**The Assemblers**

**Spring 2019**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Developing Soft and Parallel Programming Skills Using Project-Based Learning**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Authors:**

**Aaja Christie**

**Davidson Fleurantin**

**Mamadou Diallo**

**Sheng Chen**

**Matthew Kabat**

**Planning and Scheduling**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assignee Name | Email  (@student.gsu.edu) | Task | Duration (hours) | Dependency | Due Date  (Feb 22) | Note |
| Aaja  Christie | achristie3 | Task 2  And Editing the video | 5 hours | Github  Slack  Youtube | Feb 21 |  |
| Davidson Fleurantin (Coordinator) | dfleurantin1 | Parallel programming foundations | 5 hours | ARM foundation | Feb 19 |  |
| Mamadou Diallo | mdiallo15 | Task 4 | 4 hours | Raspberry Pi and The Slides | Feb 19 | Take screenshots that explains the process |
| Sheng Chen | schen36 | Parallel programming Basics(Task 3, part b) | 5 hours | Raspberry pi emulator | Feb. 20 |  |
| Matthew Kabat | mkabat1 | Task 5: Edit Report | 2 Hours | Everyone else’s parts | Feb 22 |  |

**Parallel Programming Skills**

**Foundation**

**1.** **Identify the components of the raspberry PI B+**

The raspberry pi B+ consists of several components working at tandem. They are as follows:

**The CPU/RAM component**: it is the brain of the raspberry pi which helps in arithmetic, logical, and image calculations.

**The HDMI port**: it allows connectivity to television or compatible monitors.

**The MicroUSB Power Source**: it provides 5V power to the raspberry from an outlet.

**The USB ports**: 4 USB ports to connect peripheral devices such as mouse and keyboard.

**The SD card slot**: this is the location to place an SD card. The card needs to have an operating system installed.

**The Ethernet port**: this port allows for Cat6 cables or other internet wired cables. It enables WIFI connection.

**The Composite Video and Audio socket**: this socket is used to connect headphones, speakers. this port connects only output devices.

**The Camera port**: it connects small cameras to the processor.

**The 40-pin GPIO header**: input/output pins for external connection.

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

The raspberry pi 3 B+ CPU has a quad-core processor which means it has four independent cores. This quad-core processor can operate several processes at one time.

**3.** **List three main differences between X86 (CISC) and ARM Raspberry PI (RISC). Justify your answer.**

Complex Instruction Set Computing (CISC) and Reduced Instruction Set Computing (RISC) are two types of microprocessors. Intel has a CISC processor while ARM has RISC processor. As the name implies, CISC has a complex and more specialized set of instructions. A single instruction set can include many low-level operations such as mult, add, sub, etc. to access memory. It does these operations using less registers. CISC uses complex instructions in fewer lines of code. On the other end, RISC has more simple instructions to access memory. It uses more register sets to access memory. RISC uses simpler instructions in more lines of code. It uses Load/Store instructions to perform its tasks. With RISC, an instruction can execute in a single clock cycle, so the code executes more quickly, while in CISC, an instruction can utilize several clock cycles. However, CISC instruction sets can more efficient.

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

Sequential computation breaks down processes or tasks into several, smaller instructions. These instructions are processed only when the previous instruction is completed. Sequential or Serial computation process each instruction in a sequence. The first instruction is processed by the processor, then the second instruction comes along, get processed, followed by a third instruction. All the instructions derived from a task are executed by one processor. Whereas, in parallel computation, a task is broken down into multiple instructions, and these instructions are processed at the same time, using different processors. Therefore, multiple tasks are processed at the same time, reducing the time of execution. An Intel quad-core computer is ideal for parallel computation because it has four cores to process information.

Parallel computation is more suited for complex problems. The complex problem can be broken down in several units to be processed simultaneously. This would save time and resources. Therefore, multi-core computers utilizing parallel computing have a significant advantage to solve problems rapidly and efficiently. Serial computers represented the best way to process data in the past. At that time, they were used extensively in research. with the advance of parallel computing nowadays, serial computers have been in peril. Serial computation can be seen in minicomputers where simple tasks are needed.

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

Parallel computing is subdivided into data and task parallelism. In data parallelism, each core or processor in a multiple-core computer executes the same operation of the data set simultaneously. The tasks of a data set are distributed across the many processors. For example, if the instruction was to grade 400 exams, each core would grade 100 exams. Each core is doing the same job on its end to solve the global problem of grading 400 exams. On the other end, task parallelism, each core will perform different tasks to solve the global problem. This form of parallelism also makes use of multiple cores on the same computer or multiple computers. Different tasks or operations are executed at the time.

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

When a program is executed, the operating system produces a running program. This running program is the process. Different processes are independent of each other as each of them has a separate memory address space, which means they do not share memory. A thread can be thought of as a miniature process, or a unit of a process. A process can consist of one or more threads. Threads from the same process share a single memory space. There are multithreaded processes and single-threaded processes. If it is a single-threaded process, the process and thread can be considered as one. If it is a multithreaded process, many threads exist, and the process can perform different things at the same time.

**7.** **What is OpenMP and what is OpenMP pragmas.**

OpenMP is a set of directives used by some compilers (C, C++) to create multithreaded code or applications. It uses libraries to manage and handle the generation of threads (implicit multithreading), which in turn, makes parallel programming easier for programmers. OpenMP uses the shared-memory model to generate multithreaded processes. These threads share the same memory. OpenMP pragmas are specific directives a compiler uses to create threads.

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

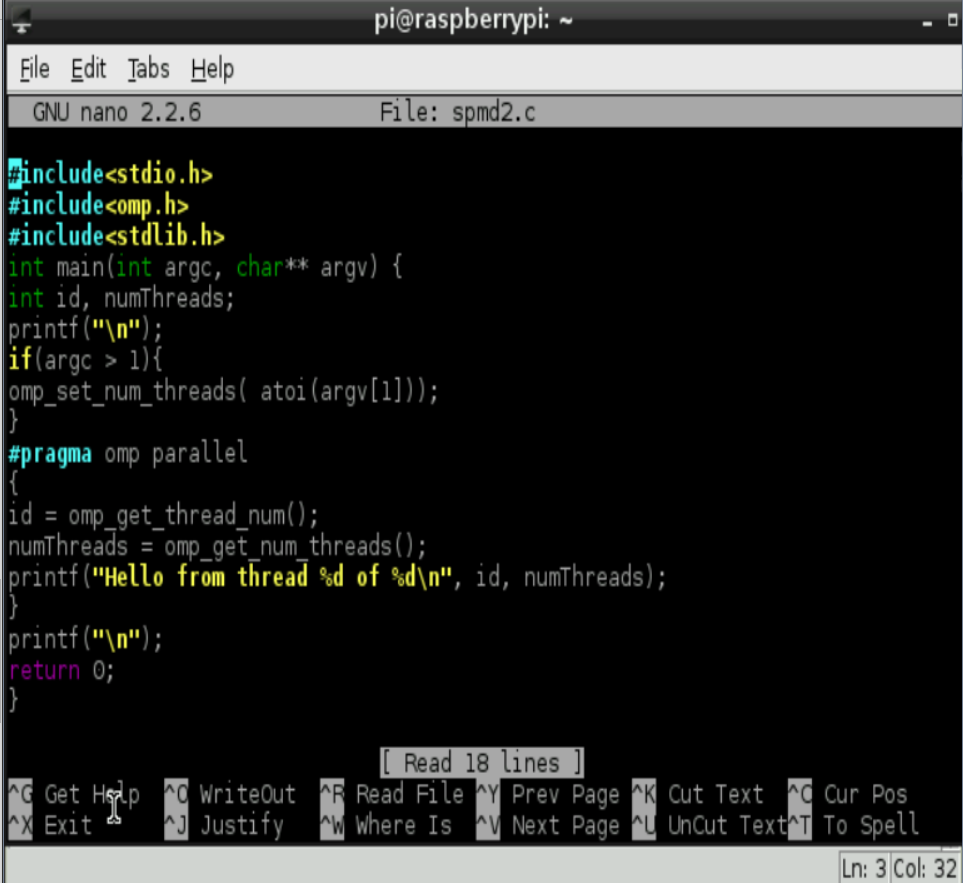
Some of the applications that benefit from multi-core include database servers, web servers, compilers and multimedia applications.

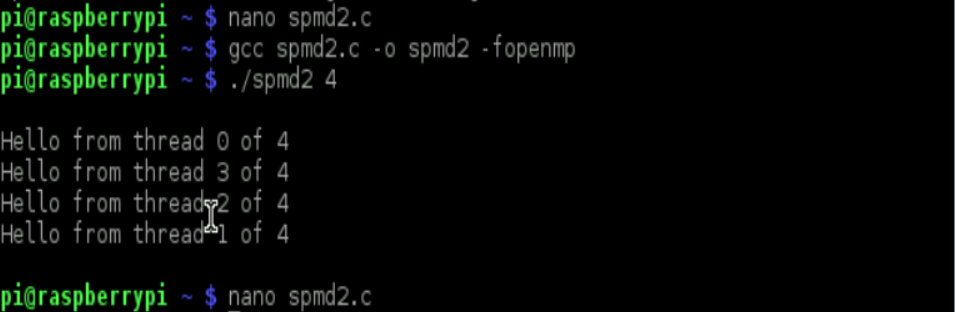
**9.** **Why Multi-core? (why not single core, list four)**

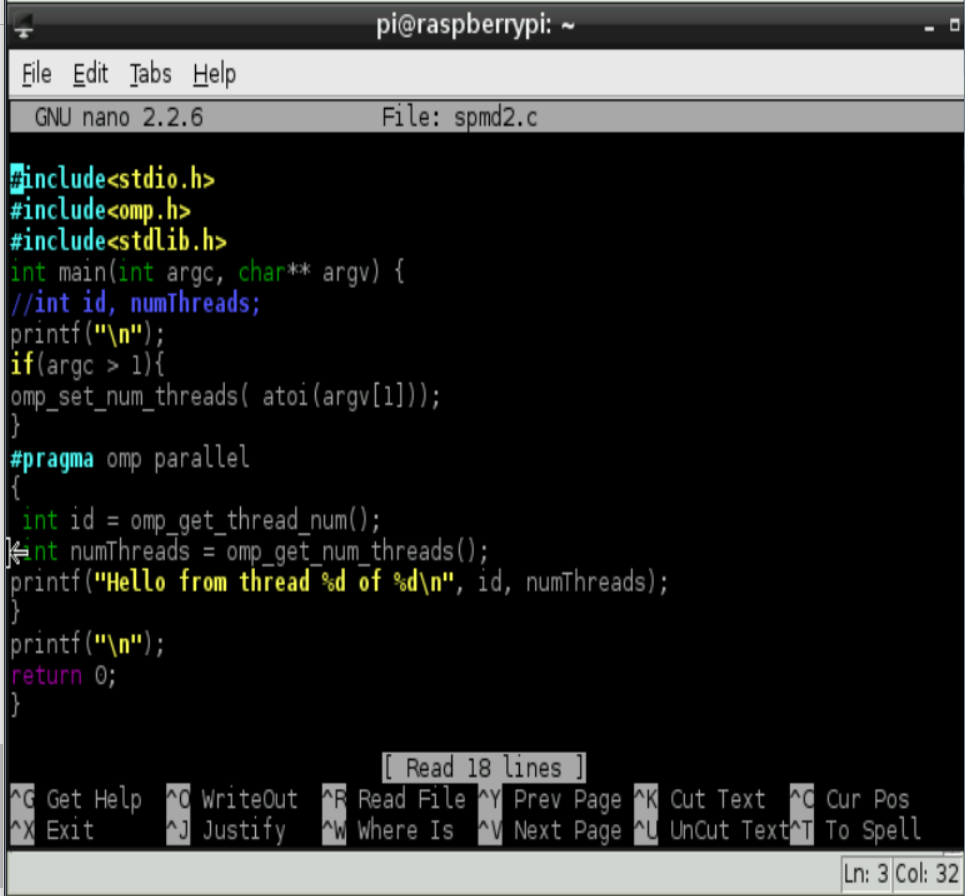
Multi-core processors are more sustainable in the long run. First off, it is difficult to increase clock frequencies or speed in single-core machines. The higher the clock frequency, the faster the processor. Secondly, Single-core processed are faced with drawbacks that are deeply pipelined in their circuits. These include heat problems, speed of light problems. They need a large team to design them. Thirdly, they need more expensive air-conditioning system maintenance. The market is moving toward more multithreaded applications so more multi-core processors are needed. Finally, the trend in computer architecture favors greatly parallel computation.

**Parallel Programming Basics**

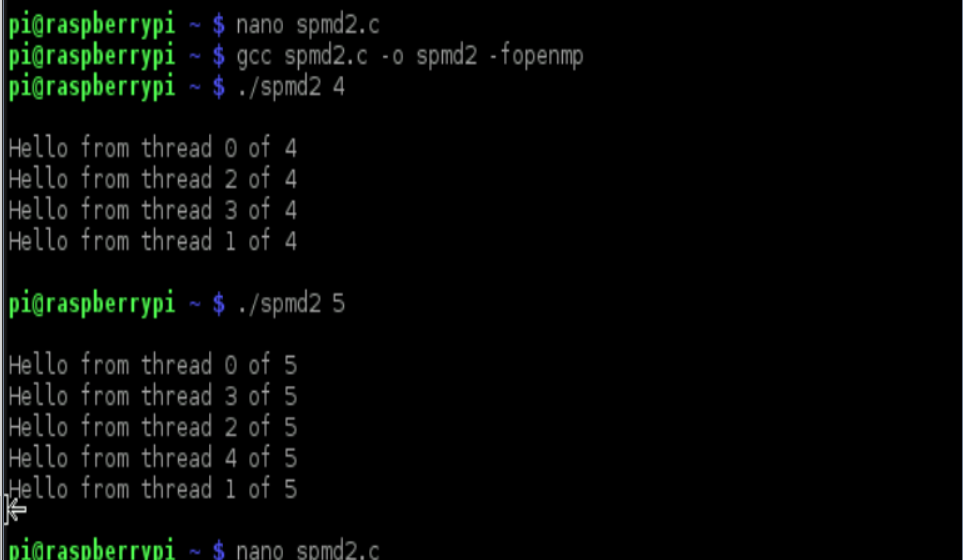
1. Writing the code and saving it as spmd2.c.



2. Creating the executable programming file by entering gcc spmd2.c -o spmd2 -fopenmp. And checking the output by typing ./spmd2 #(number of threads). 

3. Editing the line 13 and line 14 by adding “int” to make the each thread to be their own. 

4. Therefore, the output of the edited codes are still threads of 4, but they are in different orders since they have its own private copy.



**ARM Assembly Programming**

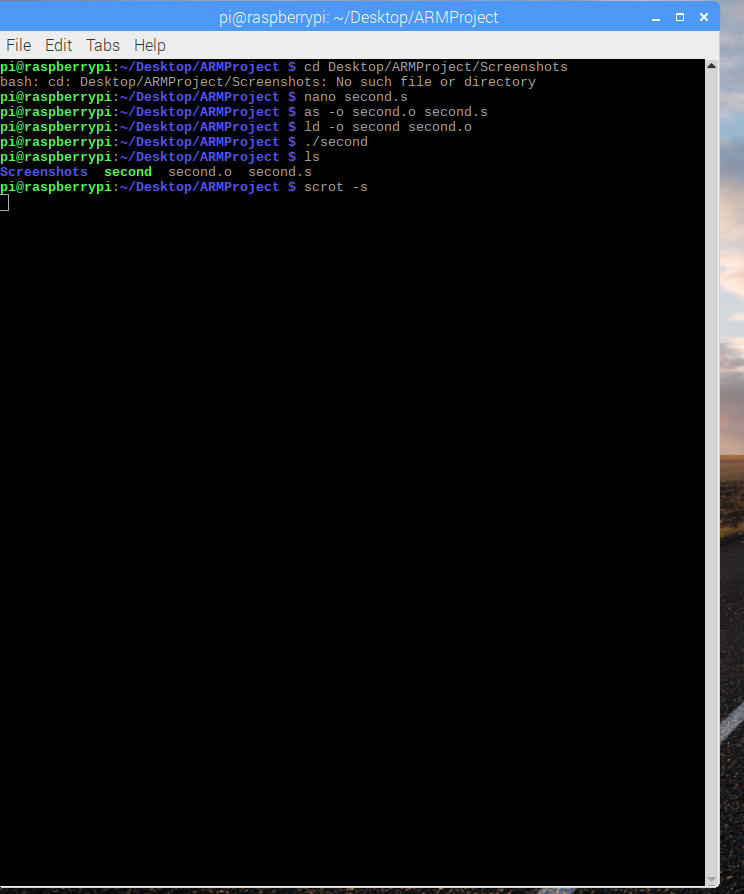
This is from step 1 to step 6 of part 1.

I created the second.s file, typed in the program, and then saved it.

Next I assembled the file using “as -o second.o second.s”

Then I linked it using “ld -o second second.s”

Although an objective file and an executable file were created, I wasn’t getting any output when I ran “./second”. That is because there is no input/output, meaning that I would have to use the debugger in order to follow what was going on in the program.

****

In the next step, I reassembled the file, this time adding the “-g” flag which will preserve the symbols and line numbers of the source code in the executable so that the debugger will be able to link in to the machine code.

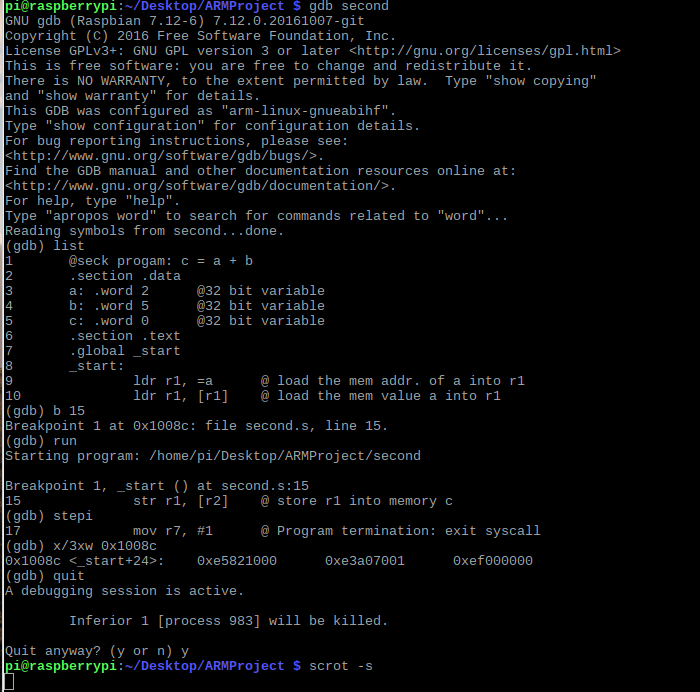
Then I the linked the object file “ld –o second second.o”, creating the executable.



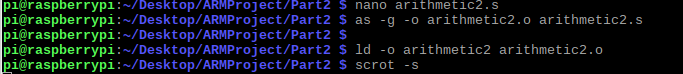
I used “gdb second” to launch the GNU debugger of second (the executable)

“list” allowed me to view the lines and its code

I typed “b 15” to set a breakpoint at line 15 then “run” to run the debugger



Here I created arithmetics2.s file. After typing the program, I assembled it with the debugging flag then linked the arithmetic2.o objective file that was assembled using “ld -o arithmetic2 arithmetic2.o”



Here I launched to debugger and used “list” to show me the code’s line numbers.

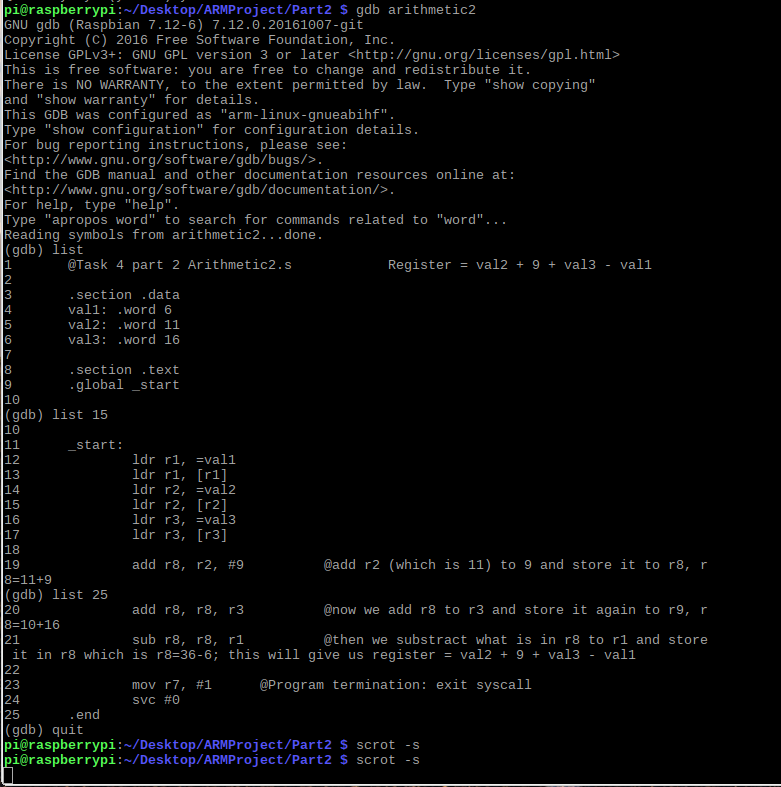
Inside the code I added comments what each line did from line 19 (the adding and subtractions)

Here I set a breakpoint at line 21 and stepped through once. Then I examined the registers to make sure I got the right answer.

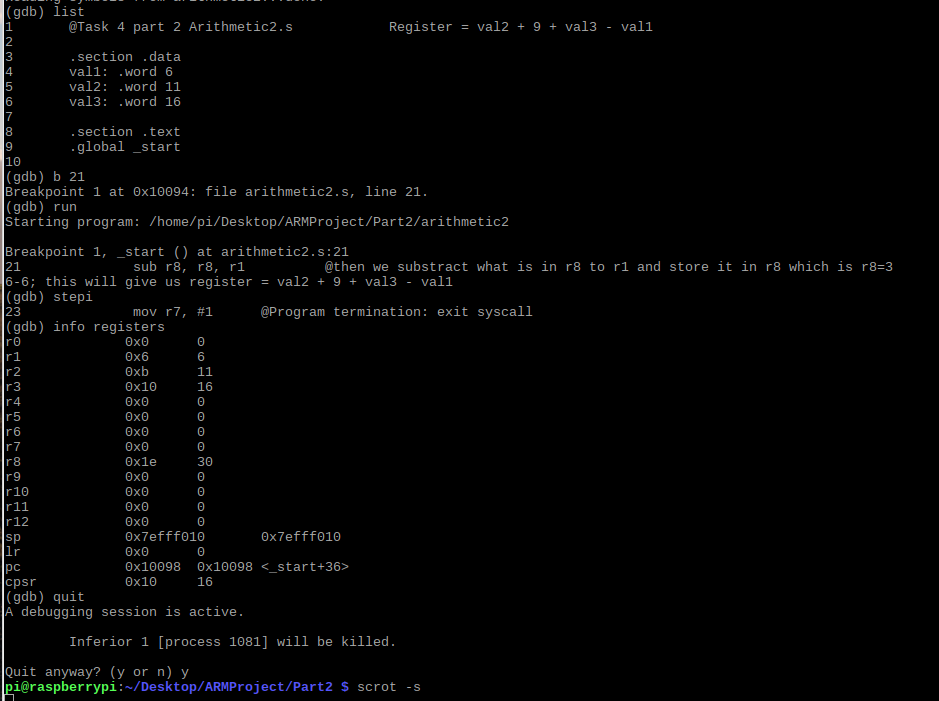
R = val2 + 9 + val3 – val1 R = 11 + 9 + 16 – val1 R should equal to 30.

Val1 was assigned in r1, val2 in r2, val3 in r3; I used r8 as the register to store the answer

The comments in the code explains the process.



Here I set a breakpoint at line 21 and stepped through once. Then I examined the registers to make sure I got the right answer using “info registers”.

****

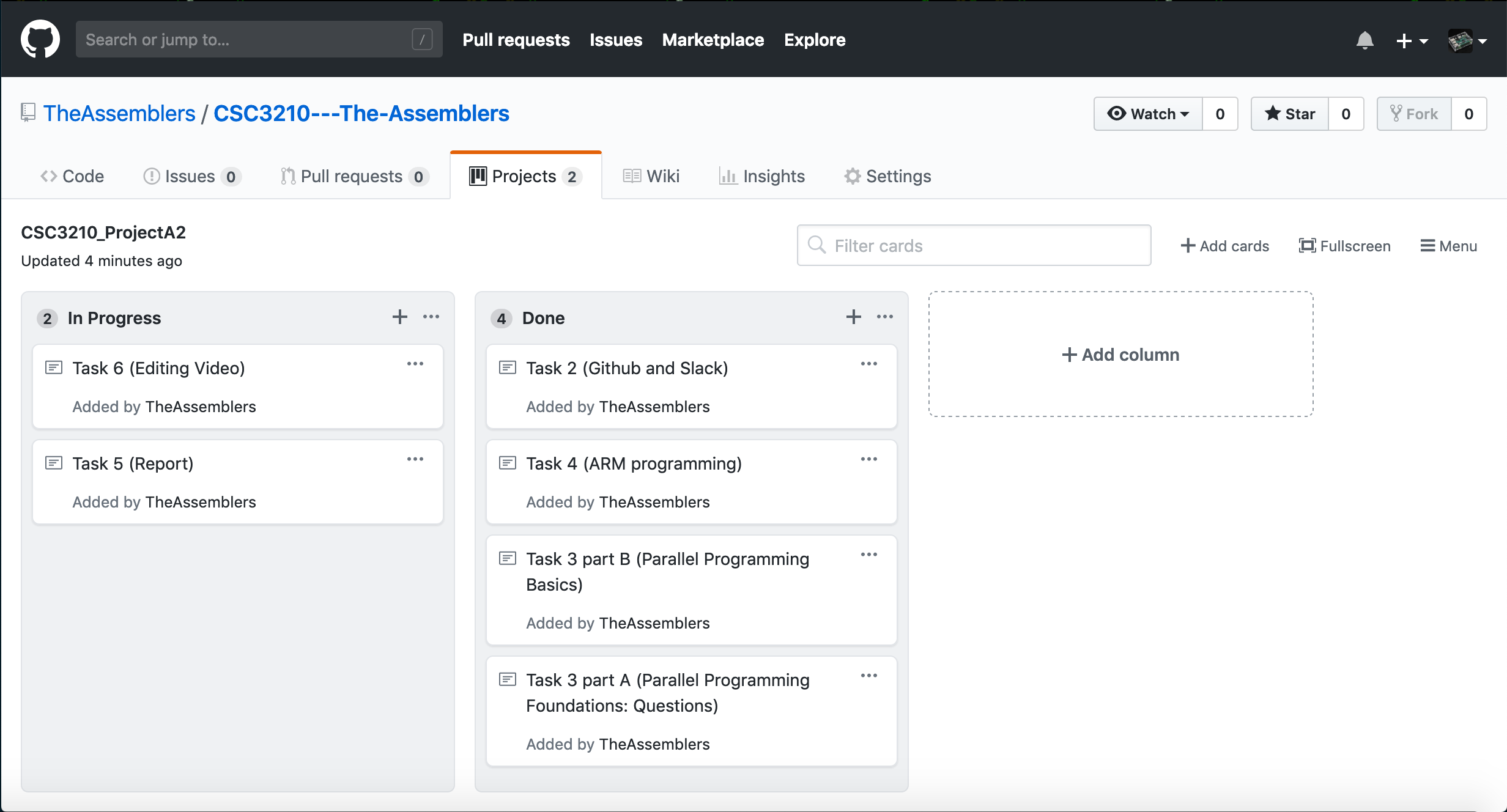
**Appendix**

Youtube Channel: [The Assemblers](https://www.youtube.com/channel/UCdmfZBj0qwSjBwdWWoW34iQ)

Assignment Video: <https://www.youtube.com/watch?v=1q2gdIrmS-4&t=8s>

Slack: <https://the-assemblers.slack.com/messages/CFSQ2GTDX/>

Github: <https://github.com/TheAssemblers>

****