CS 4390/5390 Fall 2025
Advanced Compilers
Shirley Moore, Instructor

MLIR Examples

Let's work with two of the example codes from Introduction to MLIR by Stephen Diehl.

Example 1.

```
// example.mlir
func.func @loop_add() -> (index) {
    %init = index.constant 0
    %lb = index.constant 0
    %ub = index.constant 10
    %step = index.constant 1

    %sum = scf.for %iv = %lb to %ub step %step iter_args(%acc = %init) -> (index) {
        %sum_next = arith.addi %acc, %iv : index
        scf.yield %sum_next : index
    }

    return %sum : index
}

func.func @main() -> i32 {
    %out = call @loop_add() : () -> index
    %out_i32 = arith.index_cast %out : index to i32
    func.return %out_i32 : i32
}
```

Follow the instructions at https://www.stephendiehl.com/posts/mlir_introduction/ .

Example 2.

```
// array_add.mlir
module {
  func.func @array_add(%arg0: memref<1024xf32>, %arg1: memref<1024xf32>, %arg2: memref<1024xf32>)
      attributes {llvm.emit_c_interface} {
    %c0 = arith.constant 0 : index
    %c1024 = arith.constant 1024 : index
    %c1 = arith.constant 1 : index

    scf.for %arg3 = %c0 to %c1024 step %c1 {
      %0 = memref.load %arg0[%arg3] : memref<1024xf32>
      %1 = memref.load %arg1[%arg3] : memref<1024xf32>
      %2 = arith.addf %0, %1 : f32
      memref.store %2, %arg2[%arg3] : memref<1024xf32>
    }

    return
```

```
 }
}
```

Follow the instructions at https://www.stephendiehl.com/posts/mlir_memory/ .

```
shirleymoore@Shirleys-MacBook-Pro mlir % mlir-opt array_add.mlir \
  -convert-scf-to-cf \
  -convert-arith-to-llvm \
  -convert-cf-to-llvm \
  -finalize-memref-to-llvm \
  -convert-func-to-llvm \
  -reconcile-unrealized-casts \
  -o array_add_opt.mlir
```

```
mlir-translate array_add_opt.mlir \
  -mlir-to-llvmir \
  -o array_add_opt.ll

llc -filetype=obj --relocation-model=pic array_add_opt.ll -o array_add_opt.o

clang -shared -fPIC array_add.o -o array_add.dylib
```

```
// np_memref.py -- bridge between NumPy's memory layout and MLIR's
// MemRef structure
import numpy as np
from ctypes import c_void_p, c_longlong, Structure


class MemRefDescriptor(Structure):
  """Structure matching MLIR's MemRef descriptor"""

  _fields_ = [
    ("allocated", c_void_p), # Allocated pointer
    ("aligned", c_void_p), # Aligned pointer
    ("offset", c_longlong), # Offset in elements
    ("shape", c_longlong * 1), # Array shape (1D in this case)
    ("stride", c_longlong * 1), # Strides in elements
  ]


def numpy_to_memref(arr):
  """Convert a NumPy array to a MemRef descriptor"""
  if not arr.flags["C_CONTIGUOUS"]:
    arr = np.ascontiguousarray(arr)

  desc = MemRefDescriptor()
  desc.allocated = arr.ctypes.data_as(c_void_p)
```

```
    desc.aligned = desc.allocated
    desc.offset = 0
    desc.shape[0] = arr.shape[0]
    desc.stride[0] = 1

    return desc
```

Then use libarray_add.dylib from Python in two ways:

1) load and use the compiled library from Python
2) JIT-compile the MLIR code at runtime using llvmlite (Python bindings for LLVM).