**Q1. What resources are used when a thread is created? How do they differ from those used when a process is created?**

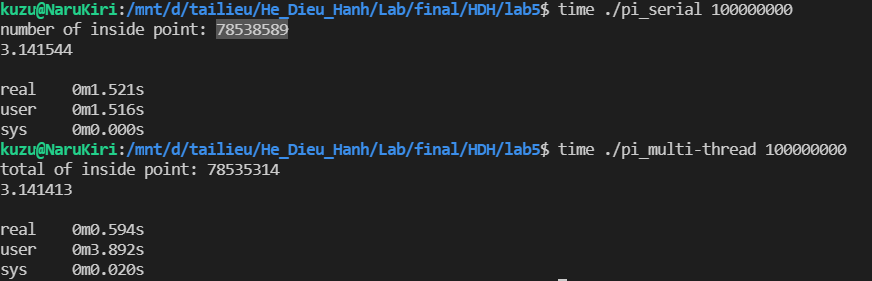
Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority. Further, all thread in the same process shares memory and resources of the process by default, and doing context-switch between threads is not costly as process. Overall, Thread creation consumes less time and memory than process creation.

**Q2. Is it possible to have concurrency but not parallelism? Explain.**

It’s possible. Actually, concurrency and parallelism are two different concepts: concurrency allowing all the tasks to make process, each task perform for a small time and then give back CPU for other process. In contrast, a parallel system can perform more than one task simultaneously. That mean, with context-switching of process or thread, we can perform concurrency, creating the illusion of parallelism by rapidly switching between processes

**Exercise**

**Problem 1**



Có thể thấy chương trình multi-thread tính pi chạy nhanh hơn so với chương trình serial tính pi, với cùng 1 lượng input.