NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Programming Techniques for Supercomputers Tutorial

Erlangen National High Performance Computing Center

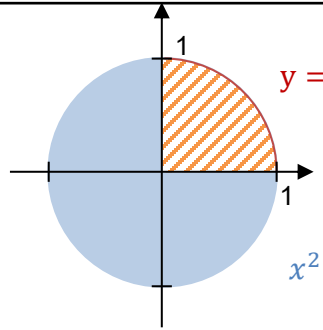
Department of Computer Science

FAU Erlangen-Nürnberg

Sommersemester 2026

Assignment 0 – Task 1

$$\pi = \int_0^1 4\sqrt{1-x^2}\,dx$$



$y = \sqrt{1-x^2}$

$A = \int_0^1 \sqrt{1-x^2}\,dx = \frac{1}{4}\pi$

$x^2 + y^2 = 1 \qquad A_o = r^2\pi = \pi$

```
double sum, x, delta_x, S, E;
int N = 10;          // initial number of slices
do {
  S = getTimeStamp();
  delta_x = 1./N;
  sum = 0.0;
  for (int i=0; i<N; i++) {
    x = (i + 0.5) * delta_x;
    sum = sum + 4.0 * sqrt(1.0-x*x));
  }
  double pi = sum * delta_x;
  E = getTimeStamp();           // T = E-S
  if(E-S > 1e-7) break;

  N *= 2;
} while (1);
```

Assignment 0 – Task 1

```
int main (int argc, char** argv) {

  const double f = 2.4e9                    // clock frequency
  double sum, x, delta_x, S, E;
  for (int N=10; N<1e9; N*=2) {             // start with N=10 and increase by factor of 2
    do {
      S = getTimeStamp();
      delta_x = 1./N;
      sum = 0.0;
      for (int i=0; i<N; i++) {
        x = (i + 0.5) * delta_x;
        sum = sum + 4.0 * sqrt(1.0-x*x));
      }
      double result = sum * delta_x;
      E = getTimeStamp();                   // T = E-S
      printf("Result=%.15lf in %.3le s -> %.2lf cy/it (N = %d)\n", result, E-S, (E-S)/N*f, N);
      if(E-S > 1e-7) break;

      N *= 2;
    } while (1);
  }
  return 0;
}
```

Assignment 0 – Task 2

- 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.

- Compilation:

```
$ module load intel
$ icx -O3 -xHost pi.c
```

- Important: Fix clock speed when running binary:

```
$ srun --cpu-freq=2400000-2400000:performance ./pi
```

Assignment 0 – Task 2

```
$ srun --cpu-freq=2400000-2400000:performance ./pi
Result=3.152411433261644 in 1.250e-07 s -> 30.00 cy/it (N = 10)
Result=3.145430588679039 in 1.090e-07 s -> 13.08 cy/it (N = 20)
Result=3.142951861717275 in 9.900e-08 s -> 5.94 cy/it (N = 40)
Result=3.142073612431767 in 1.430e-07 s -> 4.29 cy/it (N = 80)
Result=3.141762770098016 in 2.420e-07 s -> 3.63 cy/it (N = 160)
Result=3.141652811570951 in 4.430e-07 s -> 3.32 cy/it (N = 320)
Result=3.141613924895905 in 8.430e-07 s -> 3.16 cy/it (N = 640)
Result=3.141600174529618 in 1.645e-06 s -> 3.08 cy/it (N = 1280)
Result=3.141595312713829 in 3.250e-06 s -> 3.05 cy/it (N = 2560)
Result=3.141593593744534 in 6.456e-06 s -> 3.03 cy/it (N = 5120)
Result=3.141592985986884 in 1.287e-05 s -> 3.02 cy/it (N = 10240)
Result=3.141592771110302 in 2.570e-05 s -> 3.01 cy/it (N = 20480)
Result=3.141592695139643 in 5.136e-05 s -> 3.01 cy/it (N = 40960)
Result=3.141592668279891 in 1.027e-04 s -> 3.01 cy/it (N = 81920)
Result=3.141592658783534 in 2.053e-04 s -> 3.01 cy/it (N = 163840)
Result=3.141592655426058 in 4.189e-04 s -> 3.07 cy/it (N = 327680)
Result=3.141592654239025 in 8.239e-04 s -> 3.02 cy/it (N = 655360)
Result=3.141592653819334 in 1.644e-03 s -> 3.01 cy/it (N = 1310720)
Result=3.141592653670921 in 3.311e-03 s -> 3.03 cy/it (N = 2621440)
Result=3.141592653618487 in 6.614e-03 s -> 3.03 cy/it (N = 5242880)
Result=3.141592653599980 in 1.318e-02 s -> 3.02 cy/it (N = 10485760)
Result=3.141592653593140 in 2.632e-02 s -> 3.01 cy/it (N = 20971520)
Result=3.141592653591064 in 5.264e-02 s -> 3.01 cy/it (N = 41943040)
Result=3.141592653590259 in 1.053e-01 s -> 3.01 cy/it (N = 83886080)
Result=3.141592653590255 in 2.105e-01 s -> 3.01 cy/it (N = 167772160)
```

Assignment 0 – Task 3

- Loop body:

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

→ 7 flops (3 ADD, 3 MULT, 1 SQRT). Or not???

- Possible performance metrics
  - Flop/s → not portable
    - compiler might transform code
    - int→float conversion might count or not
    - SQRT might not even be an instruction but comprise several flops (depends if hardware unit available)
  - $1/T$ → Usually OK but varies with loop length in a trivial way
  - n/T → probably best metric overall in this case (it/s, it/cy)
  - Code performance: $P \approx 0.33\ \mathrm{it/cy}$

Assignment 0 – Task 4

- Not fixing the clock frequency but using the performance governor:

```
$ srun --cpu-freq=performance ./pi
...
Result=3.141592653591064 in 3.608e-02 s -> 2.06 cy/it (N = 41943040)
Result=3.141592653590259 in 7.215e-02 s -> 2.06 cy/it (N = 83886080)
Result=3.141592653590255 in 1.443e-01 s -> 2.06 cy/it (N = 167772160)
Result=3.141592653589601 in 2.886e-01 s -> 2.06 cy/it (N = 335544320)
Result=3.141592653589690 in 5.772e-01 s -> 2.06 cy/it (N = 671088640)
```

5.772e-1s instead of 2.105e-1s at N=671088640

→ the actual clock frequency is

$$\frac{5.772}{2.105} \times 2.4\ \text{GHz} \approx 3.50\ \text{GHz}$$