Laboratorio 1 – Paralela

Ejercicio 1

No pude ejecutar los valores de factor solo así dado que me daba un segmentation fault desde un principio debido a las condiciones frontera que se generaban, por lo que en mi caso debí de poner de una vez el cambio de factor = -1.0 y así.

A.

Pi Series Seq

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 1000
Pi value: 3.140593
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 2000
Pi value: 3.141093
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 5000
Pi value: 3.141393
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 10000
Pi value: 3.141493
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 20000
Pi value: 3.141543
```

Pi Series Naive

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# .
/piSeriesNaive 2 1000
Pi value: -2.666667
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 50 2000
Pi value: -3.224584
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 500 3000
Pi value: -3.136142
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 1000 10000
Pi value: 4.855350
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 2000 10000
Pi value: 4.851223
```

En el caso de la secuencial se parece bastante cuando se aumenta la cantidad de la serie, mientras que para el caso del paralelo no se parece en nada el valor por más que se le cambie el número de threads y de la serie.

B.

La dependencia que se da con la variable factor es que una dependencia de datos debido a que el valor de la variable en una iteración depende del valor de la variable en una iteración anterior debido a que tiene que irse en patrón de 1, -1, 1, -1... por lo que los valores serán incorrectos por esta razón al no existir un orden.

C.

La razón de factor negativo es por la ecuación cuando sea un k que sea potencia de 2 será positivo mientras que cuando sea negativo entonces se cambiará. Sin embargo, esto no es del todo certero debido a que no se tiene un control de manera que puede que no se siga el orden secuencial y cuando se lleguen a números pares estos sean negativos y deberían ser positivos y viceversa generando un resultado erróneo.

D.

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000 2
Pi value: 3.140593
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1500 2
Pi value: 3.140926
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000 30
Pi value: 3.140593
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000 500
Pi value: 3.140593
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 10000 500
Pi value: 3.141493
```

Se tiene un valor más aproximado que el que teníamos antes ya que tenemos un flujo de control de la variable de factor que varía entre 1 y -1, generando que el valor sea aproximado mucho más. Así mismo, es paralelo por lo que debería ser más veloz.

E.

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# gcc -o piSeriesNaive -fopenmp piSeriesNaive.c -lm
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000 1
Pi value: 3.140593
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 2000 1
Pi value: 3.141093
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 5000 1
Pi value: 3.141393
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 10000 1
Pi value: 3.141493
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 20000 1
Pi value: 3.141543
```

Para este caso, básicamente se tiene el secuencial que habíamos implementado en un principio ya que se sigue el factor alternando entre 1 y menos 1 como debe de ser y pues solo

se tiene un thread por lo que es la misma secuencia. Sin embargo, respecto al inicial es diferente ya que ahora se tiene un control de la variable factor, haciendo que realmente funcione la implementación.

F.

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# gcc -o piSeriesNaive -fopenmp piSeriesNaiveF.c -lm
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000000 2
Pi value: 3.141592
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000000 500
Pi value: 3.141592
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000000 1000
Pi value: 3.141592
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 2000000 10000
Pi value: 3.141592
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 2000000 100000
Segmentation fault
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 3000000 10000
Pi value: 3.141592
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

G.

n	Tiempo secuencial	Tiempo paralelo threads = cores	Tiempo paralelo threads = 2*cores	Tiempo paralelo (n*10 && threads = cores)
1000000	0.003215	0.000860	0.001396	0.005049
2000000	0.005316	0.001220	0.001788	0.009752
3000000	0.008026	0.001777	0.001641	0.013796
5000000	0.013305	0.002535	0.003207	0.020415
10000000	0.031178	0.041607	0.033428	0.401994

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 3000000
Pi value: 3.141592
Time taken: 0.008026 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 3000000 8
Pi value: 3.141592
Time taken: 0.001777 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 3000000 16
Pi value: 3.141592
Time taken: 0.001641 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 30000000 8
Pi value: 3.141593
Time taken: 0.013796 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 1000000
Pi value: 3.141592
Time taken: 0.003215 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000000 8
Pi value: 3.141592
Time taken: 0.000860 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 1000000 16
Pi value: 3.141592
Time taken: 0.001396 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 10000000 8
Pi value: 3.141593
Time taken: 0.005049 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesSeq 2000000
Pi value: 3.141592
Time taken: 0.005316 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 2000000 8
Pi value: 3.141592
Time taken: 0.001220 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 2000000 16
Pi value: 3.141592
Time taken: 0.001788 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 20000000 8
Pi value: 3.141593
Time taken: 0.009752 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 5000000
Pi value: 3.141592
Time taken: 0.015659 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 5000000 8
Pi value: 3.141592
Time taken: 0.002687 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 5000000 16
Pi value: 3.141592
Time taken: 0.018131 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 50000000 16
Pi value: 3.141593
Time taken: 0.015191 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesSeq 10000000
Pi value: 3.141593
Time taken: 0.031178 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 100000000 8
Pi value: 3.141593
Time taken: 0.041607 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 100000000 16
Pi value: 3.141593
Time taken: 0.033428 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio
1# ./piSeriesNaive 1000000000 8
Pi value: 3.141593
Time taken: 0.401994 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesSeq 5000000
Pi value: 3.141592
Time taken: 0.013305 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 5000000 8
Pi value: 3.141592
Time taken: 0.002535 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 5000000 16
Pi value: 3.141592
Time taken: 0.003207 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 50000000 8
Pi value: 3.141593
Time taken: 0.020415 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

N	Speedup	Eficiencia
100000, exceptuando para el último	• 37.38372093 • 23.03008596 • 6.367597544	• 4.672965116 • 1.439380372 • 0.795949693
2000000, exceptuando para el último	• 43.57377049 • 29.73154362 • 5.4511895	• 5.446721311 • 1.858221477 • 0.681398687
3000000, exceptuando para el último	• 45.16601013 • 48.90920171 • 5.817628298	• 5.645751266 • 3.056825107 • 0.072720354
5000000, exceptuando para el último	• 5.24852071 • 4.148737138 • 0.65166283	• 0.656065089 • 0.259296071 • 0.081457853
10000000, exceptuando para el último	• 0.749345062 • 0.932691157 • 0.077558371	• 0.093668133 • 0.058293197 • 0.009694796

La escalabilidad débil para 1000000 y 2000000 en n y 8 y 16 con los threads es de 0.48098434 al comparar los tiempos de 0.000860 con 0.001788. Así mismo, 0.07583462965 en 5000000 y 10000000 al comparar tiempos de 0.002535 y 0.033428 respectivamente. Esto sugiere que no hay una escalabilidad débil ya que no aumenta proporcionalmente la escalabilidad entre más threads y más carga se ponga será proporcionalmente escalable, es más hay una pérdida de eficiencia, esto puede deberse a dificultad en una gestión de threads.
Por otra parte, la escalabilidad fuerte sí existe en para n 10^6 2*10^6 y 3*10^6 siempre que no se aumenten esos valores en *10, de manera que se puede observar mediante el speed up que el programa escala bien, sin embargo, sí sería necesario aumentar un poco el número de threads a medida que se aumentan los valores de n dado que a veces la escalabilidad se queda muy corta a medida que aumenta el número de threads, sin embargo, no existe una escalabilidad débil ya que no es proporcional como se demostró.

H.

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# gcc -o piSeriesNaive -fopenmp piSeriesNaiveH.c -lm
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 16 dynamic
Pi value: 3.141593
Time taken: 0.015936 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 16 guided
Pi value: 3.141593
Time taken: 0.006754 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 16 auto
Pi value: 3.141593
Time taken: 0.006695 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 16 static
Pi value: 3.141593
Time taken: 0.006832 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 32 dynamic
Pi value: 3.141593
Time taken: 0.007580 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 32 guided
Pi value: 3.141593
Time taken: 0.005884 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 32 auto
Pi value: 3.141593
Time taken: 0.005795 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 32 static
Pi value: 3.141593
Time taken: 0.006191 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 64 dynamic
Pi value: 3.141593
Time taken: 0.006937 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 64 guided
Pi value: 3.141593
Time taken: 0.005990 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 64 auto
Pi value: 3.141593
Time taken: 0.006724 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 64 static
Pi value: 3.141593
Time taken: 0.005939 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

```
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 128 dynamic
Pi value: 3.141593
Time taken: 0.005975 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 128 guided
Pi value: 3.141593
Time taken: 0.006140 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 128 auto
Pi value: 3.141593
Time taken: 0.006116 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1# ./piSeriesNaive 14000000 8 128 static
Pi value: 3.141593
Time taken: 0.006716 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio1#
```

De manera que el mejor block size sin duda por velocidades en general más rápidas fue el de 64, sin embargo con las políticas de scheduling creo que en general dio los mejores resultados fue el de auto, probablemente tenga que ver con el tipo de problema que resuelve.

Ejercicio 2

A.

```
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.006290 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.006099 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.006040 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.005178 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.005282 seconds
```

Por lo que se puede ver, se tuvo un leve mejor rendimiento entre esta versión y la del inciso hay que en promedio los tiempos son levemente más rápidos, sin embargo, no es tan notorio esto, probablemente a que solo fue una instrucción la cambiada. Sin embargo, se puede denotar que esta fue una mejora en términos de rendimiento.

B.

```
root@b7b74964d27a:/parallel_computing/Ejercicio2# gcc -o piSeriesNaive -fopenmp piSeriesAlt.c -lm -O2
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.004494 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.002520 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.003774 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.002245 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2# ./piSeriesNaive 14000000 8
Pi value: 3.141593
Time taken: 0.002576 seconds
root@b7b74964d27a:/parallel_computing/Ejercicio2#
```

Se denota que es superior, muy superior al utilizarse la flag O2 y probablemente mejoraría con la flag O3, esto es debido a que la flag O2 lo que permite es que se alineen las funciones en memoria para mejorar la eficiencia de caché, mejorar el rendimiento de ramas en las instrucciones de salto, alinear etiquetas en la memoria, alinear bucles para mejorar el caché, así mismo, permite que se usen registros de propósito general en vez de mantener registros en un apila, mover el código fuera de bucles para evitar cálculos redundantes, eliminar instrucciones redundantes. En general, la flag permite optimizar la ejecución del código al sacar el máximo provecho de los recursos existentes utilizando memoria más rápida como lo es la caché y optimizando instrucciones para que no sean redundante.