ACM40640/PH504 Practical 3

ICHEC

2022/23 Spring

1 Hello world!

1. **Build and run** the code in Fig. 1.
2. Write a simple OpenMP code to fork a team of threads, obtain and print the thread IDs, have the master thread print the total number of threads.
3. Write a simple OpenMP code with three parallel regions each with a different number of threads, have the master thread print the total number of threads. Define the number of threads as an environment variable. Then, use **omp_set_num_threads()** function and/or **num_threads()** clause to change them for each parallel region.

```
gcc source.c -o source.X -fopenmp
```

```
gfortran source.f90 -o source.X -fopenmp
```

```c
#include<stdio.h>
#include<omp.h>
int main(void){
  int tid, nthreads;
#pragma omp parallel private(tid),shared(nthreads)
  {
    tid=omp_get_thread_num();
    nthreads=omp_get_num_threads();
    printf("Hello from thread %d out of %d\n", tid,nthreads);
  }
 }
```

```fortran
 program hello
  use omp_lib
  implicit none
  integer :: tid, nthreads
!$omp parallel private(tid), shared(nthreads)
    tid=omp_get_thread_num()
    nthreads=omp_get_num_threads()
    write(*,'(a,1x,i0,1x,a,i0)'), 'Hello from thread',tid,&
      'out of',nthreads
!$omp end parallel
 end program hello
```

Figure 1. OpenMP Hello World! samples. Top C and bottom Fortran.

2 Vector addition

Write a simple program adding two vectors of double precision numbers, $c = a + b$ with each of length n. You can modify the code inc_serial.c or inc_serial.f90 from previous week.

1. Read in n and generate the vectors. Compute the addition multiple times to get a descent run time.
2. Experiment with OpenMP parallel construct and data clauses.
3. Try using OpenMP loop construct to parallelise. We will cover this topic next week.

```c
#pragma omp for
  for(...){
    ...
  }
```

```fortran
!$omp do
  do ...
    ...
  end do
!$omp end do
```

Figure 2. OpenMP Loop Construct. Top C and bottom Fortran.