import multiprocessing as mp

import time

import math

def fib(n):

    if n < 0:

        print("Error de introducción")

    elif n == 1:

        return 0

    elif n == 2:

        return 1

    else:

        n\_cores = mp.cpu\_count()

        size\_array = math.ceil(n / n\_cores)

        MC = mp.RawArray('i', n)

        cores = []

        for core in range(n\_cores):

            i\_MC = min(core \* size\_array, n)

            f\_MC = min((core + 1) \* size\_array, n)

            cores.append(mp.Process(target = par\_core, args = (n, MC, i\_MC, f\_MC)))

        for core in cores:

            core.start()

        for core in cores:

            core.join()

        return MC[n]

def par\_core(n, MC, i\_MC, f\_MC):

    for i in range(i\_MC, f\_MC):

        MC[i] = fib(n - 1) + fib(n - 2)

if \_\_name\_\_ == "\_\_main\_\_":

    n = 21838989

    inicio = time.time()

    print(fib(n))

    fin = time.time()

    print('Tiempo de ejecucion SECUENCIAL = ', fin-inicio)

Tras utilizar parte de los dos códigos utilizados en las clases de PCD, hemos comprobado que al utilizar en numero de expediente de nuestra universidad y el numero de cores que utiliza nuestro ordenador, tras ejecutar el código que se ve adjuntado, no se permite desarrollar el paralelismo debido a que al utilizar un numero de mas de 6 cifras Visual Studios queda inhabilitado y se bloquea la pantalla.