|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Assignee Name** | **Email** | **Task** | **Duration (Hours)** | **Dependency** | **Due Date** | **Notes** |
| Frankie Sanchez  (Coordinator) | fsanchez4@student.gsu.edu | Organize & facilitate team meetings; Create the Work Breakdown Rubric; Assist with drafting Task 4 – Part A: Foundation of Parallel Programming Skills | 3 hours | None | 10/1/18 | Meeting #1: 9/27/18 (2.5 hours)  -Discussed roles & responsibilities  -Set deadlines for tasks & outlined our review period  Meeting #2: 10/1/18 (0.5 hours)  -Recorded videos, discussed updates & deadlines  Review 10/4/18 |
| Phillip King | pking17@student.gsu.edu | Assist with drafting Task 4 – Part A: Foundations of Parallel Programming Skills; Task 4 – Part B:  Raspberry Pi Installation & Setup; TA Slack invitation | 6 hours | Raspberry PI hardware & the tutorials contained in the assignment | 9/28/18 | Installed Raspberry Pi 3 B+ and system image;  Flashed to MicroSD;  Verified installation |
| Angelica Amis | aamis1@student.gsu.edu | Task 3 – Record and upload YouTube video to Team Nine30 channel; Assist with Task 5 – Technical Writing (Appendix) | 2 hours | YouTube; Team member involvement for the video recording session; Information needed for report appendix | 10/1/18 | Recorded videos on 10/1/18 after class |
| Sai Pavan Nuthalapati | snuthalapati@student.gsu.edu | Create & maintain GitHub as outlined in Task 2: Collaboration; Assist with Task 5 – Technical Writing (Body of Report) | 3 hours | GitHub; Rough draft of report | 10/1/18 (GitHub); 10/4/18 (Report) | Periodically maintain & update the work progress on GitHub |
| Noah De Joya | ndejoya@student.gsu.edu | Programming the PI according to Task 3 – Parallel Programming Basics; Assist team with Task 5 – Technical Writing | 3 hours | Raspberry PI hardware & the tutorials contained in the assignment | 10/4/18 | Created detailed lab report to describe observations of parallel programming |

**Planning and Scheduling:**

**Parallel Programming Skills - Foundation:**

1. Identify the components on the Raspberry Pi 3 B+.

The components on the Raspberry Pi 3 B+ consist of:

* Power input
* HDMI output
* Power button
* 1GB LPDDR2 SDRAM
* Broadcom Videocore-IV
* Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
* Dual-band 802.11b/g/n/ac Wi-Fi
* Bluetooth 4.2
* RJ-45 100 base Ethernet port
* 4xUSB 2.0 ports
* Ethernet Controller chip
* CPU/RAM chip
* Display header
* Camera header
* 40-pin GPIO header
* 3.5fmm analogue audio-video jack

1. How many cores does the Rasberry Pi’s B+ CPU have?

The Raspberry Pi’s B+ CPU has a total of 4 cores.

1. List four main differences between x86 (CISC) and ARM Raspberry Pi (RISC).

The four main differences between CISC and RISC are:

1. RISC processors use simple instructions which only take an average of 1.5 clock cycles to run whereas CISC has more complex instructions which take, on average, anywhere from 2 to 15 clock cycles to run.
2. RISC has multiple register sets whereas CISC only has one register set.
3. Decoding of instructions on RISC is simple compared to CISC.
4. RISC does not have a memory unit, but CISC does.
5. What is the difference between sequential and parallel computation? Identify the practical significance of each.

In sequential computation a problem is partitioned into a series of discrete instructions that are executed one at a time on a single processor. The instruction execution is comparable to a queue of instructions waiting to be executed by the processor. Sequential computation was prevalent in older generation mainframes, workstations, and single processor PCs bounded by the Von Neumann architecture. In parallel computation a problem is solved with the synchronized utilization of multiple processors (compute resources) with each executing a piece of the instructions required to solve the problem. In essence, the problem is partitioned into discrete parts that can be solved simultaneously. Each discrete part is further split into a series of instructions that can execute concurrently on different processors with the use of a control mechanism. Once the problem is partitioned into differing sets of instructions, these instructions are executed by simultaneously by the various processors. There are many practical applications today where parallel computation offers many benefits. Some examples include: multithreaded applications, multimedia applications, database servers, compilers, and many more.

1. Identify the basic form of data and task parallelism in computational problems.

Data parallelism in computational problems focuses more on the input data as opposed to the functions to be performed. By focusing on the data, the same computation is applied to multiple data items. From an algorithmic standpoint, this allows for parallelism to be proportional to the input size of the data. This lends itself a solid foundation on which to scale efficiently. Task parallelism focuses more, and is organized around, the functions to be performed on the data rather than the data itself. The functions are decomposed to balance the work and ensure that the results are representative of the work required. Task parallelism is prominent when data must be evaluated through different contextual means. Task parallelism does not scale as well as data parallelism.

1. Explain the differences between processes and threads.

A process is essentially a program in execution. Processes involve “larger” tasks such as the execution of applications and is stored in main memory. Individual (independent) processes do not share memory with each other whereas threads share the memory of the processes of which they belong. A thread is merely a subset of a process—a lightweight process that allows a single executable/process to be decomposed into smaller, independent parts. A thread executes within the context of a process and shares the same resources.

1. What is OpenMP and what is OpenMP pragmas?

OpenMP is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran, on most platforms, instruction set architectures and OSs, including Solaris, AIX, HP-UX, Linux, macOS, and Windows. It consists of a set of computer directives, library routines, and environment variables that influence run-time behavior, and uses an implicit multithreading model (library creation and management). On the other hand, OpenMP pragmas are the compiler directives that enable a compiler to generate threaded code.

1. List four applications that benefit from multi-core.
2. Multimedia applications
3. Scientific applications such as CAD and CAM
4. Database servers
5. Compilers
6. Why multi-core? (List four)
7. Multiple instructions/commands can be computed simultaneously.
8. As we progress into the future, many new applications are multithreaded and there is a trend towards more parallelism in computer architecture.
9. At the same clock frequency, a multicore processor will process more data than a single core processor. Multicore processors handle more complex tasks utilizing lower energy than a single core processor. It’s difficult to make single-core clock frequencies even higher.
10. Deeply pipelined circuits are difficult to design, verify, implement, and house. Such tendencies conjure expensive production and maintenance costs.