Instructions

In this exercise, we'll scale a vector (array) of single-precision numbers by a scalar. You'll learn to write simple CUDA Fortran codes employing explicit data transfer as well as unified memory.

The first CUDA Fortran program

Take a look at `scale_vector.cuf` and `scale_vector_um.cuf` and look for TODOs.

- The global attribute is necessary to specify your kernels.

```
attributes(global) subroutine mySub
```

- Since Fortran 2003, you can pass argument by value using `value` type declaration attribute.

```
integer,value :: myInt
```

- Use `device` attribute to declare device arrays.

```
integer,allocatable,device  :: myInt(:)
```

- Use `managed` attribute to declare unified memory arrays.

```
integer,managed,allocatable  :: myInt(:)
```

- If necessary, use native Fortran copy notation to transfer data.

```
myArr=myArr_d
```

- Be sure to load the custom modules of this task.

```
source setup.sh
```

- For compilation, use

```
make
```

- To run your code, call `srun` with the correct parameters. A shortcut is given via

```
make run
or
make run-um
```