Developing Soft and Parallel Programming Skills Using Project-Based Learning

Fall 2018

Coders3210

Steven Nguyen, Jeffrey Shu, Rabia Khan, Asier Yohannes, Aaron Morrison

Report:

Planning:

Name:	Email:	Task:	Comments:	Time Available:	Due Date
Steven Nguyen	snguyen36@student.gsu.edu	Create and manage Github, Create Written Report, Proofreading, check parallel programming answer		Tues/Thurs: 11AM Mon: 10:45 -12 Wed: 11:50 - 3:30 Fri: 9:15-12:30	Github/ Report outline 9/30/18 Finalize Report 10/4/18
Jeffrey Shu (Coordinator)	jshu3@student.gsu.edu	Setting up and coding Raspberry Pi, Video, editing video presentation		Mon 2-3:30 Tues 12PM Thurs 3PM	Set up 9/28/18 Coding 10/1/18
Rabia Khan	rkhan19@student.gsu.edu	Parallel Programming Skills (#1-3), Write Lab Report, Observing/Helping with the programming in Raspberry Pi		Mon 12:15-4:30 Tues/Thurs/Fri: free Wed 1:30 to 3:30	Questions 10/3/18 Lab report 10/4/18
Asier Yohannes	ayohannes1@student.gsu.edu	Parallel Programming Skills (#7-9)	**Needs to join Github!**	Mon 11-2 Tue after 3 Wed 11-2 Thursday after 3 friday :free	Questions 10/3/18
Aaron Morrison	amorrison16@student.gsu.edu	Parallel Programming Skills(#4-6)		**Mondays** between 11 AM and 3 PM **Tuesdays** after 12:15 PM **Wednesdays** between 12 PM and 3 PM **Thursdays** after 2 PM **Fridays** after 9:15 AM	Questions 10/3/18

Parallel Programming Skills:

1. Identify the components on the raspberry PI B+:

    DSI display port

    64 bit CPU/ 1GB RAM

    Power

    HDMI

    Camera port

    Power source

    100 Base Ethernet

    Ethernet Controller

    4 USB ports (each box has 2 ports)

2. How many cores does the raspberry PI B+ CPU have?

    There are four cores, since it's called Quad-core Multicore CPU.

3. List four main differences between x86 (CISC) and ARM Raspberry PI (RISC), justify your answer and use your own words

    1. The main difference is instruction sets. X86 is a Complex Instruction Set Computing processor which allows complex instructions to access memory, which means it has more operations and addressing nodes, since it has a larger instruction set. Whereas, ARM is a Reduced Instruction set Computing processor and has only a 100 or less instructions. Also, as compared to x86, ARM has more registers.

    2. ARM uses instructions that operate only on registers, and only Load/Store instructions can access memory since it uses a Load/Store memory model for memory. In X86, the Load/Store logic is inbuilt that's why it allows fewer instructions.

    3. Most instructions in ARM can be used for conditional execution whereas this is not true for X86.

    4. In Intel X86, little endian format is used, whereas in ARM, bi-endianness is used which has enabled for switchable endianness to exist.

4. What is the difference between sequential and parallel computation and identify the practical significance of each?

    In sequential (or “serial”) computing, a task is divided into a series of individual instructions and executed one at a time by a single processor. In parallel computing, a task is broken up into multiple parts, which are then broken up into multiple individual instructions that can be executed simultaneously by multiple processing cores. Parallel computing allows for much faster processing speeds at lower required clock rates per core, and thus less heat per core. However, some programs are difficult to design for parallel implementation.

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

    Data Parallelism involves one particular operation performed at the same time by different processors on subsets of a common larger portion of data. So, if, for example, 100 elements of an array need to be added together, instead of a processor adding each element with its subsequent element sequentially, the problem can be broken down into 4 smaller tasks of addition, dividing the execution time by 4 if working with 4 cores.

    Task Parallelism differs from Data Parallelism in that different tasks are executed in parallel by different cores (whether on the same data or different data) instead of the same task. Pipelining is a specific type of Task Parallelism in which a task is broken down into multiple smaller computations as needed and executed concurrently.

6. Explain the differences between processes and threads.

    A process is an abstract instance of a running program, as executed by a processor. Threads are essentially subsets of processes, lines of execution through a process that share common resources. A process may have one or numerous threads. Multiple threads allow for different parts of a process to be executed concurrently.

7. What is OpenMP and what is OpenMP pragmas?

    OpenMP is an application programming interface that supports multi-platform shared memory multiprocessing programming. OpenMP pragmas are compiler directives that enable the compiler to generate threaded code, using an implicit multithreading model.

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

    Database servers, Web servers (Web commerce), Compilers, and Multimedia Applications.

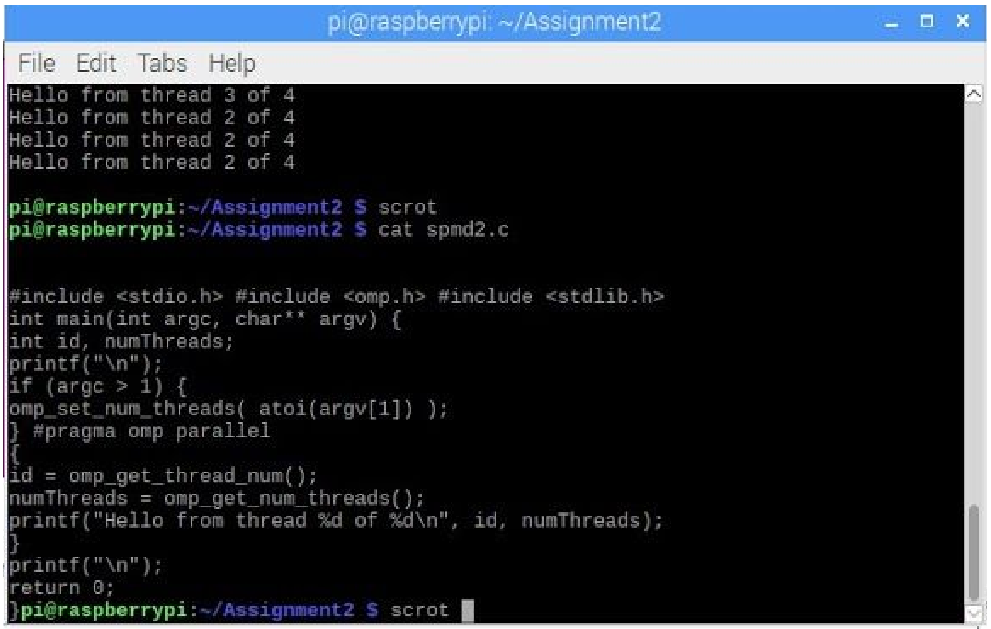
9. Why Multicore (why not single core, list four)?

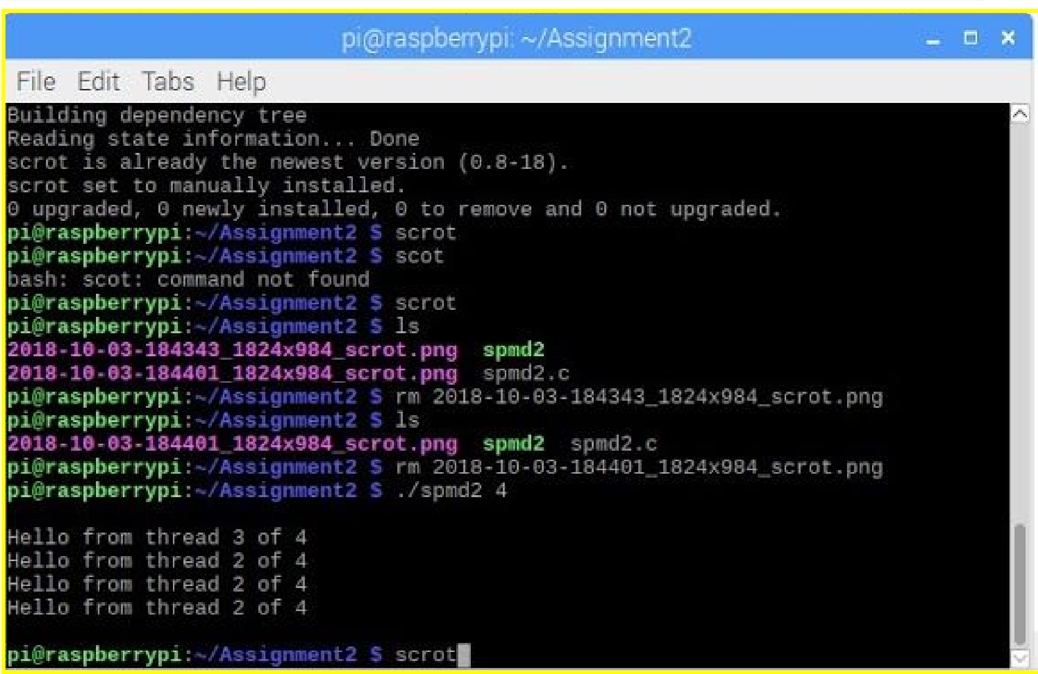
    It is difficult to make single-core clock frequencies even higher. Single core also has problems such as heat problems, difficult design and verification, and many new applications are multithreaded when they are made used for easier use.

Lab Report:

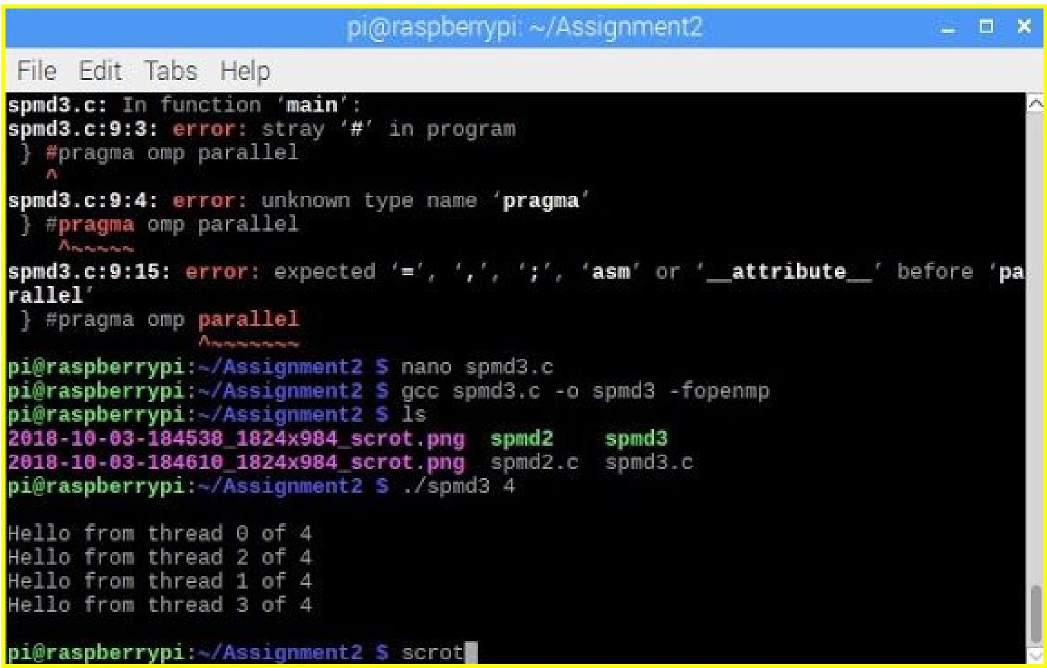
For the programming part of the lab, we had to type commands in the terminal window of the Raspberry Pi. The Raspberry Pi has a Linux-like operating system, called the Raspbian. If someone has already worked with Linux, it would be easy for them to work on the Pi. Once the

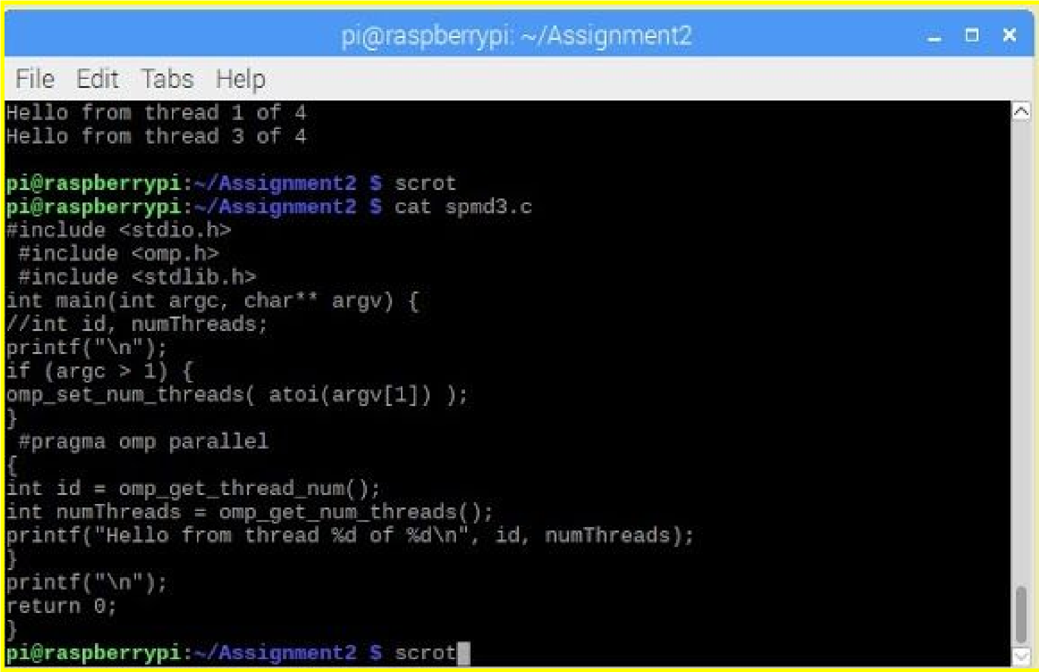
Pi was set up, we started the Pi and started to type in commands. The first set of codes that we were given, it produced four lines of output as shown below, but the thread ID numbers were repeated. We learned that the reason this was happening was because we declared two variables outside the block of code starting from line 11. The four cores in the Pi use the same bank of memory, and so the variables outside of the block were forked and run in parallel/separate threads, and all threads basically share each of the variable's memory location. This is why we saw thread ID number 2 repeated thrice as shown below.





The next step was to fix this issue, and to do that, we declared the variables int id and int numThreads inside the block starting on line 11 to line 15, as shown below. This way, each thread then had its own private copy of variables named id and numThreads. Consequently, we were able to prevent all threads sharing each of the variables location, and ended up producing unique thread ID numbers.
The thing that was interesting to learn in this entire task was how to use a fork-join programming pattern for parallel programs. The other pattern we learned was called “single program, multiple data” which means one can use one file could run parts of the code separately on a different data values stored in memory. Both of these patterns are built into OpenMP, which makes it easy to run some codes on multiple threads on different cores. Also interestingly, a four-core processor on the Raspberry Pi can use less than/more than 4 threads!





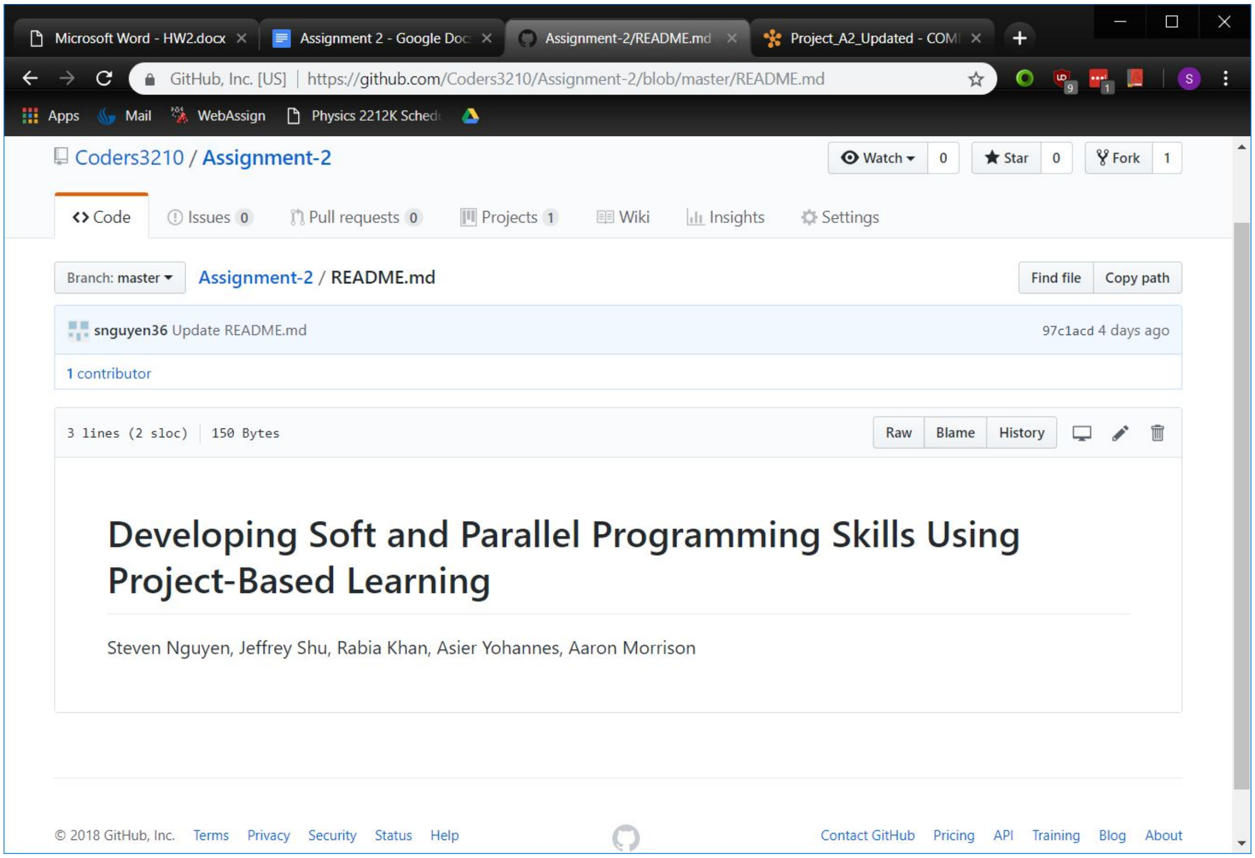
Appendix:

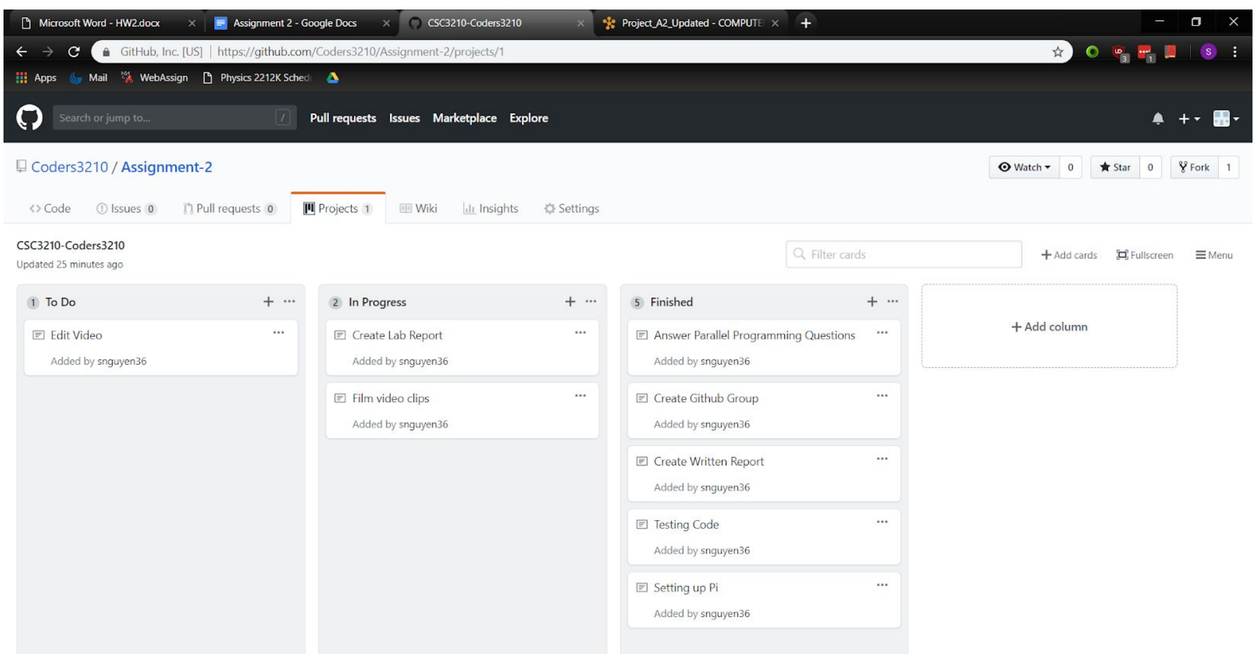
- Slack:
  - ➔ Coders3210.slack.com
- Youtube:

(NOTE: The video was recorded with all of us being recorded individually, however, we were all present there at the same place at the same time. Also while recording in the CURVE lounge we had to relocate to a different table due to a reservation conflict so a few members clips were recorded in a slightly different location).

Channel: https://www.youtube.com/channel/UCY3UyaKovGc2TEpr_QqfYRw
Video: https://www.youtube.com/watch?v=KsoktPdZRwM

- Github:
  - ➔ https://github.com/Coders3210
- Screenshot:


Microsoft Word - HW2.docx
Assignment 2 - Google Doc
Assignment-2/README.md
Project_A2_Updated - COM
GitHub, Inc. [US] | https://github.com/Coders3210/Assignment-2/blob/master/README.md
Apps
Mail
WebAssign
Physics 2212K Sched
Coders3210 / Assignment-2
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
Branch: master
Assignment-2 / README.md
Find file
Copy path
snguyen36 Update README.md
97c1acd 4 days ago
1 contributor
3 lines (2 sloc) 150 Bytes
Raw
Blame
History
Developing Soft and Parallel Programming Skills Using Project-Based Learning
Steven Nguyen, Jeffrey Shu, Rabia Khan, Asier Yohannes, Aaron Morrison
© 2018 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About


Microsoft Word - HW2.docx
Assignment 2 - Google Docs
CSC3210-Coders3210
Project_A2_Updated - COMPUTE
GitHub, Inc. [US] | https://github.com/Coders3210/Assignment-2/projects/1
Apps
Mail
WebAssign
Physics 2212K Sched
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Coders3210 / Assignment-2
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
CSC3210-Coders3210
Updated 25 minutes ago
Filter cards
Add cards
Fullscreen
Menu
1 To Do
Edit Video
Added by snguyen36
2 In Progress
Create Lab Report
Added by snguyen36
Film video clips
Added by snguyen36
5 Finished
Answer Parallel Programming Questions
Added by snguyen36
Create Github Group
Added by snguyen36
Create Written Report
Added by snguyen36
Testing Code
Added by snguyen36
Setting up Pi
Added by snguyen36
+ Add column