Assignment 0 – Task 1

- Number of cycles to execute one loop iteration:

$$c = \frac{T}{n} \times f \left[\frac{\text{cy}}{\text{It.}} = \frac{\text{s}}{\text{It.}} \times \frac{\text{cy}}{\text{s}} \right],$$

where f is the clock speed, n is the number of slices, and T is the runtime of the whole loop.

- Important: Fix clock speed on job submission:

```
$ qsub -l nodes=1:ppn=40:f2.2,walltime=01:00:00 -I
```

- Compilation:

```
$ module load intel64
$ icc -O3 -xHost div.c timing.c
```

Assignment 0 – Task 1

- Code

```
#include <stdio.h>
#include "timing.h"

int main (int argc, char**argv) {
  double f = 2.2e9;   // clock frequency in cy/s
  int n = 1000000000; // # of slices
  double delta_x = 1./n,x,sum=0.,ct,wcs,wce,Pi;
  wcs = getTimeStamp();
  for (int i=0; i < n; i++) {
    x = (i+0.5)*delta_x;
    sum += (4.0 / (1.0 + x * x));
  }
  wce = getTimeStamp(); // T = wce-wcs
  Pi = sum * delta_x;
  printf("Pi=%.15lf in %.3lf s -> %.2lf cy/it\n",Pi,wce-wcs,(wce-wcs)/n*f);
  return 0;
}
```

- Run:

```
$ ./a.out
Pi=3.141592653589828 in 3.189 s -> 7.02 cy/it
```

Assignment 0 – Task 2

- Loop body

```
x = (i+0.5)*delta_x;
sum += (4.0 / (1.0 + x * x));
```

→ 6 flops (3 ADD, 2 MULT, 1 DIV). Or not???

- Possible performance metrics
  - Flop/s → not portable
    - compiler might transform code
    - int→float conversion might count or not
    - DIV might not even be an instruction but comprise several flops
  - $1/T$ → Usually OK but varies with loop length in a trivial way
  - n/T → probably best metric overall in this case (it/s, it/cy)

Assignment 0 – Task 3

- Single precision code:

```
double f = 2.2e9;   // clock frequency in cy/s
int n = 1000000000; // # of slices
float delta_x = 1.f/n,x,sum=0.f,ct,wcs,wce,Pi;
wcs = getTimeStamp();
for (int i=0; i < n; i++) {
  x = (i+0.5f)*delta_x;
  sum += (4.0f / (1.0f + x * x));
}
wce = getTimeStamp(); // T = wce-wcs
Pi = sum * delta_x;
printf("Pi=%.7f in %.3lf s -> %.2lf cy/it\n",Pi,wce-wcs, \
        (wce-wcs)/n*f);
```

- Take care to use "**f**" qualifier for all float constants
  - Language has strict rules about type conversion
  - Compiler may be forced to generate code with runtime conversions if the types are not consistent

Assignment 0 – Task 3

- Result

```
$ ./a.out
Pi=1.0737418 in 0.512 s -> 1.13 cy/it
```

- 6 times faster than DP code
  → SP DIV appears to be much faster than DP

- Result is not at all π
  - Summing 10^9 numbers, all between 2 and 4
  - Float type has only ~7 mantissa digits
    - →Massive loss of accuracy as soon as the sum gets > 10^7
    - →Sum is much too small