Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

MIC-Native and Offload-lab: Running simple C Programs in Native and Offload Mode

In this lab you run simple programs in native and offload mode.
It demonstrates how to use persistent data on the coprocessor and how to overlap computations.

Attention:

- **Compile on supermic.smuc.lrz.de**
- **Run on Compute nodes i01r13c?? for Offload (see llq for the name of the allocated compute node)**
- **Run on MICs i01r13c??-mic0/1 for Native Mode**

Lab 5: Asynchronous Offload to Multiple Coprocessors.

- Put the MxM computation into a separate function.
- Call the function from the offload region and test the code.
- Allocate and initialize new input and result matrices on the host (of the same size).
- Use

  #pragma offload target(mic:0) signal(...)
  {…}
  #pragma offload target(mic:1) signal(…)
  {…}
  #pragma offload_wait target(mic:0) wait(…)
  #pragma offload_wait target(mic:1) wait(…)

- to do asynchronous MxM computations on both Xeon Phi Cards.

Lab 6: Explicit Worksharing using OMP sections.

- Now use the following construct to run the code on 1 MIC and on the host:

```
#pragma omp parallel
{
   #pragma omp sections
   {
         #pragma omp section
         {
         ……..// Code running on first MIC
         }

         #pragma omp section
         {
         ……..// Code running on the host
         }
   }
}
```

- Change the second region so that the code there is running on the second MIC.

Lab 7: Offloading to Intel Xeon Phi using persistent data

In this lab the main offload region of the code used in Lab4 should be replaced by 3 stages:

1. Copy data from the Host to the MIC without computations.
2. Do computations without data transfers.
3. Copy data back to the host.

- Add a *#pragma offload target(mic) in(…)* with an empty body before the main compute region.

- Add a nocopy(…) clause to the pragma of the main compute region.

- Add a *#pragma offload target(mic) out(…)* with an empty body after the main compute region.

- Compile and run the code. Interpret the runtime error.

- Define the following macros

```
#define RETAIN free_if(0)
#define REUSE alloc_if(0)
#define ALLOC alloc_if(1)
#define FREE free_if(1)
```

- Add the right combination of RETAIN/REUSE/ALLOC/FREE to the Offload Pragmas.