`-iree-codegen-bufferize-copy-only-dispatches`: Bufferize dispatches that copy to/from interfaces to convert to a linalg.copy op

`-iree-codegen-canonicalize-scf-for`: Adhoc canonicalization of selected loop-carried values/dependencies for scf.for ops

`-iree-codegen-cleanup-buffer-alloc-view`: Performs cleanups over HAL interface/buffer allocation/view operations

`-iree-codegen-concretize-pad-result-shape`: Concretizes tensor.pad op's result shape if its source opimplements OffsetSizeAndStrideOpInterface.

`-iree-codegen-convert-to-destination-passing-style`: Transforms the code to make the dispatch use destination-passing style

Options

```
-use-war-for-cooperative-matrix-codegen : WAR for failure in Cooperative matrix
codegen pipelines. See #10648.
```

`-iree-codegen-decompose-linalg-generic`: Decomposes linalg generic ops into individual ops

It is sometimes advantageous to operate on generic ops which contain at most one non-yield body operation. This is most often the case when needing to materialize individual ops (which some backends require). Note that this is often an extreme pessimization unless if part of a lowering flow which was designed for it.

Operates on tensor based linalg ops.

`-iree-codegen-erase-hal-descriptor-type-from-memref`: Erase #hal.descriptor_type from MemRef memory space

`-iree-codegen-flatten-memref-subspan`: Flatten n-D MemRef subspan ops to 1-D ones and fold byte offsets

`-iree-codegen-fold-affinemin-in-distributed-loops`: Fold `affine.min` ops in distributed loops

`-iree-codegen-fold-tensor-extract-op`: Fold `tensor.extract` operations prior to lowering to LLVM

`-iree-codegen-fuse-tensor-pad-with-consumer`: Fuse tensor.pad op into its consumer op's tiled loop nest

`-iree-codegen-gpu-tile-reduction`: Pass to tile linalg reduction dimensions.

`-iree-codegen-gpu-vectorization`: Pass to convert linalg into Vector.

Options

```
-generate-contract : Try to convert reduction to vector.contract.
-max-vector-size   : Max vector size allowed to avoid creating large vectors.
```

`-iree-codegen-iree-comprehensive-bufferize`: Convert from to Linalg ops on tensors to buffers

Options

```
-test-analysis-only : Only runs inplaceability analysis (for testing purposes only)
-print-conflicts    : Annotates IR with RaW conflicts. Requires test-analysis-only.
```

`-iree-codegen-memrefcopy-to-linalg`: Convert memref.copy to linalg op

`-iree-codegen-optimize-vector-transfer`: Run optimization transformations on vector transfer operations

Options

```
-flatten : Flatten the vector type of vector transfers where possible (contiguous row-major data).
```

`-iree-codegen-pad-dynamic-alloc`: Pass to pad dynamic alloc into static one.

`-iree-codegen-polynomial-approximation`: Convert math operations to their polynomial approximation

`-iree-codegen-reduction-to-gpu`: Convert vector reduction to gpu ops.

`-iree-codegen-rematerialize-parallel-ops`: Pass to rematerialize and merge parallel ops to avoid creating temporary allocs.

`-iree-codegen-remove-single-iteration-loop`: Remove distributed loop with single iteration.

`-iree-codegen-split-full-partial-transfer`: Split a vector.transfer operation into an in-bounds (i.e., no out-of-bounds masking) fastpath and a slowpath.

Options

```
-split-transfers : Split vector transfers between slow (masked) and fast "
        "(unmasked) variants. Possible options are:\n"
          "\tnone [default]: keep unsplit vector.transfer and pay the price\n"
          "\tlinalg-copy: use linalg.fill + linalg.generic for the slow path\n"
          "\tvector-transfers: use extra small unmasked vector.transfers for"
          " the slow path\n
```

`-iree-codegen-test-partitionable-loops-interface`: Test the PartitionableLoopsInterface

`-iree-codegen-tile-and-distribute-to-workgroups`: Tile and distribute operations to workgroups

`-iree-codegen-type-propagation`: Propogate the type of tensor to avoid load/stores of illegal bit widths

`-iree-codegen-vectorize-tensor-pad`: Vectorize a very specific form of tensor.pad with control flows

`-iree-codegen-workgroup-specialization`: Specialize workgroup distribution loops

`-iree-convert-to-llvm`: Perform final conversion from Linalg/HAL/Shape/Vector/Standard to LLVMIR dialect

Options

```
-reassociateFpReductions : Specifies if FP add and mult reductions can be reordered
```

`-iree-convert-to-nvvm`: Perform final conversion from builtin/GPU/HAL/standard dialect to LLVM and NVVM dialects

`-iree-convert-to-rocdl`: Perform final conversion from builtin/GPU/HAL/standard dialect to LLVM and ROCDL dialects

`-iree-convert-to-spirv`: Perform the final conversion to SPIR-V dialect

`-iree-eliminate-empty-tensors`: Eliminate tensor.empty ops to avoid buffer allocations

`-iree-gpu-distribute-shared-memory-copy`: Pass to distribute shared memory copies to threads.

`-iree-gpu-multi-buffering`: Pass to do multi buffering.

`-iree-gpu-pipelining`: Pass to do software pipelining.

Options

```
-epilogue-peeling : Try to use un-peeling epilogue when false, peeled epilouge o.w.
```

`-iree-gpu-reduce-bank-conflicts`: Pass to try to reduce the number of bank conflicts.

`-iree-llvmcpu-assign-constant-ordinals`: Assigns executable constant ordinals across all LLVMCPU variants.

`-iree-llvmcpu-assign-import-ordinals`: Assigns executable import ordinals across all LLVMCPU variants.

`-iree-llvmcpu-check-ir-before-llvm-conversion`: Checks CPU backend specific IR constraints (like no allocas)

`-iree-llvmcpu-emit-vectorization-remarks`: Emit vectorization remarks on Linalg ops

`-iree-llvmcpu-link-executables`: Links LLVMCPU HAL executables within the top-level program module.

`-iree-llvmcpu-lower-executable-target`: Lower executable target using an IREE::HAL::DispatchLoweringPassPipeline

`-iree-llvmcpu-materialize-encoding`: Materialize the encoding for tensor as specified by the backend

`-iree-llvmcpu-mmt4d-vector-lowering`: Apply vector lowering logic to vector ops

`-iree-llvmcpu-synchronize-symbol-visibility`: Synchronizes LLVM linkage with MLIR symbol visibility

`-iree-llvmcpu-unfuse-fma-pass`: Convert llvm.fma into unfused mulf and addf ops

`-iree-llvmcpu-vector-contract-custom-kernels`: Enable custom kernels (inline assembly or intrinsics) for some vector.contract ops

`-iree-llvmcpu-verify-linalg-transform-legality`: Verify that only supported IR constructs are passed to the compiler.

`-iree-llvmgpu-alloc`: Pass to create allocation for some values.

`-iree-llvmgpu-distribute`: Pass to distribute foreachthread ops.

`-iree-llvmgpu-lower-executable-target`: Perform lowering of executable target using one of the IREE::HAL::DispatchLoweringPassPipeline

`-iree-llvmgpu-tensor-pad`: Pass to pad out tensors up to static dimensions.

`-iree-llvmgpu-tensorcore-vectorization`: Pass to convert linalg into Vector and transform it to a form that can be lowered to GPU MMA ops

`-iree-llvmgpu-tile-and-distribute`: Pass to tile and distribute linalg ops within a workgroup.

`-iree-llvmgpu-tile-tensor`: Pass to tile linalg on tensor ops within a workgroup.

`-iree-llvmgpu-vector-lowering`: Pass to lower Vector ops before conversion to LLVM.

`-iree-llvmgpu-vector-to-gpu`: Pass to convert vector to gpu.

`-iree-spirv-annotate-winograd-loops`: Annotate innermost Winograd loops with spirv distribute attribute

`-iree-spirv-breakdown-large-vector`: Break down vectors not natively supported by SPIR-V

`-iree-spirv-create-fast-slow-path`: Create separate fast and slow paths to handle padding

`-iree-spirv-distribute`: Distribute tiled loop nests to invocations

`-iree-spirv-emulate-i64`: Emulate 64-bit integer opts with 32-bit integer ops

`-iree-spirv-lower-executable-target-pass`: Lower the executable target to SPIR-V using one of the IREE::HAL::DispatchLoweringPassPipeline

`-iree-spirv-map-memref-storage-class`: Map MemRef memory spaces to SPIR-V storage classes

`-iree-spirv-tile`: Tile Linalg ops with tensor semantics to invocations

`-iree-spirv-tile-and-distribute`: Tile and distribute Linalg ops with buffer semantics to invocations

`-iree-spirv-tile-and-promote`: Promote tiled Linalg ops with buffer semantics to use workgroup memory and then tile to invocations

Options

```
-promote-c   : Promote C matrix to use shared memory
-skip-thread : Skip tiling and distributing to GPU threads
```

`-iree-spirv-tile-to-cooperative-ops`: Tile Linalg ops with buffer semantics to subgroups and vectorize to vector ops suitable for lowering to SPIR-V cooperative ops

`-iree-spirv-vector-to-gpu-subgroup-mma-ops`: Pass to convert vector ops to GPU subgroup MMA ops.

`-iree-spirv-vectorize`: Vectorize Linalg ops with buffer semantics

`-iree-spirv-vectorize-load-store`: Vectorize load/store of memrefs for better memory access

`-iree-spirv-vectorize-to-cooperative-ops`: Tile Linalg ops with buffer semantics to subgroups and vectorize to vector ops suitable for lowering to SPIR-V cooperative ops

`-iree-test-llvmgpu-legalize-ops`: Test pass for several legalization patterns.

`-iree-transform-dialect-interpreter`: Pass to apply transform dialect operations.

Options

```
-transform-file-name      : Optional filename containing a transform dialect
specification to apply. If left empty, the IR is assumed to contain one top-level
```

```
transform dialect operation somewhere in the module.
-debug-payload-root-tag   : Select the operation with 'transform.iree_tag' attribute
having the given value as payload IR root. This allows user control on what operation
to transform in debug mode, without requiring intimate knowledge of the IREE nested
pass pipeline.\nIf empty (normal operation mode), select the pass anchor operation in
the IREE pipeline, as the payload IR root.
-debug-transform-root-tag : Select the operation with 'transform.iree_tag' attribute
having the given value as container IR for top-level transform ops. This allows user
control on what transformation to apply in debug mode, without requiring intimate
knowledge of the IREE nested pass pipeline.\nIf empty (normal operation mode), select
the container of the top-level transform op.
```

`-iree-vmvx-assign-constant-ordinals`: Assigns executable constant ordinals across all VMVX variants.

`-iree-vmvx-link-executables`: Links VMVX HAL executables within the top-level program module.

`-iree-vmvx-lower-linalg-microkernels`: Lowers linalg ops to the VMVX microkernel library

Options

```
-warn-on-unconverted : Warns on any unconverted linalg ops which remain live
```

`-iree-vmvx-materialize-encoding`: Materialize the encoding for tensor as specified by the backend

`-iree-wgsl-replace-push-constants`: Replaces push constant loads with binding loads for when using WGSL without push constant support

`-iree-workgroup-swizzle`: swizzle the workgroup ids for better cache reuse

Options

```
-logTile : pass the tile value for unit testing
```