (class in *iree.compiler.ir*), 85  
IntegerSetConstraintList (class in *iree.compiler.ir*), 85  
IntegerType (class in *iree.compiler.ir*), 85  
INTERNAL (*iree.runtime.Linkage* attribute), 6  
inverse (*iree.compiler.dialects.tosa.FFT2dOp* property), 56  
invoke() (*iree.runtime.VmContext* method), 8  
IPowIOp (class in *iree.compiler.dialects.math*), 34  
iree.compiler.dialects.builtin module, 17  
iree.compiler.dialects.linalg module, 18  
iree.compiler.dialects.math module, 31  
iree.compiler.dialects.memref module, 37  
iree.compiler.dialects.shape module, 42  
iree.compiler.dialects.tensor module, 48  
iree.compiler.dialects.tosa module, 52  
iree.compiler.dialects.vector module, 64  
iree.compiler.ir module, 72  
iree.compiler.passmanager module, 92  
iree.compiler.tools module, 11  
iree.compiler.tools.debugging module, 13  
iree.compiler.tools.tf module, 16  
iree.compiler.tools.tflite module, 15  
iree.runtime module, 1  
is\_attribute() (*iree.compiler.dialects.linalg.OperandDef* method), 23  
is\_canonical\_empty (*iree.compiler.ir.IntegerSet* property), 84  
is\_dynamic (*iree.runtime.SystemContext* property), 8  
is\_dynamic\_dim() (*iree.compiler.ir.ShapedType* method), 89

`is_dynamic_size()` (*iree.compiler.ir.ShapedType* static method), 89  
`is_dynamic_stride_or_offset()` (*iree.compiler.ir.ShapedType* method), 89  
`is_eq` (*iree.compiler.ir.IntegerSetConstraint* property), 85  
`is_host_accessible` (*iree.runtime.DeviceArray* property), 3  
`is_input()` (*iree.compiler.dialects.linalg.OperandDef* method), 23  
`is_permutation` (*iree.compiler.ir.AffineMap* property), 75  
`is_projected_permutation` (*iree.compiler.ir.AffineMap* property), 75  
`is_signed` (*iree.compiler.ir.IntegerType* property), 85  
`is_signless` (*iree.compiler.ir.IntegerType* property), 85  
`is_splat` (*iree.compiler.ir.DenseElementsAttr* property), 79  
`is_tensor()` (*iree.compiler.dialects.linalg.OperandDef* method), 23  
`is_unsigned` (*iree.compiler.ir.IntegerType* property), 85  
`IsBroadcastableOp` (class in *iree.compiler.dialects.shape*), 45  
`isDataCache` (*iree.compiler.dialects.memref.PrefetchOp* property), 40  
`isinstance()` (*iree.compiler.ir.AffineAddExpr* static method), 72  
`isinstance()` (*iree.compiler.ir.AffineBinaryExpr* static method), 72  
`isinstance()` (*iree.compiler.ir.AffineCeilDivExpr* static method), 72  
`isinstance()` (*iree.compiler.ir.AffineConstantExpr* static method), 72  
`isinstance()` (*iree.compiler.ir.AffineDimExpr* static method), 72  
`isinstance()` (*iree.compiler.ir.AffineFloorDivExpr* static method), 74  
`isinstance()` (*iree.compiler.ir.AffineMapAttr* static method), 75  
`isinstance()` (*iree.compiler.ir.AffineModExpr* static method), 76  
`isinstance()` (*iree.compiler.ir.AffineMulExpr* static method), 76  
`isinstance()` (*iree.compiler.ir.AffineSymbolExpr* static method), 76  
`isinstance()` (*iree.compiler.ir.ArrayAttr* static method), 76  
`isinstance()` (*iree.compiler.ir.BF16Type* static method), 76  
`isinstance()` (*iree.compiler.ir.BlockArgument* static method), 77  
`isinstance()` (*iree.compiler.ir.BoolAttr* static method), 78  
`isinstance()` (*iree.compiler.ir.ComplexType* static method), 78  
`isinstance()` (*iree.compiler.ir.DenseBoolArrayAttr* static method), 78  
`isinstance()` (*iree.compiler.ir.DenseElementsAttr* static method), 79  
`isinstance()` (*iree.compiler.ir.DenseF32ArrayAttr* static method), 79  
`isinstance()` (*iree.compiler.ir.DenseF64ArrayAttr* static method), 79  
`isinstance()` (*iree.compiler.ir.DenseFPElementsAttr* static method), 79  
`isinstance()` (*iree.compiler.ir.DenseI16ArrayAttr* static method), 80  
`isinstance()` (*iree.compiler.ir.DenseI32ArrayAttr* static method), 80  
`isinstance()` (*iree.compiler.ir.DenseI64ArrayAttr* static method), 80  
`isinstance()` (*iree.compiler.ir.DenseI8ArrayAttr* static method), 80  
`isinstance()` (*iree.compiler.ir.DenseIntElementsAttr* static method), 80  
`isinstance()` (*iree.compiler.ir.DictAttr* static method), 81  
`isinstance()` (*iree.compiler.ir.F16Type* static method), 82  
`isinstance()` (*iree.compiler.ir.F32Type* static method), 82  
`isinstance()` (*iree.compiler.ir.F64Type* static method), 82  
`isinstance()` (*iree.compiler.ir.FlatSymbolRefAttr* static method), 82  
`isinstance()` (*iree.compiler.ir.Float8E4M3B11FNUZType* static method), 82  
`isinstance()` (*iree.compiler.ir.Float8E4M3FNType* static method), 82  
`isinstance()` (*iree.compiler.ir.Float8E4M3FNUZType* static method), 82  
`isinstance()` (*iree.compiler.ir.Float8E5M2FNUZType* static method), 82  
`isinstance()` (*iree.compiler.ir.Float8E5M2Type* static method), 83  
`isinstance()` (*iree.compiler.ir.FloatAttr* static method), 83  
`isinstance()` (*iree.compiler.ir.FunctionType* static method), 83  
`isinstance()` (*iree.compiler.ir.IndexType* static method), 83  
`isinstance()` (*iree.compiler.ir.IntegerAttr* static method), 84  
`isinstance()` (*iree.compiler.ir.IntegerType* static method), 85  
`isinstance()` (*iree.compiler.ir.MemRefType* static method), 86  
`isinstance()` (*iree.compiler.ir.NoneType* static method), 86

[method](#)), 87  
[isinstance\(\)](#) ([iree.compiler.ir.OpaqueAttr](#) static method), 88  
[isinstance\(\)](#) ([iree.compiler.ir.OpaqueType](#) static method), 88  
[isinstance\(\)](#) ([iree.compiler.ir.OpResult](#) static method), 87  
[isinstance\(\)](#) ([iree.compiler.ir.RankedTensorType](#) static method), 89  
[isinstance\(\)](#) ([iree.compiler.ir.ShapedType](#) static method), 90  
[isinstance\(\)](#) ([iree.compiler.ir.StridedLayoutAttr](#) static method), 90  
[isinstance\(\)](#) ([iree.compiler.ir.StringAttr](#) static method), 90  
[isinstance\(\)](#) ([iree.compiler.ir.TupleType](#) static method), 91  
[isinstance\(\)](#) ([iree.compiler.ir.TypeAttr](#) static method), 91  
[isinstance\(\)](#) ([iree.compiler.ir.UnitAttr](#) static method), 91  
[isinstance\(\)](#) ([iree.compiler.ir.UnrankedMemRefType](#) static method), 92  
[isinstance\(\)](#) ([iree.compiler.ir.UnrankedTensorType](#) static method), 92  
[isinstance\(\)](#) ([iree.compiler.ir.VectorType](#) static method), 92  
[isWrite](#) ([iree.compiler.dialects.memref.PrefetchOp](#) property), 40  
[iterator\\_types](#) ([iree.compiler.dialects.linalg.LinalgStructuredOp](#) property), 22

## K

[kind](#) ([iree.compiler.dialects.linalg.OperandDefConfig](#) property), 24

## L

[laneid](#) ([iree.compiler.dialects.vector.WarpExecuteOnLaneOp](#) property), 71  
[layout](#) ([iree.compiler.ir.MemRefType](#) property), 86  
[lhs](#) ([iree.compiler.dialects.math.Atan2Op](#) property), 32  
[lhs](#) ([iree.compiler.dialects.math.CopySignOp](#) property), 32  
[lhs](#) ([iree.compiler.dialects.math.FPowIOp](#) property), 34  
[lhs](#) ([iree.compiler.dialects.math.IPowIOp](#) property), 34  
[lhs](#) ([iree.compiler.dialects.math.PowFOp](#) property), 35  
[lhs](#) ([iree.compiler.dialects.shape.AddOp](#) property), 42  
[lhs](#) ([iree.compiler.dialects.shape.ConcatOp](#) property), 43  
[lhs](#) ([iree.compiler.dialects.shape.DivOp](#) property), 44  
[lhs](#) ([iree.compiler.dialects.shape.MaxOp](#) property), 45  
[lhs](#) ([iree.compiler.dialects.shape.MinOp](#) property), 46  
[lhs](#) ([iree.compiler.dialects.shape.MulOp](#) property), 46  
[lhs](#) ([iree.compiler.dialects.vector.ContractionOp](#) property), 65  
[lhs](#) ([iree.compiler.dialects.vector.FMAOp](#) property), 66  
[lhs](#) ([iree.compiler.dialects.vector.MatmulOp](#) property), 68  
[lhs](#) ([iree.compiler.dialects.vector.OuterProductOp](#) property), 68  
[lhs](#) ([iree.compiler.ir.AffineBinaryExpr](#) property), 72  
[lhs\\_columns](#) ([iree.compiler.dialects.vector.MatmulOp](#) property), 68  
[lhs\\_rows](#) ([iree.compiler.dialects.vector.MatmulOp](#) property), 68  
[library\\_call](#) ([iree.compiler.dialects.linalg.GenericOp](#) property), 21  
[linalg\\_structured\\_op\(\)](#) (in [module iree.compiler.dialects.linalg](#)), 31  
[LinalgOpConfig](#) (class in [iree.compiler.dialects.linalg](#)), 21  
[LinalgOpDef](#) (class in [iree.compiler.dialects.linalg](#)), 22  
[LinalgStructuredOpConfig](#) (class in [iree.compiler.dialects.linalg](#)), 22  
[Linkage](#) (class in [iree.runtime](#)), 6  
[linkage](#) ([iree.runtime.VmFunction](#) property), 8  
[load\\_vm\\_flatbuffer\(\)](#) (in [module iree.runtime](#)), 10  
[load\\_vm\\_flatbuffer\\_file\(\)](#) (in [module iree.runtime](#)), 10  
[load\\_vm\\_module\(\)](#) (in [module iree.runtime](#)), 10  
[load\\_vm\\_modules\(\)](#) (in [module iree.runtime](#)), 10  
[LoadOp](#) (class in [iree.compiler.dialects.memref](#)), 40  
[LoadOpConfig](#) (class in [iree.compiler.dialects.vector](#)), 67  
[local\\_dim\\_count](#) ([iree.compiler.dialects.linalg.AffineBuildState](#) property), 18  
[local\\_symbol\\_count](#) ([iree.compiler.dialects.linalg.AffineBuildState](#) property), 18  
[localityHint](#) ([iree.compiler.dialects.memref.PrefetchOp](#) property), 40  
[Location](#) (class in [iree.compiler.ir](#)), 85  
[location](#) ([iree.compiler.ir.Diagnostic](#) property), 80  
[location](#) ([iree.compiler.ir.DiagnosticInfo](#) property), 81  
[log](#) ([iree.compiler.dialects.linalg.UnaryFn](#) attribute), 29  
[Log10Op](#) (class in [iree.compiler.dialects.math](#)), 34  
[Log1pOp](#) (class in [iree.compiler.dialects.math](#)), 34  
[Log2Op](#) (class in [iree.compiler.dialects.math](#)), 35  
[LogicalAndOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[LogicalLeftShiftOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[LogicalNotOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[LogicalOrOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[LogicalRightShiftOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[LogicalXorOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[LogOp](#) (class in [iree.compiler.dialects.math](#)), 35  
[LogOp](#) (class in [iree.compiler.dialects.tosa](#)), 58  
[lookup\\_function\(\)](#) ([iree.runtime.VmModule](#) method),



9  
low (*iree.compiler.dialects.tensor.PadOp* property), 51

## M

map() (*iree.runtime.HalBufferView* method), 4  
map\_to\_dtype() (*iree.runtime.HalElementType* static method), 6  
MapOp (class in *iree.compiler.dialects.linalg*), 22  
mapper (*iree.compiler.dialects.linalg.MapOp* property), 23  
MAPPING (*iree.runtime.BufferUsage* attribute), 2  
MAPPING\_ACCESS\_RANDOM (*iree.runtime.BufferUsage* attribute), 2  
MAPPING\_ACCESS\_SEQUENTIAL\_WRITE (*iree.runtime.BufferUsage* attribute), 2  
MAPPING\_OPTIONAL (*iree.runtime.BufferUsage* attribute), 2  
MAPPING\_PERSISTENT (*iree.runtime.BufferUsage* attribute), 2  
MAPPING\_SCOPED (*iree.runtime.BufferUsage* attribute), 2  
mask (*iree.compiler.dialects.vector.CompressStoreOp* property), 64  
mask (*iree.compiler.dialects.vector.ExpandLoadOp* property), 65  
mask (*iree.compiler.dialects.vector.GatherOp* property), 66  
mask (*iree.compiler.dialects.vector.MaskedLoadOp* property), 67  
mask (*iree.compiler.dialects.vector.MaskedStoreOp* property), 68  
mask (*iree.compiler.dialects.vector.MaskOp* property), 67  
mask (*iree.compiler.dialects.vector.ScatterOp* property), 69  
mask (*iree.compiler.dialects.vector.TransferReadOp* property), 70  
mask (*iree.compiler.dialects.vector.TransferWriteOp* property), 71  
MaskedLoadOp (class in *iree.compiler.dialects.vector*), 67  
MaskedStoreOp (class in *iree.compiler.dialects.vector*), 67  
MaskOp (class in *iree.compiler.dialects.vector*), 67  
maskRegion (*iree.compiler.dialects.vector.MaskOp* property), 67  
masks (*iree.compiler.dialects.vector.ContractionOp* property), 65  
MatmulOp (class in *iree.compiler.dialects.linalg*), 23  
MatMulOp (class in *iree.compiler.dialects.tosa*), 59  
MatmulOp (class in *iree.compiler.dialects.vector*), 68  
MatmulTransposeBOp (class in *iree.compiler.dialects.linalg*), 23  
MatmulUnsignedOp (class in *iree.compiler.dialects.linalg*), 23

matrix (*iree.compiler.dialects.vector.FlatTransposeOp* property), 66  
MatvecOp (class in *iree.compiler.dialects.linalg*), 23  
max\_fp (*iree.compiler.dialects.tosa.ClampOp* property), 54  
max\_int (*iree.compiler.dialects.tosa.ClampOp* property), 54  
max\_signed (*iree.compiler.dialects.linalg.BinaryFn* attribute), 18  
max\_signed (*iree.compiler.dialects.linalg.ReduceFn* attribute), 25  
max\_unsigned (*iree.compiler.dialects.linalg.BinaryFn* attribute), 18  
max\_unsigned (*iree.compiler.dialects.linalg.ReduceFn* attribute), 25  
MaximumOp (class in *iree.compiler.dialects.tosa*), 59  
MaxOp (class in *iree.compiler.dialects.shape*), 45  
MaxPool2dOp (class in *iree.compiler.dialects.tosa*), 59  
MeetOp (class in *iree.compiler.dialects.shape*), 46  
memory\_space (*iree.compiler.ir.MemRefType* property), 86  
memory\_space (*iree.compiler.ir.UnrankedMemRefType* property), 92  
MemoryAccess (class in *iree.runtime*), 6  
MemorySpaceCastOp (class in *iree.compiler.dialects.memref*), 40  
MemoryType (class in *iree.runtime*), 7  
memref (*iree.compiler.dialects.memref.AllocOp* property), 37  
memref (*iree.compiler.dialects.memref.AllocOp* property), 37  
memref (*iree.compiler.dialects.memref.AssumeAlignmentOp* property), 37  
memref (*iree.compiler.dialects.memref.AtomicRMWOp* property), 37  
memref (*iree.compiler.dialects.memref.DeallocOp* property), 38  
memref (*iree.compiler.dialects.memref.GenericAtomicRMWOp* property), 39  
memref (*iree.compiler.dialects.memref.PrefetchOp* property), 40  
memref (*iree.compiler.dialects.memref.RankOp* property), 40  
memref (*iree.compiler.dialects.memref.StoreOp* property), 41  
memref (*iree.compiler.dialects.memref.TensorStoreOp* property), 41  
memref (*iree.compiler.dialects.vector.TypeCastOp* property), 71  
MemRefType (class in *iree.compiler.ir*), 86  
message (*iree.compiler.ir.Diagnostic* property), 80  
message (*iree.compiler.ir.DiagnosticInfo* property), 81  
MHLO (*iree.compiler.tools.InputType* attribute), 13  
min\_fp (*iree.compiler.dialects.tosa.ClampOp* property),

54  
 min\_int (*iree.compiler.dialects.tosa.ClampOp* property), 54  
 min\_signed (*iree.compiler.dialects.linalg.BinaryFn* attribute), 18  
 min\_signed (*iree.compiler.dialects.linalg.ReduceFn* attribute), 25  
 min\_unsigned (*iree.compiler.dialects.linalg.BinaryFn* attribute), 18  
 min\_unsigned (*iree.compiler.dialects.linalg.ReduceFn* attribute), 25  
 MinimumOp (class in *iree.compiler.dialects.tosa*), 59  
 MinOp (class in *iree.compiler.dialects.shape*), 46  
 MLIR\_TEXT (*iree.compiler.tools.OutputFormat* attribute), 13  
 MLIRError, 86  
 Mmt4DOp (class in *iree.compiler.dialects.linalg*), 23  
 mode (*iree.compiler.dialects.tosa.ResizeOp* property), 62  
 module  
   *iree.compiler.dialects.builtin*, 17  
   *iree.compiler.dialects.linalg*, 18  
   *iree.compiler.dialects.math*, 31  
   *iree.compiler.dialects.memref*, 37  
   *iree.compiler.dialects.shape*, 42  
   *iree.compiler.dialects.tensor*, 48  
   *iree.compiler.dialects.tosa*, 52  
   *iree.compiler.dialects.vector*, 64  
   *iree.compiler.ir*, 72  
   *iree.compiler.passmanager*, 92  
   *iree.compiler.tools*, 11  
   *iree.compiler.tools.debugging*, 13  
   *iree.compiler.tools.tf*, 16  
   *iree.compiler.tools.tflite*, 15  
   *iree.runtime*, 1  
 Module (class in *iree.compiler.ir*), 86  
 module\_name (*iree.runtime.VmFunction* property), 8  
 ModuleOp (class in *iree.compiler.dialects.builtin*), 17  
 modules (*iree.runtime.SystemContext* property), 8  
 msg (*iree.compiler.dialects.shape.CstrRequireOp* property), 44  
 mul (*iree.compiler.dialects.linalg.BinaryFn* attribute), 19  
 mul (*iree.compiler.dialects.linalg.ReduceFn* attribute), 25  
 MulOp (class in *iree.compiler.dialects.shape*), 46  
 MulOp (class in *iree.compiler.dialects.tosa*), 59  
 MultiDimReductionOp (class in *iree.compiler.dialects.vector*), 68  
 multiplier (*iree.compiler.dialects.tosa.ApplyScaleOp* property), 53

## N

n\_dims (*iree.compiler.ir.AffineMap* property), 75  
 n\_dims (*iree.compiler.ir.IntegerSet* property), 84  
 n\_equalities (*iree.compiler.ir.IntegerSet* property), 84

n\_inequalities (*iree.compiler.ir.IntegerSet* property), 84  
 n\_inputs (*iree.compiler.ir.AffineMap* property), 75  
 n\_inputs (*iree.compiler.ir.IntegerSet* property), 84  
 n\_symbols (*iree.compiler.ir.AffineMap* property), 75  
 n\_symbols (*iree.compiler.ir.IntegerSet* property), 85  
 name (*iree.compiler.dialects.linalg.Enum* attribute), 21  
 name (*iree.compiler.dialects.linalg.OperandDefConfig* property), 24  
 name (*iree.compiler.ir.DiagnosticSeverity* property), 81  
 name (*iree.compiler.ir.NamedAttribute* property), 87  
 name (*iree.compiler.ir.Operation* property), 89  
 name (*iree.runtime.BufferCompatibility* property), 1  
 name (*iree.runtime.BufferUsage* property), 3  
 name (*iree.runtime.HalElementType* property), 6  
 name (*iree.runtime.Linkage* property), 6  
 name (*iree.runtime.MemoryAccess* property), 7  
 name (*iree.runtime.MemoryType* property), 7  
 name (*iree.runtime.VmFunction* property), 8  
 name (*iree.runtime.VmModule* property), 9  
 name() (*iree.compiler.ir.Location* static method), 86  
 NamedAttribute (class in *iree.compiler.ir*), 87  
 namespace (*iree.compiler.ir.DialectDescriptor* property), 81  
 NegateOp (class in *iree.compiler.dialects.tosa*), 59  
 negf (*iree.compiler.dialects.linalg.UnaryFn* attribute), 29  
 nofold (*iree.compiler.dialects.tensor.PadOp* property), 51  
 NONE (*iree.compiler.tools.InputType* attribute), 13  
 NONE (*iree.runtime.BufferCompatibility* attribute), 1  
 NONE (*iree.runtime.BufferUsage* attribute), 2  
 NONE (*iree.runtime.HalElementType* attribute), 6  
 NONE (*iree.runtime.MemoryAccess* attribute), 7  
 NONE (*iree.runtime.MemoryType* attribute), 7  
 NoneType (class in *iree.compiler.ir*), 87  
 nontemporal (*iree.compiler.dialects.memref.StoreOp* property), 41  
 normalize\_value() (in module *iree.runtime*), 10  
 NOTE (*iree.compiler.ir.DiagnosticSeverity* attribute), 81  
 notes (*iree.compiler.ir.Diagnostic* property), 80  
 notes (*iree.compiler.ir.DiagnosticInfo* property), 81  
 num\_types (*iree.compiler.ir.TupleType* property), 91  
 numElements (*iree.compiler.dialects.memref.DmaWaitOp* property), 39  
 NumElementsOp (class in *iree.compiler.dialects.shape*), 46

## O

OBJECT\_GRAPH (*iree.compiler.tools.tf.ImportType* attribute), 17  
 offset (*iree.compiler.dialects.memref.ExtractStridedMetadataOp* property), 39  
 offset (*iree.compiler.ir.StridedLayoutAttr* property), 90

[offsets \(iree.compiler.dialects.memref.ReinterpretCastOp property\), 41](#)  
[offsets \(iree.compiler.dialects.memref.SubViewOp property\), 41](#)  
[offsets \(iree.compiler.dialects.tensor.ExtractSliceOp property\), 49](#)  
[offsets \(iree.compiler.dialects.tensor.InsertSliceOp property\), 50](#)  
[offsets \(iree.compiler.dialects.tensor.ParallelInsertSliceOp property\), 51](#)  
[on\\_false \(iree.compiler.dialects.tosa.SelectOp property\), 62](#)  
[on\\_true \(iree.compiler.dialects.tosa.SelectOp property\), 62](#)  
[OPAQUE\\_16 \(iree.runtime.HalElementType attribute\), 6](#)  
[OPAQUE\\_32 \(iree.runtime.HalElementType attribute\), 6](#)  
[OPAQUE\\_64 \(iree.runtime.HalElementType attribute\), 6](#)  
[OPAQUE\\_8 \(iree.runtime.HalElementType attribute\), 6](#)  
[OpaqueAttr \(class in iree.compiler.ir\), 87](#)  
[OpaqueType \(class in iree.compiler.ir\), 88](#)  
[OpAttributeMap \(class in iree.compiler.ir\), 87](#)  
[OpDefinitionDef \(class in iree.compiler.dialects.linalg\), 23](#)  
[operand \(iree.compiler.dialects.math.AbsFOp property\), 31](#)  
[operand \(iree.compiler.dialects.math.AbsIOp property\), 32](#)  
[operand \(iree.compiler.dialects.math.AtanOp property\), 32](#)  
[operand \(iree.compiler.dialects.math.CbrtOp property\), 32](#)  
[operand \(iree.compiler.dialects.math.CeilOp property\), 32](#)  
[operand \(iree.compiler.dialects.math.CosOp property\), 33](#)  
[operand \(iree.compiler.dialects.math.CountLeadingZerosOp property\), 33](#)  
[operand \(iree.compiler.dialects.math.CountTrailingZerosOp property\), 33](#)  
[operand \(iree.compiler.dialects.math.CtPopOp property\), 33](#)  
[operand \(iree.compiler.dialects.math.ErfOp property\), 33](#)  
[operand \(iree.compiler.dialects.math.Exp2Op property\), 33](#)  
[operand \(iree.compiler.dialects.math.ExpM1Op property\), 33](#)  
[operand \(iree.compiler.dialects.math.ExpOp property\), 34](#)  
[operand \(iree.compiler.dialects.math.FloorOp property\), 34](#)  
[operand \(iree.compiler.dialects.math.Log10Op property\), 34](#)  
[operand \(iree.compiler.dialects.math.Log1pOp property\), 35](#)  
[operand \(iree.compiler.dialects.math.Log2Op property\), 35](#)  
[operand \(iree.compiler.dialects.math.LogOp property\), 35](#)  
[operand \(iree.compiler.dialects.math.RoundEvenOp property\), 35](#)  
[operand \(iree.compiler.dialects.math.RoundOp property\), 35](#)  
[operand \(iree.compiler.dialects.math.RsqrtOp property\), 35](#)  
[operand \(iree.compiler.dialects.math.SinOp property\), 36](#)  
[operand \(iree.compiler.dialects.math.SqrtOp property\), 36](#)  
[operand \(iree.compiler.dialects.math.TanhOp property\), 36](#)  
[operand \(iree.compiler.dialects.math.TanOp property\), 36](#)  
[operand \(iree.compiler.dialects.math.TruncOp property\), 36](#)  
[operand \(iree.compiler.dialects.shape.SplitAtOp property\), 47](#)  
[operand \(iree.compiler.dialects.shape.WithOp property\), 48](#)  
[operand\\_number \(iree.compiler.ir.OpOperand property\), 87](#)  
[OperandDef \(class in iree.compiler.dialects.linalg\), 23](#)  
[OperandDefConfig \(class in iree.compiler.dialects.linalg\), 23](#)  
[operands\\_ \(iree.compiler.dialects.memref.DmaStartOp property\), 38](#)  
[operands\\_ \(iree.compiler.dialects.shape.AssumingYieldOp property\), 43](#)  
[operands\\_ \(iree.compiler.dialects.shape.ReturnOp property\), 47](#)  
[operands\\_ \(iree.compiler.dialects.shape.YieldOp property\), 48](#)  
[operands\\_ \(iree.compiler.dialects.vector.CreateMaskOp property\), 65](#)  
[operands\\_ \(iree.compiler.dialects.vector.YieldOp property\), 71](#)  
[Operation \(class in iree.compiler.ir\), 88](#)  
[operation \(iree.compiler.ir.InferTypeOpInterface property\), 83](#)  
[operation \(iree.compiler.ir.Module property\), 86](#)  
[operation \(iree.compiler.ir.OpView property\), 87](#)  
[OPERATION\\_NAME \(iree.compiler.dialects.builtin.UnrealizedConversionCast attribute\), 17](#)  
[OPERATION\\_NAME \(iree.compiler.dialects.linalg.BroadcastOp attribute\), 19](#)  
[OPERATION\\_NAME \(iree.compiler.dialects.linalg.GenericOp attribute\), 21](#)  
[OPERATION\\_NAME \(iree.compiler.dialects.linalg.IndexOp](#)

<i>attribute</i> ), 21	<i>attribute</i> ), 35
OPERATION_NAME ( <i>iree.compiler.dialects.linalg.MapOp</i> <i>attribute</i> ), 23	OPERATION_NAME ( <i>iree.compiler.dialects.math.PowFOp</i> <i>attribute</i> ), 35
OPERATION_NAME ( <i>iree.compiler.dialects.linalg.ReduceOp</i> <i>attribute</i> ), 25	OPERATION_NAME ( <i>iree.compiler.dialects.math.RoundEvenOp</i> <i>attribute</i> ), 35
OPERATION_NAME ( <i>iree.compiler.dialects.linalg.TransposeOp</i> <i>attribute</i> ), 28	OPERATION_NAME ( <i>iree.compiler.dialects.math.RoundOp</i> <i>attribute</i> ), 35
OPERATION_NAME ( <i>iree.compiler.dialects.linalg.YieldOp</i> <i>attribute</i> ), 30	OPERATION_NAME ( <i>iree.compiler.dialects.math.RsqrtOp</i> <i>attribute</i> ), 35
OPERATION_NAME ( <i>iree.compiler.dialects.math.AbsFOp</i> <i>attribute</i> ), 31	OPERATION_NAME ( <i>iree.compiler.dialects.math.SinOp</i> <i>attribute</i> ), 36
OPERATION_NAME ( <i>iree.compiler.dialects.math.AbsIOp</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.math.SqrtOp</i> <i>attribute</i> ), 36
OPERATION_NAME ( <i>iree.compiler.dialects.math.Atan2Op</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.math.TanhOp</i> <i>attribute</i> ), 36
OPERATION_NAME ( <i>iree.compiler.dialects.math.AtanOp</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.math.TanOp</i> <i>attribute</i> ), 36
OPERATION_NAME ( <i>iree.compiler.dialects.math.CbrtOp</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.math.TruncOp</i> <i>attribute</i> ), 36
OPERATION_NAME ( <i>iree.compiler.dialects.math.CeilOp</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AllocOp</i> <i>attribute</i> ), 37
OPERATION_NAME ( <i>iree.compiler.dialects.math.CopySignOp</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AllocScopeOp</i> <i>attribute</i> ), 37
OPERATION_NAME ( <i>iree.compiler.dialects.math.CosOp</i> <i>attribute</i> ), 32	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AllocScopeReturnOp</i> <i>attribute</i> ), 37
OPERATION_NAME ( <i>iree.compiler.dialects.math.CountLeadingZerosOp</i> <i>attribute</i> ), 33	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AllocOp</i> <i>attribute</i> ), 37
OPERATION_NAME ( <i>iree.compiler.dialects.math.CountTrailingZerosOp</i> <i>attribute</i> ), 33	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AssumeAlignmentOp</i> <i>attribute</i> ), 37
OPERATION_NAME ( <i>iree.compiler.dialects.math.CtPopOp</i> <i>attribute</i> ), 33	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AtomicRMWOp</i> <i>attribute</i> ), 37
OPERATION_NAME ( <i>iree.compiler.dialects.math.ErfOp</i> <i>attribute</i> ), 33	OPERATION_NAME ( <i>iree.compiler.dialects.memref.AtomicYieldOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.Exp2Op</i> <i>attribute</i> ), 33	OPERATION_NAME ( <i>iree.compiler.dialects.memref.CastOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.ExpM1Op</i> <i>attribute</i> ), 33	OPERATION_NAME ( <i>iree.compiler.dialects.memref.CollapseShapeOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.ExpOp</i> <i>attribute</i> ), 34	OPERATION_NAME ( <i>iree.compiler.dialects.memref.CopyOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.FloorOp</i> <i>attribute</i> ), 34	OPERATION_NAME ( <i>iree.compiler.dialects.memref.DeallocOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.FmaOp</i> <i>attribute</i> ), 34	OPERATION_NAME ( <i>iree.compiler.dialects.memref.DimOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.FPowIOp</i> <i>attribute</i> ), 34	OPERATION_NAME ( <i>iree.compiler.dialects.memref.DmaStartOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.IPowIOp</i> <i>attribute</i> ), 34	OPERATION_NAME ( <i>iree.compiler.dialects.memref.DmaWaitOp</i> <i>attribute</i> ), 38
OPERATION_NAME ( <i>iree.compiler.dialects.math.Log10Op</i> <i>attribute</i> ), 34	OPERATION_NAME ( <i>iree.compiler.dialects.memref.ExpandShapeOp</i> <i>attribute</i> ), 39
OPERATION_NAME ( <i>iree.compiler.dialects.math.Log1pOp</i> <i>attribute</i> ), 35	OPERATION_NAME ( <i>iree.compiler.dialects.memref.ExtractAlignedPointerAsIntPtrOp</i> <i>attribute</i> ), 39
OPERATION_NAME ( <i>iree.compiler.dialects.math.Log2Op</i> <i>attribute</i> ), 35	OPERATION_NAME ( <i>iree.compiler.dialects.memref.ExtractStridedMetadataOp</i> <i>attribute</i> ), 39
OPERATION_NAME ( <i>iree.compiler.dialects.math.LogOp</i> <i>attribute</i> ), 35	OPERATION_NAME ( <i>iree.compiler.dialects.memref.GenericAtomicRMWOp</i> <i>attribute</i> ), 39



attribute), 39

OPERATION\_NAME (iree.compiler.dialects.memref.GetGlobalOp attribute), 39

OPERATION\_NAME (iree.compiler.dialects.memref.GlobalOp attribute), 40

OPERATION\_NAME (iree.compiler.dialects.memref.MemorySpace attribute), 40

OPERATION\_NAME (iree.compiler.dialects.memref.PrefetchOp attribute), 40

OPERATION\_NAME (iree.compiler.dialects.memref.RankOp attribute), 40

OPERATION\_NAME (iree.compiler.dialects.memref.ReallocOp attribute), 40

OPERATION\_NAME (iree.compiler.dialects.memref.ReinterpretOp attribute), 41

OPERATION\_NAME (iree.compiler.dialects.memref.ReshapeOp attribute), 41

OPERATION\_NAME (iree.compiler.dialects.memref.StoreOp attribute), 41

OPERATION\_NAME (iree.compiler.dialects.memref.SubViewOp attribute), 41

OPERATION\_NAME (iree.compiler.dialects.memref.TensorStoreOp attribute), 41

OPERATION\_NAME (iree.compiler.dialects.memref.TransposeOp attribute), 42

OPERATION\_NAME (iree.compiler.dialects.memref.ViewOp attribute), 42

OPERATION\_NAME (iree.compiler.dialects.shape.AddOp attribute), 42

OPERATION\_NAME (iree.compiler.dialects.shape.AnyOp attribute), 42

OPERATION\_NAME (iree.compiler.dialects.shape.AssumingAlignment attribute), 42

OPERATION\_NAME (iree.compiler.dialects.shape.AssumingOp attribute), 42

OPERATION\_NAME (iree.compiler.dialects.shape.AssumingYield attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.BroadcastOp attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.ConcatOp attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.ConstShapeOp attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.ConstSizeOp attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.ConstWitnessOp attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.CstrBroadcastOp attribute), 43

OPERATION\_NAME (iree.compiler.dialects.shape.CstrEqOp attribute), 44

OPERATION\_NAME (iree.compiler.dialects.shape.CstrRequired attribute), 44

OPERATION\_NAME (iree.compiler.dialects.shape.DebugPrintOp attribute), 44

attribute), 44

OPERATION\_NAME (iree.compiler.dialects.shape.DimOp attribute), 44

OPERATION\_NAME (iree.compiler.dialects.shape.DivOp attribute), 44

OPERATION\_NAME (iree.compiler.dialects.shape.FromExtentsOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.FromExtentTensorOp attribute), 44

OPERATION\_NAME (iree.compiler.dialects.shape.FuncOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.FunctionLibraryOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.GetExtentOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.IndexToSizeOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.IsBroadcastableOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.MaxOp attribute), 45

OPERATION\_NAME (iree.compiler.dialects.shape.MeetOp attribute), 46

OPERATION\_NAME (iree.compiler.dialects.shape.MinOp attribute), 46

OPERATION\_NAME (iree.compiler.dialects.shape.MulOp attribute), 46

OPERATION\_NAME (iree.compiler.dialects.shape.NumElementsOp attribute), 46

OPERATION\_NAME (iree.compiler.dialects.shape.RankOp attribute), 46

OPERATION\_NAME (iree.compiler.dialects.shape.ReduceOp attribute), 46

OPERATION\_NAME (iree.compiler.dialects.shape.ReturnOp attribute), 47

OPERATION\_NAME (iree.compiler.dialects.shape.ShapeEqOp attribute), 47

OPERATION\_NAME (iree.compiler.dialects.shape.ShapeOfOp attribute), 47

OPERATION\_NAME (iree.compiler.dialects.shape.SizeToIndexOp attribute), 47

OPERATION\_NAME (iree.compiler.dialects.shape.SplitAtOp attribute), 47

OPERATION\_NAME (iree.compiler.dialects.shape.ToExtentTensorOp attribute), 47

OPERATION\_NAME (iree.compiler.dialects.shape.ValueAsShapeOp attribute), 48

OPERATION\_NAME (iree.compiler.dialects.shape.ValueOfOp attribute), 48

OPERATION\_NAME (iree.compiler.dialects.shape.WithOp attribute), 48

OPERATION\_NAME (iree.compiler.dialects.shape.YieldOp attribute), 48

OPERATION\_NAME (iree.compiler.dialects.tensor.CastOp attribute), 48

[attribute](#)), 48  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.CollapseShapeOp attribute\)](#), 48  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.DimOp attribute\)](#), 49  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.ExpandShapeOp attribute\)](#), 49  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.ExtractOp attribute\)](#), 49  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.ExtractSliceOp attribute\)](#), 49  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.FromElementsOp attribute\)](#), 49  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.GatherOp attribute\)](#), 50  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.GenerateOp attribute\)](#), 50  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.InsertOp attribute\)](#), 50  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.InsertSliceOp attribute\)](#), 50  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.PackOp attribute\)](#), 50  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.PadOp attribute\)](#), 51  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.ParallelInsertOp attribute\)](#), 51  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.RankOp attribute\)](#), 51  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.ReshapeOp attribute\)](#), 51  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.ScatterOp attribute\)](#), 52  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.SplatOp attribute\)](#), 52  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.UnPackOp attribute\)](#), 52  
[OPERATION\\_NAME \(iree.compiler.dialects.tensor.YieldOp attribute\)](#), 52  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.AbsOp attribute\)](#), 52  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.AddOp attribute\)](#), 52  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ApplyScaleOp attribute\)](#), 53  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ArgMaxOp attribute\)](#), 53  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ArithmeticRightShiftOp attribute\)](#), 53  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.AvgPool2dOp attribute\)](#), 53  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.BitwiseAndOp attribute\)](#), 53  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.BitwiseNotOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.BitwiseOrOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.BitwiseXorOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.CastOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.CeilOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ClampOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ClzOp attribute\)](#), 54  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ConcatOp attribute\)](#), 55  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ConstOp attribute\)](#), 55  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.Conv2DOp attribute\)](#), 55  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.Conv3DOp attribute\)](#), 55  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.CustomOp attribute\)](#), 55  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.DepthwiseConv2DOp attribute\)](#), 56  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.DivOp attribute\)](#), 56  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.EqualOp attribute\)](#), 56  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.ExpOp attribute\)](#), 56  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.FFT2dOp attribute\)](#), 56  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.FloorOp attribute\)](#), 56  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.FullyConnectedOp attribute\)](#), 57  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.GatherOp attribute\)](#), 57  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.GreaterEqualOp attribute\)](#), 57  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.GreaterOp attribute\)](#), 57  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.IdentityOp attribute\)](#), 57  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.IfOp attribute\)](#), 57  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.LogicalAndOp attribute\)](#), 58  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.LogicalLeftShiftOp attribute\)](#), 58  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.LogicalNotOp attribute\)](#), 58  
[OPERATION\\_NAME \(iree.compiler.dialects.tosa.LogicalOrOp attribute\)](#), 58

<a href="#">attribute), 58</a>	<a href="#">attribute), 62</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.LogicalRightShiftOp attribute), 58</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.SliceOp attribute), 63</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.LogicalXorOp attribute), 58</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.SubOp attribute), 63</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.LogOp attribute), 58</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.TableOp attribute), 63</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.MatMulOp attribute), 59</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.TanhOp attribute), 63</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.MaximumOp attribute), 59</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.TileOp attribute), 63</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.MaxPool2dOp attribute), 59</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.TransposeConv2DOp attribute), 63</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.MinimumOp attribute), 59</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.TransposeOp attribute), 64</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.MulOp attribute), 59</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.WhileOp attribute), 64</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.NegateOp attribute), 59</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.YieldOp attribute), 64</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.PadOp attribute), 60</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.BitCastOp attribute), 64</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.PowOp attribute), 60</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.BroadcastOp attribute), 64</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReciprocalOp attribute), 60</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.CompressStoreOp attribute), 64</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReduceAllOp attribute), 60</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.ConstantMaskOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReduceAnyOp attribute), 60</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.ContractionOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReduceMaxOp attribute), 61</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.CreateMaskOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReduceMinOp attribute), 61</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.ExpandLoadOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReduceProdOp attribute), 61</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.ExtractElementOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReduceSumOp attribute), 61</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.ExtractOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.RescaleOp attribute), 61</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.ExtractStridedSliceOp attribute), 65</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReshapeOp attribute), 61</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.FlatTransposeOp attribute), 66</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ResizeOp attribute), 62</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.FMAOp attribute), 66</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ReverseOp attribute), 62</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.GatherOp attribute), 66</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.RFFT2dOp attribute), 60</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.InsertElementOp attribute), 66</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.RsqrtOp attribute), 62</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.InsertOp attribute), 66</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.ScatterOp attribute), 62</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.InsertStridedSliceOp attribute), 67</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.SelectOp attribute), 62</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.LoadOp attribute), 67</a>
<a href="#">OPERATION_NAME (iree.compiler.dialects.tosa.SigmoidOp attribute), 62</a>	<a href="#">OPERATION_NAME (iree.compiler.dialects.vector.MaskedLoadOp attribute), 67</a>

[attribute](#)), 67  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.MaskedStoreOp attribute\)](#)), 67  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.MaskOp attribute\)](#)), 67  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.MatmulOp attribute\)](#)), 68  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.MultiDimReduceOp attribute\)](#)), 68  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.OuterProductOp attribute\)](#)), 68  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.PrintOp attribute\)](#)), 68  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ReductionOp attribute\)](#)), 68  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ReshapeOp attribute\)](#)), 69  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ScalableExpandOp attribute\)](#)), 69  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ScalableInsertOp attribute\)](#)), 69  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ScanOp attribute\)](#)), 69  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ScatterOp attribute\)](#)), 69  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ShapeCastOp attribute\)](#)), 70  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.ShuffleOp attribute\)](#)), 70  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.SplatOp attribute\)](#)), 70  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.StoreOp attribute\)](#)), 70  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.TransferReadOp attribute\)](#)), 70  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.TransferWriteOp attribute\)](#)), 70  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.TransposeOp attribute\)](#)), 71  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.TypeCastOp attribute\)](#)), 71  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.VectorScaleOp attribute\)](#)), 71  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.WarpExecuteOnLaneOp attribute\)](#)), 71  
[OPERATION\\_NAME \(iree.compiler.dialects.vector.YieldOp attribute\)](#)), 71  
[OperationIterator \(class in iree.compiler.ir\)](#), 89  
[OperationList \(class in iree.compiler.ir\)](#), 89  
[operations \(iree.compiler.ir.Block property\)](#), 77  
[OpInterfaceDef \(class in iree.compiler.dialects.linalg\)](#), 23  
[OpMetadataDef \(class in iree.compiler.dialects.linalg\)](#), 23  
[OpOperand \(class in iree.compiler.ir\)](#), 87  
[OpOperandIterator \(class in iree.compiler.ir\)](#), 87  
[OpOperandList \(class in iree.compiler.ir\)](#), 87  
[OpResult \(class in iree.compiler.ir\)](#), 87  
[OpResultList \(class in iree.compiler.ir\)](#), 87  
[OPTIMAL \(iree.runtime.MemoryType attribute\)](#), 7  
[OpView \(class in iree.compiler.ir\)](#), 87  
[OpViewOp \(iree.compiler.ir.InferTypeOpInterface property\)](#), 83  
[OpViewOp \(iree.compiler.ir.Operation property\)](#), 89  
[ordered\\_dims \(iree.compiler.dialects.linalg.LinalgStructuredOpConfig property\)](#), 22  
[ordered\\_operands \(iree.compiler.dialects.linalg.LinalgStructuredOpConfig property\)](#), 22  
[ordinal \(iree.runtime.VmFunction property\)](#), 8  
[OuterProductOp \(class in iree.compiler.dialects.vector\)](#), 68  
[output \(iree.compiler.dialects.shape.DebugPrintOp property\)](#), 44  
[output \(iree.compiler.dialects.tosa.AbsOp property\)](#), 52  
[output \(iree.compiler.dialects.tosa.AddOp property\)](#), 53  
[output \(iree.compiler.dialects.tosa.ApplyScaleOp property\)](#), 53  
[output \(iree.compiler.dialects.tosa.ArgMaxOp property\)](#), 53  
[output \(iree.compiler.dialects.tosa.ArithmeticRightShiftOp property\)](#), 53  
[output \(iree.compiler.dialects.tosa.AvgPool2dOp property\)](#), 53  
[output \(iree.compiler.dialects.tosa.BitwiseAndOp property\)](#), 53  
[output \(iree.compiler.dialects.tosa.BitwiseNotOp property\)](#), 54  
[output \(iree.compiler.dialects.tosa.BitwiseOrOp property\)](#), 54  
[output \(iree.compiler.dialects.tosa.BitwiseXorOp property\)](#), 54  
[output \(iree.compiler.dialects.tosa.CastOp property\)](#), 54  
[output \(iree.compiler.dialects.tosa.CeilOp property\)](#), 54  
[output \(iree.compiler.dialects.tosa.ClampOp property\)](#), 54  
[output \(iree.compiler.dialects.tosa.ClzOp property\)](#), 55  
[output \(iree.compiler.dialects.tosa.ConcatOp property\)](#), 55  
[output \(iree.compiler.dialects.tosa.ConstOp property\)](#), 55  
[output \(iree.compiler.dialects.tosa.Conv2DOp property\)](#), 55  
[output \(iree.compiler.dialects.tosa.Conv3DOp property\)](#), 55  
[output \(iree.compiler.dialects.tosa.DepthwiseConv2DOp property\)](#), 56  
[output \(iree.compiler.dialects.tosa.DivOp property\)](#), 56  
[output \(iree.compiler.dialects.tosa.EqualOp property\)](#), 56



56  
 output (*iree.compiler.dialects.tosa.ExpOp* property), 56  
 output (*iree.compiler.dialects.tosa.FloorOp* property), 56  
 output (*iree.compiler.dialects.tosa.FullyConnectedOp* property), 57  
 output (*iree.compiler.dialects.tosa.GatherOp* property), 57  
 output (*iree.compiler.dialects.tosa.GreaterEqualOp* property), 57  
 output (*iree.compiler.dialects.tosa.GreaterOp* property), 57  
 output (*iree.compiler.dialects.tosa.IdentityOp* property), 57  
 output (*iree.compiler.dialects.tosa.IfOp* property), 57  
 output (*iree.compiler.dialects.tosa.LogicalLeftShiftOp* property), 58  
 output (*iree.compiler.dialects.tosa.LogicalNotOp* property), 58  
 output (*iree.compiler.dialects.tosa.LogicalRightShiftOp* property), 58  
 output (*iree.compiler.dialects.tosa.LogOp* property), 58  
 output (*iree.compiler.dialects.tosa.MaximumOp* property), 59  
 output (*iree.compiler.dialects.tosa.MaxPool2dOp* property), 59  
 output (*iree.compiler.dialects.tosa.MinimumOp* property), 59  
 output (*iree.compiler.dialects.tosa.MulOp* property), 59  
 output (*iree.compiler.dialects.tosa.NegateOp* property), 60  
 output (*iree.compiler.dialects.tosa.PadOp* property), 60  
 output (*iree.compiler.dialects.tosa.ReciprocalOp* property), 60  
 output (*iree.compiler.dialects.tosa.ReduceAllOp* property), 60  
 output (*iree.compiler.dialects.tosa.ReduceAnyOp* property), 60  
 output (*iree.compiler.dialects.tosa.ReduceMaxOp* property), 61  
 output (*iree.compiler.dialects.tosa.ReduceMinOp* property), 61  
 output (*iree.compiler.dialects.tosa.ReduceProdOp* property), 61  
 output (*iree.compiler.dialects.tosa.ReduceSumOp* property), 61  
 output (*iree.compiler.dialects.tosa.RescaleOp* property), 61  
 output (*iree.compiler.dialects.tosa.ReshapeOp* property), 62  
 output (*iree.compiler.dialects.tosa.ResizeOp* property), 62  
 output (*iree.compiler.dialects.tosa.ReverseOp* property), 62  
 output (*iree.compiler.dialects.tosa.RsqrtOp* property), 62  
 output (*iree.compiler.dialects.tosa.SelectOp* property), 62  
 output (*iree.compiler.dialects.tosa.SigmoidOp* property), 63  
 output (*iree.compiler.dialects.tosa.SliceOp* property), 63  
 output (*iree.compiler.dialects.tosa.SubOp* property), 63  
 output (*iree.compiler.dialects.tosa.TableOp* property), 63  
 output (*iree.compiler.dialects.tosa.TanhOp* property), 63  
 output (*iree.compiler.dialects.tosa.TileOp* property), 63  
 output (*iree.compiler.dialects.tosa.TransposeConv2DOp* property), 63  
 output (*iree.compiler.dialects.tosa.TransposeOp* property), 64  
 output (*iree.compiler.dialects.tosa.WhileOp* property), 64  
 output\_imag (*iree.compiler.dialects.tosa.FFT2dOp* property), 56  
 output\_imag (*iree.compiler.dialects.tosa.RFFT2dOp* property), 60  
 output\_real (*iree.compiler.dialects.tosa.FFT2dOp* property), 56  
 output\_real (*iree.compiler.dialects.tosa.RFFT2dOp* property), 60  
 output\_shape (*iree.compiler.dialects.vector.ReshapeOp* property), 69  
 OUTPUT\_TENSOR (*iree.compiler.dialects.linalg.OperandKind* attribute), 24  
 output\_zp (*iree.compiler.dialects.tosa.RescaleOp* property), 61  
 outputs (*iree.compiler.dialects.builtin.UnrealizedConversionCastOp* property), 17  
 outputs (*iree.compiler.dialects.linalg.GenericOp* property), 21  
 outputs (*iree.compiler.dialects.tosa.CustomOp* property), 55  
 owner (*iree.compiler.ir.Block* property), 77  
 owner (*iree.compiler.ir.BlockArgument* property), 77  
 owner (*iree.compiler.ir.OpOperand* property), 87  
 owner (*iree.compiler.ir.OpResult* property), 87  
 owner (*iree.compiler.ir.Region* property), 89  
 owner (*iree.compiler.ir.Value* property), 92

## P

PackOp (class in *iree.compiler.dialects.tensor*), 50  
 pad\_const (*iree.compiler.dialects.tosa.PadOp* property), 60  
 padding (*iree.compiler.dialects.tosa.PadOp* property), 60  
 padding (*iree.compiler.dialects.vector.TransferReadOp* property), 70

padding\_value (*iree.compiler.dialects.tensor.PackOp* property), 51

PadOp (class in *iree.compiler.dialects.tensor*), 51

PadOp (class in *iree.compiler.dialects.tosa*), 60

ParallelInsertSliceOp (class in *iree.compiler.dialects.tensor*), 51

parent (*iree.compiler.ir.Operation* property), 89

parse() (*iree.compiler.ir.Attribute* static method), 76

parse() (*iree.compiler.ir.Module* static method), 86

parse() (*iree.compiler.ir.Operation* static method), 89

parse() (*iree.compiler.ir.OpView* class method), 87

parse() (*iree.compiler.ir.Type* static method), 91

parse() (*iree.compiler.passmanager.PassManager* static method), 92

pass\_thru (*iree.compiler.dialects.vector.ExpandLoadOp* property), 65

pass\_thru (*iree.compiler.dialects.vector.GatherOp* property), 66

pass\_thru (*iree.compiler.dialects.vector.MaskedLoadOp* property), 67

passing (*iree.compiler.dialects.shape.ConstWitnessOp* property), 43

PassManager (class in *iree.compiler.passmanager*), 92

passthru (*iree.compiler.dialects.vector.MaskOp* property), 67

per\_channel (*iree.compiler.dialects.tosa.RescaleOp* property), 61

perms (*iree.compiler.dialects.tosa.TransposeOp* property), 64

persist\_vm\_module() (*iree.runtime.Tracer* method), 8

PoolingNchwMaxOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNchwSumOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNcwMaxOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNcwSumOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNdhwMaxOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNdhwMinOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNdhwSumOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNhwcMaxOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNhwcMaxUnsignedOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNhwcMinOp (class in *iree.compiler.dialects.linalg*), 24

PoolingNhwcMinUnsignedOp (class in *iree.compiler.dialects.linalg*), 25

PoolingNhwcSumOp (class in *iree.compiler.dialects.linalg*), 25

PoolingNwcMaxOp (class in *iree.compiler.dialects.linalg*), 25

PoolingNwcMaxUnsignedOp (class in *iree.compiler.dialects.linalg*), 25

PoolingNwcMinOp (class in *iree.compiler.dialects.linalg*), 25

PoolingNwcMinUnsignedOp (class in *iree.compiler.dialects.linalg*), 25

PoolingNwcSumOp (class in *iree.compiler.dialects.linalg*), 25

pos (*iree.compiler.dialects.vector.ScalableExtractOp* property), 69

pos (*iree.compiler.dialects.vector.ScalableInsertOp* property), 69

position (*iree.compiler.dialects.vector.ExtractElementOp* property), 65

position (*iree.compiler.dialects.vector.InsertElementOp* property), 66

position (*iree.compiler.ir.AffineDimExpr* property), 72

position (*iree.compiler.ir.AffineSymbolExpr* property), 76

PowFOp (class in *iree.compiler.dialects.math*), 35

PowOp (class in *iree.compiler.dialects.tosa*), 60

pred (*iree.compiler.dialects.shape.CstrRequireOp* property), 44

pred (*iree.compiler.dialects.tosa.SelectOp* property), 62

PrefetchOp (class in *iree.compiler.dialects.memref*), 40

PrintOp (class in *iree.compiler.dialects.vector*), 68

push\_float() (*iree.runtime.VmVariantList* method), 9

push\_int() (*iree.runtime.VmVariantList* method), 9

push\_list() (*iree.runtime.VmVariantList* method), 9

push\_ref() (*iree.runtime.VmVariantList* method), 9

PyModuleInterface (class in *iree.runtime*), 7

## Q

QuantizedBatchMatmulOp (class in *iree.compiler.dialects.linalg*), 25

QuantizedMatmulOp (class in *iree.compiler.dialects.linalg*), 25

query() (*iree.runtime.HalDriver* static method), 4

query\_available\_devices() (*iree.runtime.HalDriver* method), 4

query\_available\_drivers() (in module *iree.runtime*), 10

query\_buffer\_compatibility() (*iree.runtime.HalAllocator* method), 4

QUEUE\_DISPATCH (*iree.runtime.BufferCompatibility* attribute), 1

QUEUE\_TRANSFER (*iree.runtime.BufferCompatibility* attribute), 1

## R

rank (*iree.compiler.dialects.shape.RankOp* property), 46

rank (*iree.compiler.ir.ShapedType* property), 90

RankedTensorType (class in *iree.compiler.ir*), 89  
 RankOp (class in *iree.compiler.dialects.memref*), 40  
 RankOp (class in *iree.compiler.dialects.shape*), 46  
 RankOp (class in *iree.compiler.dialects.tensor*), 51  
 READ (*iree.runtime.MemoryAccess* attribute), 7  
 ReallocOp (class in *iree.compiler.dialects.memref*), 40  
 ReciprocalOp (class in *iree.compiler.dialects.tosa*), 60  
 ReduceAllOp (class in *iree.compiler.dialects.tosa*), 60  
 ReduceAnyOp (class in *iree.compiler.dialects.tosa*), 60  
 ReduceFn (class in *iree.compiler.dialects.linalg*), 25  
 ReduceFnType (class in *iree.compiler.dialects.linalg*), 25  
 ReduceFnUse (class in *iree.compiler.dialects.linalg*), 25  
 ReduceMaxOp (class in *iree.compiler.dialects.tosa*), 61  
 ReduceMinOp (class in *iree.compiler.dialects.tosa*), 61  
 ReduceOp (class in *iree.compiler.dialects.linalg*), 25  
 ReduceOp (class in *iree.compiler.dialects.shape*), 46  
 ReduceProdOp (class in *iree.compiler.dialects.tosa*), 61  
 ReduceSumOp (class in *iree.compiler.dialects.tosa*), 61  
 reduction\_dim (*iree.compiler.dialects.vector.ScanOp* property), 69  
 ReductionOp (class in *iree.compiler.dialects.vector*), 68  
 ref (*iree.runtime.HalBufferView* property), 4  
 ref (*iree.runtime.VmBuffer* property), 8  
 ref (*iree.runtime.VmVariantList* property), 9  
 reflection (*iree.runtime.VmFunction* property), 8  
 Region (class in *iree.compiler.ir*), 89  
 region (*iree.compiler.dialects.linalg.BroadcastOp* property), 19  
 region (*iree.compiler.dialects.linalg.GenericOp* property), 21  
 region (*iree.compiler.dialects.linalg.TransposeOp* property), 29  
 region (*iree.compiler.dialects.shape.ReduceOp* property), 47  
 region (*iree.compiler.dialects.tensor.PadOp* property), 51  
 region (*iree.compiler.ir.Block* property), 77  
 RegionIterator (class in *iree.compiler.ir*), 89  
 RegionSequence (class in *iree.compiler.ir*), 89  
 register\_attribute\_builder() (in module *iree.compiler.ir*), 92  
 register\_modules() (*iree.runtime.VmContext* method), 8  
 ReinterpretCastOp (class in *iree.compiler.dialects.memref*), 41  
 REMARK (*iree.compiler.ir.DiagnosticSeverity* attribute), 81  
 replace() (*iree.compiler.ir.AffineMap* method), 75  
 replace\_all\_symbol\_uses() (*iree.compiler.ir.SymbolTable* static method), 90  
 res (*iree.compiler.dialects.vector.FlatTransposeOp* property), 66  
 res (*iree.compiler.dialects.vector.InsertOp* property), 67  
 res (*iree.compiler.dialects.vector.InsertStridedSliceOp* property), 67  
 res (*iree.compiler.dialects.vector.MatmulOp* property), 68  
 res (*iree.compiler.dialects.vector.ScalableExtractOp* property), 69  
 res (*iree.compiler.dialects.vector.ScalableInsertOp* property), 69  
 res (*iree.compiler.dialects.vector.VectorScaleOp* property), 71  
 RescaleOp (class in *iree.compiler.dialects.tosa*), 61  
 reshape() (*iree.runtime.DeviceArray* method), 3  
 ReshapeOp (class in *iree.compiler.dialects.memref*), 41  
 ReshapeOp (class in *iree.compiler.dialects.tensor*), 51  
 ReshapeOp (class in *iree.compiler.dialects.tosa*), 61  
 ReshapeOp (class in *iree.compiler.dialects.vector*), 68  
 ResizeOp (class in *iree.compiler.dialects.tosa*), 62  
 resolve\_dimension\_name() (*iree.compiler.dialects.linalg.index* method), 31  
 result (*iree.compiler.dialects.linalg.BroadcastOp* property), 19  
 result (*iree.compiler.dialects.linalg.IndexOp* property), 21  
 result (*iree.compiler.dialects.linalg.MapOp* property), 23  
 result (*iree.compiler.dialects.linalg.TransposeOp* property), 29  
 result (*iree.compiler.dialects.math.AbsFOp* property), 31  
 result (*iree.compiler.dialects.math.AbsIOp* property), 32  
 result (*iree.compiler.dialects.math.Atan2Op* property), 32  
 result (*iree.compiler.dialects.math.AtanOp* property), 32  
 result (*iree.compiler.dialects.math.CbrtOp* property), 32  
 result (*iree.compiler.dialects.math.CeilOp* property), 32  
 result (*iree.compiler.dialects.math.CopySignOp* property), 32  
 result (*iree.compiler.dialects.math.CosOp* property), 33  
 result (*iree.compiler.dialects.math.CountLeadingZerosOp* property), 33  
 result (*iree.compiler.dialects.math.CountTrailingZerosOp* property), 33  
 result (*iree.compiler.dialects.math.CtPopOp* property), 33  
 result (*iree.compiler.dialects.math.ErfOp* property), 33  
 result (*iree.compiler.dialects.math.Exp2Op* property), 33  
 result (*iree.compiler.dialects.math.ExpM1Op* property), 33  
 result (*iree.compiler.dialects.math.ExpOp* property), 34

result (iree.compiler.dialects.math.FloorOp property), 34	42
result (iree.compiler.dialects.math.FmaOp property), 34	result (iree.compiler.dialects.shape.AssumingAllOp property), 42
result (iree.compiler.dialects.math.FPowIOp property), 34	result (iree.compiler.dialects.shape.BroadcastOp prop- erty), 43
result (iree.compiler.dialects.math.IPowIOp property), 34	result (iree.compiler.dialects.shape.ConcatOp prop- erty), 43
result (iree.compiler.dialects.math.Log10Op property), 34	result (iree.compiler.dialects.shape.ConstShapeOp property), 43
result (iree.compiler.dialects.math.Log1pOp property), 35	result (iree.compiler.dialects.shape.ConstSizeOp prop- erty), 43
result (iree.compiler.dialects.math.Log2Op property), 35	result (iree.compiler.dialects.shape.ConstWitnessOp property), 43
result (iree.compiler.dialects.math.LogOp property), 35	result (iree.compiler.dialects.shape.CstrBroadcastableOp property), 43
result (iree.compiler.dialects.math.PowFOp property), 35	result (iree.compiler.dialects.shape.CstrEqOp prop- erty), 44
result (iree.compiler.dialects.math.RoundEvenOp prop- erty), 35	result (iree.compiler.dialects.shape.CstrRequireOp property), 44
result (iree.compiler.dialects.math.RoundOp property), 35	result (iree.compiler.dialects.shape.DivOp property), 44
result (iree.compiler.dialects.math.RsqrtOp property), 36	result (iree.compiler.dialects.shape.FromExtentTensorOp property), 44
result (iree.compiler.dialects.math.SinOp property), 36	result (iree.compiler.dialects.shape.IndexToSizeOp property), 45
result (iree.compiler.dialects.math.SqrtOp property), 36	result (iree.compiler.dialects.shape.IsBroadcastableOp property), 45
result (iree.compiler.dialects.math.TanhOp property), 36	result (iree.compiler.dialects.shape.MaxOp property), 46
result (iree.compiler.dialects.math.TanOp property), 36	result (iree.compiler.dialects.shape.MeetOp property), 46
result (iree.compiler.dialects.math.TruncOp property), 36	result (iree.compiler.dialects.shape.MinOp property), 46
result (iree.compiler.dialects.memref.AtomicRMWOp property), 37	result (iree.compiler.dialects.shape.MulOp property), 46
result (iree.compiler.dialects.memref.AtomicYieldOp property), 38	result (iree.compiler.dialects.shape.NumElementsOp property), 46
result (iree.compiler.dialects.memref.CollapseShapeOp property), 38	result (iree.compiler.dialects.shape.ReduceOp prop- erty), 47
result (iree.compiler.dialects.memref.DimOp property), 38	result (iree.compiler.dialects.shape.ShapeEqOp prop- erty), 47
result (iree.compiler.dialects.memref.ExpandShapeOp property), 39	result (iree.compiler.dialects.shape.ShapeOfOp prop- erty), 47
result (iree.compiler.dialects.memref.GenericAtomicRMWOp property), 39	result (iree.compiler.dialects.shape.SizeToIndexOp property), 47
result (iree.compiler.dialects.memref.GetGlobalOp property), 39	result (iree.compiler.dialects.shape.ToExtentTensorOp property), 47
result (iree.compiler.dialects.memref.ReinterpretCastOp property), 41	result (iree.compiler.dialects.shape.ValueAsShapeOp property), 48
result (iree.compiler.dialects.memref.ReshapeOp prop- erty), 41	result (iree.compiler.dialects.shape.ValueOfOp prop- erty), 48
result (iree.compiler.dialects.memref.SubViewOp prop- erty), 41	result (iree.compiler.dialects.shape.WithOp property), 48
result (iree.compiler.dialects.shape.AddOp property), 42	result (iree.compiler.dialects.tensor.CollapseShapeOp
result (iree.compiler.dialects.shape.AnyOp property),	



property), 48

result (*iree.compiler.dialects.tensor.DimOp* property), 49

result (*iree.compiler.dialects.tensor.ExpandShapeOp* property), 49

result (*iree.compiler.dialects.tensor.ExtractOp* property), 49

result (*iree.compiler.dialects.tensor.ExtractSliceOp* property), 49

result (*iree.compiler.dialects.tensor.FromElementsOp* property), 49

result (*iree.compiler.dialects.tensor.GatherOp* property), 50

result (*iree.compiler.dialects.tensor.GenerateOp* property), 50

result (*iree.compiler.dialects.tensor.InsertOp* property), 50

result (*iree.compiler.dialects.tensor.InsertSliceOp* property), 50

result (*iree.compiler.dialects.tensor.PackOp* property), 51

result (*iree.compiler.dialects.tensor.PadOp* property), 51

result (*iree.compiler.dialects.tensor.ReshapeOp* property), 51

result (*iree.compiler.dialects.tensor.ScatterOp* property), 52

result (*iree.compiler.dialects.tensor.UnPackOp* property), 52

result (*iree.compiler.dialects.vector.BitCastOp* property), 64

result (*iree.compiler.dialects.vector.ExpandLoadOp* property), 65

result (*iree.compiler.dialects.vector.ExtractElementOp* property), 65

result (*iree.compiler.dialects.vector.FMAOp* property), 66

result (*iree.compiler.dialects.vector.GatherOp* property), 66

result (*iree.compiler.dialects.vector.InsertElementOp* property), 66

result (*iree.compiler.dialects.vector.LoadOp* property), 67

result (*iree.compiler.dialects.vector.MaskedLoadOp* property), 67

result (*iree.compiler.dialects.vector.ReshapeOp* property), 69

result (*iree.compiler.dialects.vector.ShapeCastOp* property), 70

result (*iree.compiler.dialects.vector.TransferWriteOp* property), 71

result (*iree.compiler.dialects.vector.TransposeOp* property), 71

result (*iree.compiler.dialects.vector.TypeCastOp* property), 71

result\_number (*iree.compiler.ir.OpResult* property), 87

result\_tensors (*iree.compiler.dialects.linalg.GenericOp* property), 21

results (*iree.compiler.ir.AffineMap* property), 75

results (*iree.compiler.ir.FunctionType* property), 83

results\_ (*iree.compiler.dialects.memref.AllocScopeOp* property), 37

results\_ (*iree.compiler.dialects.memref.AllocScopeReturnOp* property), 37

results\_ (*iree.compiler.dialects.shape.AssumingOp* property), 42

results\_ (*iree.compiler.dialects.vector.MaskOp* property), 67

results\_ (*iree.compiler.dialects.vector.WarpExecuteOnLane0Op* property), 71

ReturnOp (class in *iree.compiler.dialects.shape*), 47

ReverseOp (class in *iree.compiler.dialects.tosa*), 62

RFFT2dOp (class in *iree.compiler.dialects.tosa*), 60

rhs (*iree.compiler.dialects.math.Atan2Op* property), 32

rhs (*iree.compiler.dialects.math.CopySignOp* property), 32

rhs (*iree.compiler.dialects.math.FPowIOp* property), 34

rhs (*iree.compiler.dialects.math.IPowIOp* property), 34

rhs (*iree.compiler.dialects.math.PowFOp* property), 35

rhs (*iree.compiler.dialects.shape.AddOp* property), 42

rhs (*iree.compiler.dialects.shape.ConcatOp* property), 43

rhs (*iree.compiler.dialects.shape.DivOp* property), 44

rhs (*iree.compiler.dialects.shape.MaxOp* property), 46

rhs (*iree.compiler.dialects.shape.MinOp* property), 46

rhs (*iree.compiler.dialects.shape.MulOp* property), 46

rhs (*iree.compiler.dialects.vector.ContractionOp* property), 65

rhs (*iree.compiler.dialects.vector.FMAOp* property), 66

rhs (*iree.compiler.dialects.vector.MatmulOp* property), 68

rhs (*iree.compiler.dialects.vector.OuterProductOp* property), 68

rhs (*iree.compiler.ir.AffineBinaryExpr* property), 72

rhs\_columns (*iree.compiler.dialects.vector.MatmulOp* property), 68

round (*iree.compiler.dialects.tosa.ArithmeticRightShiftOp* property), 53

RoundEvenOp (class in *iree.compiler.dialects.math*), 35

RoundOp (class in *iree.compiler.dialects.math*), 35

rows (*iree.compiler.dialects.vector.FlatTransposeOp* property), 66

RsqrtOp (class in *iree.compiler.dialects.math*), 35

RsqrtOp (class in *iree.compiler.dialects.tosa*), 62

run() (*iree.compiler.passmanager.PassManager* method), 93

## S

- ScalableExtractOp (class in *iree.compiler.dialects.vector*), 69
- ScalableInsertOp (class in *iree.compiler.dialects.vector*), 69
- SCALAR (*iree.compiler.dialects.linalg*.OperandKind attribute), 24
- scalar (*iree.compiler.dialects.tensor.InsertOp* property), 50
- scalar\_name (*iree.compiler.dialects.linalg.ScalarDef* property), 26
- ScalarArg (class in *iree.compiler.dialects.linalg*), 26
- ScalarAssign (class in *iree.compiler.dialects.linalg*), 26
- ScalarConst (class in *iree.compiler.dialects.linalg*), 26
- ScalarDef (class in *iree.compiler.dialects.linalg*), 26
- ScalarExpression (class in *iree.compiler.dialects.linalg*), 26
- ScalarFn (class in *iree.compiler.dialects.linalg*), 26
- ScalarIndex (class in *iree.compiler.dialects.linalg*), 27
- scale32 (*iree.compiler.dialects.tosa.RescaleOp* property), 61
- ScanOp (class in *iree.compiler.dialects.vector*), 69
- ScatterOp (class in *iree.compiler.dialects.tensor*), 51
- ScatterOp (class in *iree.compiler.dialects.tosa*), 62
- ScatterOp (class in *iree.compiler.dialects.vector*), 69
- SelectOp (class in *iree.compiler.dialects.tosa*), 62
- set\_symbol\_name() (*iree.compiler.ir.SymbolTable* static method), 91
- set\_type() (*iree.compiler.ir.BlockArgument* method), 77
- set\_visibility() (*iree.compiler.ir.SymbolTable* static method), 91
- severity (*iree.compiler.ir.Diagnostic* property), 80
- severity (*iree.compiler.ir.DiagnosticInfo* property), 81
- Shape (class in *iree.runtime*), 8
- shape (*iree.compiler.dialects.memref.ReshapeOp* property), 41
- shape (*iree.compiler.dialects.shape.ConstShapeOp* property), 43
- shape (*iree.compiler.dialects.shape.FromExtentsOp* property), 45
- shape (*iree.compiler.dialects.shape.GetExtentOp* property), 45
- shape (*iree.compiler.dialects.shape.NumElementsOp* property), 46
- shape (*iree.compiler.dialects.shape.RankOp* property), 46
- shape (*iree.compiler.dialects.shape.ReduceOp* property), 47
- shape (*iree.compiler.dialects.shape.WithOp* property), 48
- shape (*iree.compiler.dialects.tensor.ReshapeOp* property), 51
- shape (*iree.compiler.ir.ShapedType* property), 90
- shape (*iree.runtime.DeviceArray* property), 3
- shape (*iree.runtime.HalBufferView* property), 4
- ShapeCastOp (class in *iree.compiler.dialects.vector*), 70
- ShapedType (class in *iree.compiler.ir*), 89
- ShapeEqOp (class in *iree.compiler.dialects.shape*), 47
- ShapeOfOp (class in *iree.compiler.dialects.shape*), 47
- shapes (*iree.compiler.dialects.shape.BroadcastOp* property), 43
- shapes (*iree.compiler.dialects.shape.CstrBroadcastableOp* property), 44
- shapes (*iree.compiler.dialects.shape.CstrEqOp* property), 44
- shapes (*iree.compiler.dialects.shape.IsBroadcastableOp* property), 45
- shapes (*iree.compiler.dialects.shape.ShapeEqOp* property), 47
- SHARING\_CONCURRENT (*iree.runtime.BufferUsage* attribute), 2
- SHARING\_EXPORT (*iree.runtime.BufferUsage* attribute), 2
- SHARING\_IMMUTABLE (*iree.runtime.BufferUsage* attribute), 2
- SHARING\_REPLICATE (*iree.runtime.BufferUsage* attribute), 2
- shift (*iree.compiler.dialects.tosa.ApplyScaleOp* property), 53
- shift (*iree.compiler.dialects.tosa.MulOp* property), 59
- ShuffleOp (class in *iree.compiler.dialects.vector*), 70
- SigmoidOp (class in *iree.compiler.dialects.tosa*), 62
- SIGNATURE\_DEF (*iree.compiler.tools.tf.ImportType* attribute), 17
- SinOp (class in *iree.compiler.dialects.math*), 36
- SINT\_16 (*iree.runtime.HalElementType* attribute), 6
- SINT\_32 (*iree.runtime.HalElementType* attribute), 6
- SINT\_4 (*iree.runtime.HalElementType* attribute), 6
- SINT\_64 (*iree.runtime.HalElementType* attribute), 6
- SINT\_8 (*iree.runtime.HalElementType* attribute), 6
- size (*iree.runtime.VmVariantList* property), 9
- sizes (*iree.compiler.dialects.memref.ExtractStridedMetadataOp* property), 39
- sizes (*iree.compiler.dialects.memref.ReinterpretCastOp* property), 41
- sizes (*iree.compiler.dialects.memref.SubViewOp* property), 41
- sizes (*iree.compiler.dialects.memref.ViewOp* property), 42
- sizes (*iree.compiler.dialects.tensor.ExtractSliceOp* property), 49
- sizes (*iree.compiler.dialects.tensor.InsertSliceOp* property), 50
- sizes (*iree.compiler.dialects.tensor.ParallelInsertSliceOp* property), 51
- SizeToIndexOp (class in *iree.compiler.dialects.shape*), 47
- SliceOp (class in *iree.compiler.dialects.tosa*), 63

<code>source</code> ( <code>iree.compiler.dialects.memref.CastOp</code> property), 38	<code>source</code> ( <code>iree.compiler.dialects.vector.MultiDimReductionOp</code> property), 68
<code>source</code> ( <code>iree.compiler.dialects.memref.CopyOp</code> property), 38	<code>source</code> ( <code>iree.compiler.dialects.vector.PrintOp</code> property), 68
<code>source</code> ( <code>iree.compiler.dialects.memref.DimOp</code> property), 38	<code>source</code> ( <code>iree.compiler.dialects.vector.ScalableExtractOp</code> property), 69
<code>source</code> ( <code>iree.compiler.dialects.memref.ExtractAlignedPointerInplaceOp</code> property), 39	<code>source</code> ( <code>iree.compiler.dialects.vector.ScalableInsertOp</code> property), 69
<code>source</code> ( <code>iree.compiler.dialects.memref.ExtractStridedMetadataOp</code> property), 39	<code>source</code> ( <code>iree.compiler.dialects.vector.ScanOp</code> property), 69
<code>source</code> ( <code>iree.compiler.dialects.memref.MemorySpaceCastOp</code> property), 40	<code>source</code> ( <code>iree.compiler.dialects.vector.ShapeCastOp</code> property), 70
<code>source</code> ( <code>iree.compiler.dialects.memref.ReallocOp</code> property), 40	<code>source</code> ( <code>iree.compiler.dialects.vector.TransferReadOp</code> property), 70
<code>source</code> ( <code>iree.compiler.dialects.memref.ReinterpretCastOp</code> property), 41	<code>source</code> ( <code>iree.compiler.dialects.vector.TransferWriteOp</code> property), 71
<code>source</code> ( <code>iree.compiler.dialects.memref.ReshapeOp</code> property), 41	<code>SplatOp</code> (class in <code>iree.compiler.dialects.tensor</code> ), 52
<code>source</code> ( <code>iree.compiler.dialects.memref.SubViewOp</code> property), 41	<code>SplatOp</code> (class in <code>iree.compiler.dialects.vector</code> ), 70
<code>source</code> ( <code>iree.compiler.dialects.memref.ViewOp</code> property), 42	<code>SplitAtOp</code> (class in <code>iree.compiler.dialects.shape</code> ), 47
<code>source</code> ( <code>iree.compiler.dialects.tensor.CastOp</code> property), 48	<code>SqrtOp</code> (class in <code>iree.compiler.dialects.math</code> ), 36
<code>source</code> ( <code>iree.compiler.dialects.tensor.DimOp</code> property), 49	<code>src</code> ( <code>iree.compiler.dialects.memref.CollapseShapeOp</code> property), 38
<code>source</code> ( <code>iree.compiler.dialects.tensor.ExtractSliceOp</code> property), 49	<code>src</code> ( <code>iree.compiler.dialects.memref.ExpandShapeOp</code> property), 39
<code>source</code> ( <code>iree.compiler.dialects.tensor.GatherOp</code> property), 50	<code>src</code> ( <code>iree.compiler.dialects.tensor.CollapseShapeOp</code> property), 48
<code>source</code> ( <code>iree.compiler.dialects.tensor.InsertSliceOp</code> property), 50	<code>src</code> ( <code>iree.compiler.dialects.tensor.ExpandShapeOp</code> property), 49
<code>source</code> ( <code>iree.compiler.dialects.tensor.PackOp</code> property), 51	<code>stashed_flatbuffer_blob</code> ( <code>iree.runtime.VmModule</code> property), 9
<code>source</code> ( <code>iree.compiler.dialects.tensor.PadOp</code> property), 51	<code>statistics</code> ( <code>iree.runtime.HalAllocator</code> property), 4
<code>source</code> ( <code>iree.compiler.dialects.tensor.ParallelInsertSliceOp</code> property), 51	<code>StoreOp</code> (class in <code>iree.compiler.dialects.memref</code> ), 41
<code>source</code> ( <code>iree.compiler.dialects.tensor.ReshapeOp</code> property), 51	<code>StoreOp</code> (class in <code>iree.compiler.dialects.vector</code> ), 70
<code>source</code> ( <code>iree.compiler.dialects.tensor.ScatterOp</code> property), 52	<code>StridedLayoutAttr</code> (class in <code>iree.compiler.ir</code> ), 90
<code>source</code> ( <code>iree.compiler.dialects.tensor.UnPackOp</code> property), 52	<code>strides</code> ( <code>iree.compiler.dialects.memref.ExtractStridedMetadataOp</code> property), 39
<code>source</code> ( <code>iree.compiler.dialects.vector.BitCastOp</code> property), 64	<code>strides</code> ( <code>iree.compiler.dialects.memref.ReinterpretCastOp</code> property), 41
<code>source</code> ( <code>iree.compiler.dialects.vector.BroadcastOp</code> property), 64	<code>strides</code> ( <code>iree.compiler.dialects.memref.SubViewOp</code> property), 41
<code>source</code> ( <code>iree.compiler.dialects.vector.InsertElementOp</code> property), 66	<code>strides</code> ( <code>iree.compiler.dialects.tensor.ExtractSliceOp</code> property), 49
<code>source</code> ( <code>iree.compiler.dialects.vector.InsertOp</code> property), 67	<code>strides</code> ( <code>iree.compiler.dialects.tensor.InsertSliceOp</code> property), 50
<code>source</code> ( <code>iree.compiler.dialects.vector.InsertStridedSliceOp</code> property), 67	<code>strides</code> ( <code>iree.compiler.dialects.tensor.ParallelInsertSliceOp</code> property), 51
	<code>strides</code> ( <code>iree.compiler.ir.StridedLayoutAttr</code> property), 90
	<code>StringAttr</code> (class in <code>iree.compiler.ir</code> ), 90
	<code>sub</code> ( <code>iree.compiler.dialects.linalg.BinaryFn</code> attribute), 19
	<code>SubOp</code> (class in <code>iree.compiler.dialects.tosa</code> ), 63
	<code>SubViewOp</code> (class in <code>iree.compiler.dialects.memref</code> ), 41
	<code>sym_name</code> ( <code>iree.compiler.dialects.memref.GlobalOp</code> property), 40

[sym\\_name \(iree.compiler.dialects.shape.FuncOp property\), 45](#)  
[sym\\_name \(iree.compiler.dialects.shape.FunctionLibraryOp property\), 45](#)  
[sym\\_visibility \(iree.compiler.dialects.memref.GlobalOp property\), 40](#)  
[sym\\_visibility \(iree.compiler.dialects.shape.FuncOp property\), 45](#)  
[sym\\_visibility \(iree.compiler.dialects.shape.FunctionLibraryOp property\), 45](#)  
[symbol\\_count \(iree.compiler.dialects.linalg.AffineBuildState property\), 18](#)  
[SymbolDef \(class in iree.compiler.dialects.linalg\), 27](#)  
[symbolOperands \(iree.compiler.dialects.memref.AllocOp property\), 37](#)  
[symbolOperands \(iree.compiler.dialects.memref.AllocOp property\), 37](#)  
[SymbolTable \(class in iree.compiler.ir\), 90](#)  
[SystemContext \(class in iree.runtime\), 8](#)

## T

[table \(iree.compiler.dialects.tosa.TableOp property\), 63](#)  
[TableOp \(class in iree.compiler.dialects.tosa\), 63](#)  
[tagIndices \(iree.compiler.dialects.memref.DmaWaitOp property\), 39](#)  
[tagMemRef \(iree.compiler.dialects.memref.DmaWaitOp property\), 39](#)  
[tail \(iree.compiler.dialects.shape.SplitAtOp property\), 47](#)  
[TanhOp \(class in iree.compiler.dialects.math\), 36](#)  
[TanhOp \(class in iree.compiler.dialects.tosa\), 63](#)  
[TanOp \(class in iree.compiler.dialects.math\), 36](#)  
[target \(iree.compiler.dialects.memref.CopyOp property\), 38](#)  
[TEMP\\_PATH\\_ENV\\_KEY \(iree.compiler.tools.debugging.TempFileSaver attribute\), 14](#)  
[TempFileSaver \(class in iree.compiler.tools.debugging\), 14](#)  
[tensor \(iree.compiler.dialects.memref.TensorStoreOp property\), 42](#)  
[tensor \(iree.compiler.dialects.tensor.ExtractOp property\), 49](#)  
[tensor \(iree.compiler.dialects.tensor.RankOp property\), 51](#)  
[tensor\\_name \(iree.compiler.dialects.linalg.TensorUse property\), 28](#)  
[TensorDef \(class in iree.compiler.dialects.linalg\), 27](#)  
[TensorExpression \(class in iree.compiler.dialects.linalg\), 27](#)  
[TensorFn \(class in iree.compiler.dialects.linalg\), 28](#)  
[TensorReduceFn \(class in iree.compiler.dialects.linalg\), 28](#)  
[TensorStoreOp \(class in iree.compiler.dialects.memref\), 41](#)  
[TensorUse \(class in iree.compiler.dialects.linalg\), 28](#)  
[then\\_branch \(iree.compiler.dialects.tosa.IfOp property\), 58](#)  
[TileOp \(class in iree.compiler.dialects.tosa\), 63](#)  
[TM\\_TENSOR \(iree.compiler.tools.InputType attribute\), 13](#)  
[to\\_host\(\) \(iree.runtime.DeviceArray method\), 3](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.const method\), 30](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.index method\), 31](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.ScalarDef method\), 26](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.TensorExpression method\), 28](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.TensorFn method\), 28](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.TensorReduceFn method\), 28](#)  
[to\\_scalar\\_expression\(\) \(iree.compiler.dialects.linalg.TensorUse method\), 28](#)  
[to\\_yaml\(\) \(iree.compiler.dialects.linalg.YAMLObject class method\), 30](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.LinalgOpConfig method\), 22](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.LinalgStructuredOpConfig method\), 22](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.OperandDefConfig method\), 24](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.OpMetadataDef method\), 23](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.ScalarAssign method\), 26](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.ScalarExpression method\), 26](#)  
[to\\_yaml\\_custom\\_dict\(\) \(iree.compiler.dialects.linalg.YAMLObject method\), 30](#)  
[ToExtentTensorOp \(class in iree.compiler.dialects.shape\), 47](#)  
[TOSA \(iree.compiler.tools.InputType attribute\), 13](#)  
[Tracer \(class in iree.runtime\), 8](#)  
[tracer \(iree.runtime.Config attribute\), 3](#)  
[TRANSFER \(iree.runtime.BufferUsage attribute\), 2](#)



TRANSFER\_SOURCE (*iree.runtime.BufferUsage* attribute), 2  
 TRANSFER\_TARGET (*iree.runtime.BufferUsage* attribute), 3  
 TransferReadOp (class in *iree.compiler.dialects.vector*), 70  
 TransferWriteOp (class in *iree.compiler.dialects.vector*), 70  
 TransposeConv2D0p (class in *iree.compiler.dialects.tosa*), 63  
 TransposeOp (class in *iree.compiler.dialects.linalg*), 28  
 TransposeOp (class in *iree.compiler.dialects.memref*), 42  
 TransposeOp (class in *iree.compiler.dialects.tosa*), 63  
 TransposeOp (class in *iree.compiler.dialects.vector*), 71  
 trim() (*iree.runtime.HalAllocator* method), 4  
 TruncOp (class in *iree.compiler.dialects.math*), 36  
 TupleType (class in *iree.compiler.ir*), 91  
 Type (class in *iree.compiler.ir*), 91  
 TYPE (*iree.compiler.dialects.linalg.FunctionKind* attribute), 21  
 type (*iree.compiler.ir.AffineMapAttr* property), 75  
 type (*iree.compiler.ir.ArrayAttr* property), 76  
 type (*iree.compiler.ir.Attribute* property), 76  
 type (*iree.compiler.ir.BoolAttr* property), 78  
 type (*iree.compiler.ir.DenseBoolArrayAttr* property), 78  
 type (*iree.compiler.ir.DenseElementsAttr* property), 79  
 type (*iree.compiler.ir.DenseF32ArrayAttr* property), 79  
 type (*iree.compiler.ir.DenseF64ArrayAttr* property), 79  
 type (*iree.compiler.ir.DenseFPElementsAttr* property), 79  
 type (*iree.compiler.ir.DenseI16ArrayAttr* property), 80  
 type (*iree.compiler.ir.DenseI32ArrayAttr* property), 80  
 type (*iree.compiler.ir.DenseI64ArrayAttr* property), 80  
 type (*iree.compiler.ir.DenseI8ArrayAttr* property), 80  
 type (*iree.compiler.ir.DenseIntElementsAttr* property), 80  
 type (*iree.compiler.ir.DictAttr* property), 81  
 type (*iree.compiler.ir.FlatSymbolRefAttr* property), 82  
 type (*iree.compiler.ir.FloatAttr* property), 83  
 type (*iree.compiler.ir.IntegerAttr* property), 84  
 type (*iree.compiler.ir.OpaqueAttr* property), 88  
 type (*iree.compiler.ir.StridedLayoutAttr* property), 90  
 type (*iree.compiler.ir.StringAttr* property), 90  
 type (*iree.compiler.ir.TypeAttr* property), 91  
 type (*iree.compiler.ir.UnitAttr* property), 91  
 type (*iree.compiler.ir.Value* property), 92  
 TYPE\_FN\_ATTR (*iree.compiler.dialects.linalg.OperandKind* attribute), 24  
 type\_var (*iree.compiler.dialects.linalg.OperandDefConfig* property), 24  
 TypeAttr (class in *iree.compiler.ir*), 91  
 TypeCastOp (class in *iree.compiler.dialects.vector*), 71  
 TypeFn (class in *iree.compiler.dialects.linalg*), 29  
 TypeFnAttrDef (class in *iree.compiler.dialects.linalg*), 29  
 TypeFnType (class in *iree.compiler.dialects.linalg*), 29  
 types (*iree.compiler.ir.BlockArgumentList* property), 77  
 types (*iree.compiler.ir.OpResultList* property), 87  
 TypeVar (class in *iree.compiler.dialects.linalg*), 29

## U

UINT\_16 (*iree.runtime.HalElementType* attribute), 6  
 UINT\_32 (*iree.runtime.HalElementType* attribute), 6  
 UINT\_4 (*iree.runtime.HalElementType* attribute), 6  
 UINT\_64 (*iree.runtime.HalElementType* attribute), 6  
 UINT\_8 (*iree.runtime.HalElementType* attribute), 6  
 UNARY (*iree.compiler.dialects.linalg.FunctionKind* attribute), 21  
 UNARY\_FN\_ATTR (*iree.compiler.dialects.linalg.OperandKind* attribute), 24  
 UnaryFn (class in *iree.compiler.dialects.linalg*), 29  
 UnaryFnAttrDef (class in *iree.compiler.dialects.linalg*), 29  
 UnaryFnType (class in *iree.compiler.dialects.linalg*), 30  
 unique (*iree.compiler.dialects.tensor.GatherOp* property), 50  
 unique (*iree.compiler.dialects.tensor.ScatterOp* property), 52  
 UnitAttr (class in *iree.compiler.ir*), 91  
 unknown() (*iree.compiler.ir.Location* static method), 86  
 UnPackOp (class in *iree.compiler.dialects.tensor*), 52  
 UnrankedMemRefType (class in *iree.compiler.ir*), 91  
 UnrankedTensorType (class in *iree.compiler.ir*), 92  
 UnrealizedConversionCastOp (class in *iree.compiler.dialects.builtin*), 17  
 uses (*iree.compiler.ir.Value* property), 92

## V

v1 (*iree.compiler.dialects.vector.ShuffleOp* property), 70  
 v2 (*iree.compiler.dialects.vector.ShuffleOp* property), 70  
 Value (class in *iree.compiler.ir*), 92  
 value (*iree.compiler.dialects.linalg.Enum* attribute), 21  
 value (*iree.compiler.dialects.memref.AtomicRMWOp* property), 37  
 value (*iree.compiler.dialects.memref.StoreOp* property), 41  
 value (*iree.compiler.dialects.shape.ConstSizeOp* property), 43  
 value (*iree.compiler.dialects.shape.DimOp* property), 44  
 value (*iree.compiler.dialects.tensor.YieldOp* property), 52  
 value (*iree.compiler.dialects.tosa.ApplyScaleOp* property), 53  
 value (*iree.compiler.ir.AffineConstantExpr* property), 72  
 value (*iree.compiler.ir.BoolAttr* property), 78  
 value (*iree.compiler.ir.DiagnosticSeverity* property), 81  
 value (*iree.compiler.ir.FlatSymbolRefAttr* property), 82

- `value` (*iree.compiler.ir.FloatAttr* property), 83
- `value` (*iree.compiler.ir.IntegerAttr* property), 84
- `value` (*iree.compiler.ir.StringAttr* property), 90
- `value` (*iree.compiler.ir.TypeAttr* property), 91
- `value` (*iree.runtime.BufferCompatibility* property), 1
- `value` (*iree.runtime.BufferUsage* property), 3
- `value` (*iree.runtime.HalElementType* property), 6
- `value` (*iree.runtime.Linkage* property), 6
- `value` (*iree.runtime.MemoryAccess* property), 7
- `value` (*iree.runtime.MemoryType* property), 7
- `ValueAsShapeOp` (class in *iree.compiler.dialects.shape*), 47
- `ValueOfOp` (class in *iree.compiler.dialects.shape*), 48
- `values` (*iree.compiler.dialects.linalg.YieldOp* property), 30
- `values` (*iree.compiler.dialects.tosa.GatherOp* property), 57
- `values_in` (*iree.compiler.dialects.tosa.ScatterOp* property), 62
- `values_out` (*iree.compiler.dialects.tosa.ScatterOp* property), 62
- `valueToStore` (*iree.compiler.dialects.vector.CompressStoreOp* property), 64
- `valueToStore` (*iree.compiler.dialects.vector.MaskedStoreOp* property), 68
- `valueToStore` (*iree.compiler.dialects.vector.ScatterOp* property), 70
- `valueToStore` (*iree.compiler.dialects.vector.StoreOp* property), 70
- `VecmatOp` (class in *iree.compiler.dialects.linalg*), 30
- `vector` (*iree.compiler.dialects.vector.BroadcastOp* property), 64
- `vector` (*iree.compiler.dialects.vector.ExtractElementOp* property), 65
- `vector` (*iree.compiler.dialects.vector.ExtractOp* property), 65
- `vector` (*iree.compiler.dialects.vector.ExtractStridedSliceOp* property), 65
- `vector` (*iree.compiler.dialects.vector.ReductionOp* property), 68
- `vector` (*iree.compiler.dialects.vector.ReshapeOp* property), 69
- `vector` (*iree.compiler.dialects.vector.ShuffleOp* property), 70
- `vector` (*iree.compiler.dialects.vector.TransferReadOp* property), 70
- `vector` (*iree.compiler.dialects.vector.TransferWriteOp* property), 71
- `vector` (*iree.compiler.dialects.vector.TransposeOp* property), 71
- `VectorScaleOp` (class in *iree.compiler.dialects.vector*), 71
- `VectorType` (class in *iree.compiler.ir*), 92
- `version` (*iree.runtime.VmModule* property), 9
- `ViewOp` (class in *iree.compiler.dialects.memref*), 42
- `visit_affine_exprs()` (*iree.compiler.dialects.linalg.AffineExprDef* method), 18
- `visit_tensor_exprs()` (*iree.compiler.dialects.linalg.TensorExpression* method), 28
- `visit_tensor_exprs()` (*iree.compiler.dialects.linalg.TensorFn* method), 28
- `visit_tensor_exprs()` (*iree.compiler.dialects.linalg.TensorReduceFn* method), 28
- `vm_context` (*iree.runtime.SystemContext* property), 8
- `vm_function` (*iree.runtime.FunctionInvoker* property), 3
- `vm_instance` (*iree.runtime.Config* attribute), 3
- `VmBuffer` (class in *iree.runtime*), 8
- `VmContext` (class in *iree.runtime*), 8
- `VmFunction` (class in *iree.runtime*), 8
- `VmInstance` (class in *iree.runtime*), 8
- `VmModule` (class in *iree.runtime*), 8
- `VmVariantList` (class in *iree.runtime*), 9
- W**
- `walk_symbol_tables()` (*iree.compiler.ir.SymbolTable* static method), 91
- `WARNING` (*iree.compiler.ir.DiagnosticSeverity* attribute), 81
- `warp_size` (*iree.compiler.dialects.vector.WarpExecuteOnLane0Op* property), 71
- `WarpExecuteOnLane0Op` (class in *iree.compiler.dialects.vector*), 71
- `warpRegion` (*iree.compiler.dialects.vector.WarpExecuteOnLane0Op* property), 71
- `weight` (*iree.compiler.dialects.tosa.Conv2DOp* property), 55
- `weight` (*iree.compiler.dialects.tosa.Conv3DOp* property), 55
- `weight` (*iree.compiler.dialects.tosa.DepthwiseConv2DOp* property), 56
- `weight` (*iree.compiler.dialects.tosa.FullyConnectedOp* property), 57
- `WhileOp` (class in *iree.compiler.dialects.tosa*), 64
- `width` (*iree.compiler.ir.IntegerType* property), 85
- `WithOp` (class in *iree.compiler.dialects.shape*), 48
- `witness` (*iree.compiler.dialects.shape.AssumingOp* property), 42
- `WRITE` (*iree.runtime.MemoryAccess* attribute), 7
- X**
- `XLA` (*iree.compiler.tools.InputType* attribute), 13
- Y**
- `yaml_dump()` (in module *iree.compiler.dialects.linalg*),

31  
[yaml\\_dump\\_all\(\)](#) (in [module iree.compiler.dialects.linalg](#)), 31  
[yaml\\_tag](#) ([iree.compiler.dialects.linalg.LinalgOpConfig](#) attribute), 22  
[yaml\\_tag](#) ([iree.compiler.dialects.linalg.LinalgStructuredOpConfig](#) attribute), 22  
[yaml\\_tag](#) ([iree.compiler.dialects.linalg.OperandDefConfig](#) attribute), 24  
[yaml\\_tag](#) ([iree.compiler.dialects.linalg.OpMetadataDef](#) attribute), 23  
[yaml\\_tag](#) ([iree.compiler.dialects.linalg.ScalarAssign](#) attribute), 26  
[yaml\\_tag](#) ([iree.compiler.dialects.linalg.ScalarExpression](#) attribute), 26  
[YAMLObject](#) (class in [iree.compiler.dialects.linalg](#)), 30  
[YieldOp](#) (class in [iree.compiler.dialects.linalg](#)), 30  
[YieldOp](#) (class in [iree.compiler.dialects.shape](#)), 48  
[YieldOp](#) (class in [iree.compiler.dialects.tensor](#)), 52  
[YieldOp](#) (class in [iree.compiler.dialects.tosa](#)), 64  
[YieldOp](#) (class in [iree.compiler.dialects.vector](#)), 71

## Z

[z](#) ([iree.compiler.dialects.tosa.LogicalAndOp](#) property), 58  
[z](#) ([iree.compiler.dialects.tosa.LogicalOrOp](#) property), 58  
[z](#) ([iree.compiler.dialects.tosa.LogicalXorOp](#) property), 59  
[z](#) ([iree.compiler.dialects.tosa.PowOp](#) property), 60