Parallel Programming

Part A

1. **Identifying the components on the raspberry PI B+**

CPU/ RAM, Ethernet, USB, Ethernet Controller, Display, HDMI, Camera,

1. **How many cores does the Raspberry Pi’s B+ CPU have**

Raspberry Pi’s B+ CPU has 4 cores

1. **List three main differences between X86 (CISC) and ARM Raspberry PI (RISC). Justify you answer and use your own words (do not copy and past)**

A few of the main differences between x86 and ARM are that: 1) x86 is a CISC (complex instruction set computing) which allows the x86’s processor to have a bigger feature rich instruction set that allows multiple ways to access the memory, but it has far less registers than ARM. 2) ARM is a RISC (Reduced instruction set computing) which is a much more simplified instruction set and has more general registers that use load/store to access memory. 3) Most instruction used in ARM programming can be used for conditional execution; Intel x86’s processors use the little-endian format

1. **What is the difference between sequential and parallel computation and identify the practical significance of each?**
   1. Sequential computations processes are executed in a continuous/ ordered process one after the other, while the parallel computation processes are executed simultaneously.
   2. There is no need to change a sequential computation when using and if it is being used in different modules it does not require any communication at the module interfaces. Parallel computation on the other hand has the ability to be converted to a sequential composition so that it can enhance scalability and locality.
2. **Identify the basic form of data and task parallelism in computational problems.**

Data parallelism uses multiple processing elements to solve problems by breaking the problem into independent parts. Task is a form of parallelization of computer code across multiple processors in the parallel computing environments.

1. **Explain the differences between processes and threads.**

The difference between processes and threads, processes is the abstraction of a running program, and they do not share a memory with each other. A thread is a lightweight process that allows a single executable/process to be decomposed to smaller, independent parts. Every thread shares common memories of the processes they belong to.

1. **What is OpenMP and what is OpenMP pragmas?**

OpenMP is a library/language. OpenMP pragmas are compiler directives that enable the compiler to generate threaded code.

1. **What applications benefit from multi-core (list four)?**

Database servers; Web servers; Compilers; Multimedia applications; Scientific applications; In general, applications with thread-level parallelism

1. **Why Multicore? (why not single core, list four)**

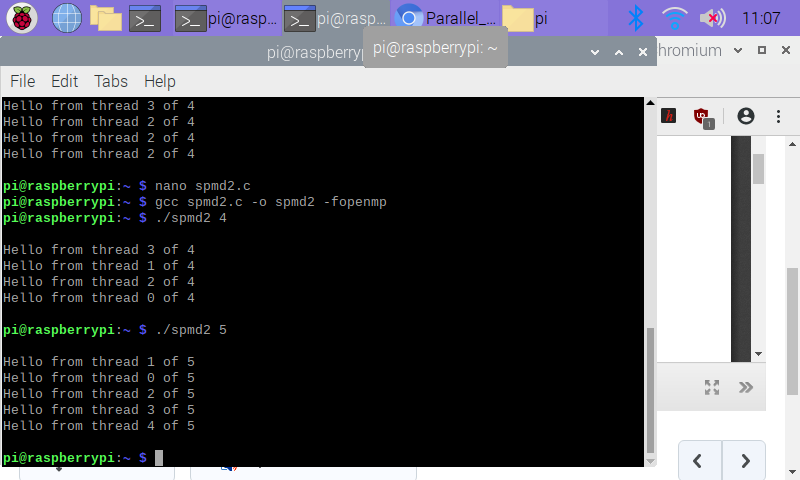
Multicore is better because it is difficult to make single-core clock frequencies higher, and many new applications are also multithreaded, which currently is a trend in computer architecture

Part B

A screenshot of a computer

Description automatically generated

This image shows the threads sharing the same fork/thread because the code is not correct. Also, the threads are not in order, even if they share threadsA screenshot of a cell phone

Description automatically generated 

The images above show the corrected code and the result after the threads are printed out. The threads now have independent fork/ threads, but are still out of order because once a thread is finished and printed, the order does not matter.