Developing Soft and Parallel Programming Skills Using Project-Based Learning
Fall 2019
Team CHABU
Chris Lavey, Harsh Jivani, Anggiela Yupanqui, Binh Ha Duy Le, Ugonma Nnakwe

1. Planning and Scheduling

1.1 Team Name

We settled on the name "Team CHABU," which combines the first letters of each of our given names (Chris, Harsh, Anggiela, Binh, Ugonma).

1.2 Work Breakdown

Assignee Name	Email	Task	Duration (Hours)	Dependency	Due Date	Note
Harsh Jivani **(coordinator)**	hjivani1@student.gsu.edu	Slack, ARM Assembly Programming, Pushing files to GitHub	3	Raspberry OS installation	09/15/2019	Slack must be done before.
Chris Lavey	clavey@student.gsu.edu	Technical writing (getting the report ready) as described in assignment, ARM Assembly Programming	4	Raspberry OS installation, Slack, GitHub, and video. (these have to be done first)	09/15/2019	Must be ready at least 2 days early
Anggiela Yupanqui	ayupanquirojas1@student.gsu.edu	Setting up the GitHub	1	None	09/15/2019	GitHub must be done to add code later.
Ugonma Nnakwe	unnakwe1@student.gsu.edu	Video Editing, Create YouTube Channel	4	None	09/15/2019	Youtube channel needs to be done first.
Binh Le	ble8@student.gsu.edu	Setting up the Raspberry Pi, ARM Assembly Programming	2	None	09/15/2019	

2. Teamwork Basics

2.1. Maintaining Efficiency and Group Satisfaction

The easiest way for us to efficiently accomplish tasks while keeping each member's satisfaction is for us to get to know each other, including what areas we are strong and weak in. From there, we can set some ground rules for the group and communicate as regularly as possible. In addition, the coordinator of the group for each project will also act as a facilitator, which essentially means they are tasked with ensuring that everyone completes their tasks on time and handling conflicts that may arise.

2.2. Work Norms, Facilitator Norms, and Communication Norms
2.2.1 Work Norms

Work distribution will be decided by discussion with the group as a whole. If there are conflicts in interest, we will solve who completes which task using a vote. Deadlines will also be discussed when work is distributed, but the coordinator will be the one to set deadlines. In the event that someone does not follow through on their commitments, then that person will receive either partial or no credit for that assignment depending on how much of their task is complete. If this becomes a recurring issue, it will be discussed with the instructor and appropriate penalties will be issued. The incomplete task or tasks will then be given to the person that has the least workload for that assignment to try and finish before the submission deadline. Work will be reviewed regularly by the other members, and will be revised based on critiques if necessary. While some members may have different work ethics, as long as their tasks are completed before the deadline then there will be no penalty. Once they have reached their deadline, however, this may affect the tasks of other members and will be dealt with as described above.

2.2.2 Facilitator Norms

Our group plans to have each project's coordinator also act as a facilitator. Since the coordinator rotates with each project, this means that every member should be a coordinator and facilitator at least once. The responsibilities of the facilitator are to ensure that tasks are completed on time and to settle disputes that may occur within the group.

2.2.3 Communication Norms

For our group, we plan to primarily communicate through either Slack, phone calls, text messages, or e-mails. While we plan to have Slack, text messages, and e-mails be open as a form of communication at anytime, phone calls will strictly be during standard office hours, as any phone calls made before 9 A.M. or after 5 P.M. could interfere with the lives of other members.

2.3. How to Handle Difficult Behavior
2.3.1. A member of the group is too quiet:

If a member is too quiet or is not actively participating in group discussions, then the other members will make it a habit to try and get them more involved. They will try to ask them more questions about the topic at hand, or they will ask the member their opinion on how the current task should be completed.

2.3.2. A member of the team is argumentative or critical of alternative ideas:

If another member tends to be argumentative, then try to discuss with them whether or not their criticism is valid. If they believe that a certain solution

to a task may not be adequate, try and see if they are correct. If they are simply being critical of the ideas of the other members of the group, then the coordinator will discuss with them how this negatively affects the morale of the group as a whole.

2.4. How to Reach a Consensus

If there are conflicts of interest when it comes to making a decision, we will take a vote to determine which path the group should take. Since our group has an odd number of members, then a simple majority vote will usually be enough to solve a conflict. However if there are multiple choices for solutions to a task, then we may decide to take a multivote. This essentially means we would have each person vote on their top four choices, then have the group decide the best four options by majority vote. We then will discuss these four options before voting on the best two options. We will then repeat this process to decide which option the group will go with.

2.5. How to Ensure No Member Rushes the Group to Make a Decision

To ensure that everyone is able to voice their opinion on how to proceed during a particular assignment, if someone is ready to make a decision then the facilitator will check to see if everyone else in the group is also ready. If that is not the case, then we will discuss the topic further until everyone is able to decide how they wish to proceed. If we are unable to reach a consensus, then we will solve the issue using the methods described in **Section 2.4.**
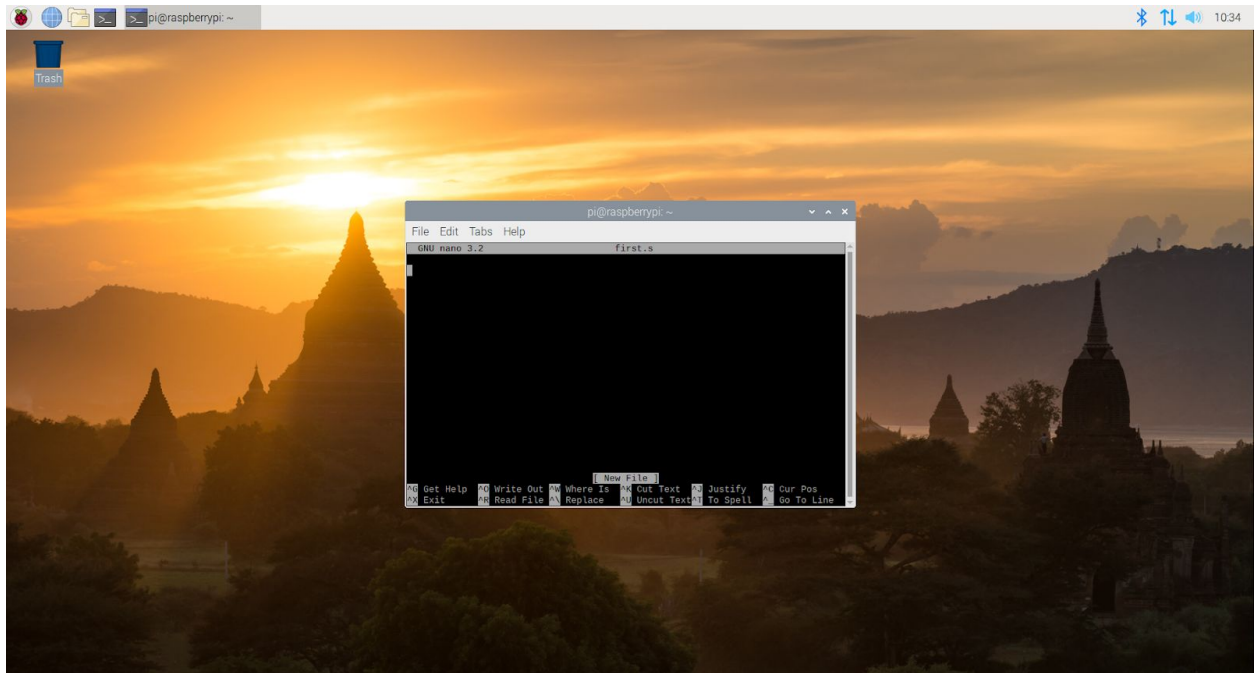
2.6. How to Handle Differences in Priority

Our goal as a group is to get an 'A' on every project. If this changes throughout the course of this semester, we may discuss it as a group or on an individual basis with the coordinator as necessary.

3. Raspberry PI Installation and ARM Assembly Programming

3.1 First Program in Assembly

For the first part of the programming portion of this project, we were tasked with setting up a Raspberry PI unit and doing some basic assembly programming in ARM. After installing the Raspbian OS and starting up the unit, we launched the terminal.

To save space and make the text written inside the terminal easier to read, all future images will be cropped to only display the terminal. We began by writing **first.s** using **nano**, which is a text editor for Unix-based devices. After finishing writing the command, we save the file to memory. The final code can be seen below:


```
@ first program
.section .data
.section .text
.globl _start
_start:
    mov r1,#5     @ load r1 with 5
    sub r1,r1,#1  @ subtract 1 from r1
    add r1,r1,#4  @ add 4 to r1

    mov r7,#1     @ Program Termination: exit syscall
    svc #0        @ Program Termination: wake kernel
.end
```

First.s is a very simple program. Its first task is to load 5 into register 1 using the **mov** command. It then subtracts 1 from the value stored in register 1 using the **sub** command, making the value stored in register 1 4. Its last command is to add 4 to register 1 using the **add** command, making the final value of register 1 8. To properly end itself, it moves 1 to register 7 to act as a system call to exit and calls the **svc** command to wake the kernel.

Once **First.s** was saved to memory, we then proceeded to assemble and link our program using two lines in the console:

as -o first.o first.s
ld -o first first.o

as assembled **first.s** and created the objective file **first.o**. **ld** linked the newly assembled file and created an executable file named **first**. We then ran our executable using the following command:
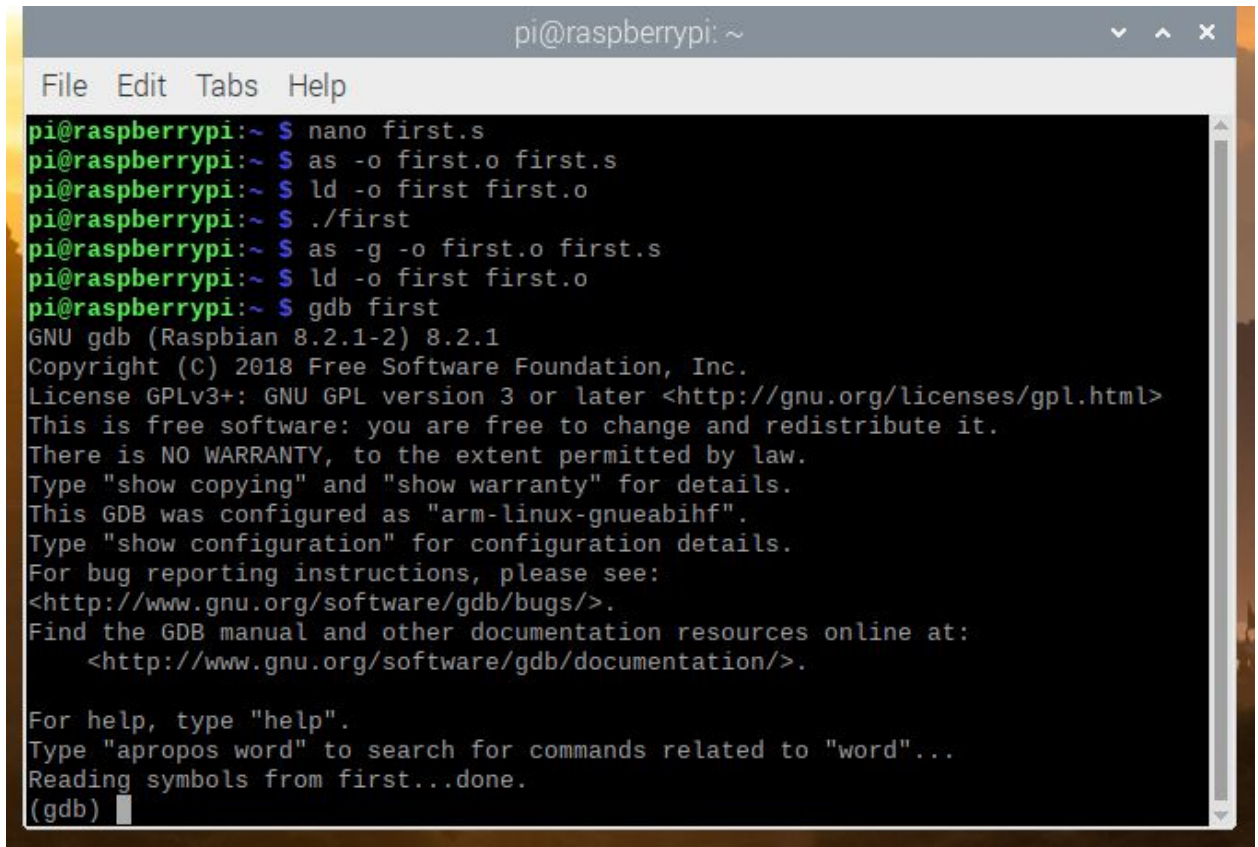
./first

This resulted in the console remaining empty. While we did modify the value stored in register 1, we did not print any output to the console, resulting in us not being able to verify if our program worked properly. To ensure that **first** works as intended, we decided to debug our program. To access our program using the debugger, we used the following commands in the console:

as -g -o first.o first.s
ld -o first first.o
gdb first

By using the **-g** flag when assembling **first.s**, we showed that we wanted to debug our program. Once we linked the program using **ld**, the debugger linked the machine code to source code line by line. We then launched the GNU debugger using **gdb** followed by the file we wanted to debug, which in this case was our executable **first**. Below is an image of the debugger upon initially launching:

```
                              pi@raspberrypi: ~                    v  ^  x
 File  Edit  Tabs  Help
pi@raspberrypi:~ $ nano first.s
pi@raspberrypi:~ $ as -o first.o first.s
pi@raspberrypi:~ $ ld -o first first.o
pi@raspberrypi:~ $ ./first
pi@raspberrypi:~ $ as -g -o first.o first.s
pi@raspberrypi:~ $ ld -o first first.o
pi@raspberrypi:~ $ gdb first
GNU gdb (Raspbian 8.2.1-2) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabihf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from first...done.
(gdb)
```
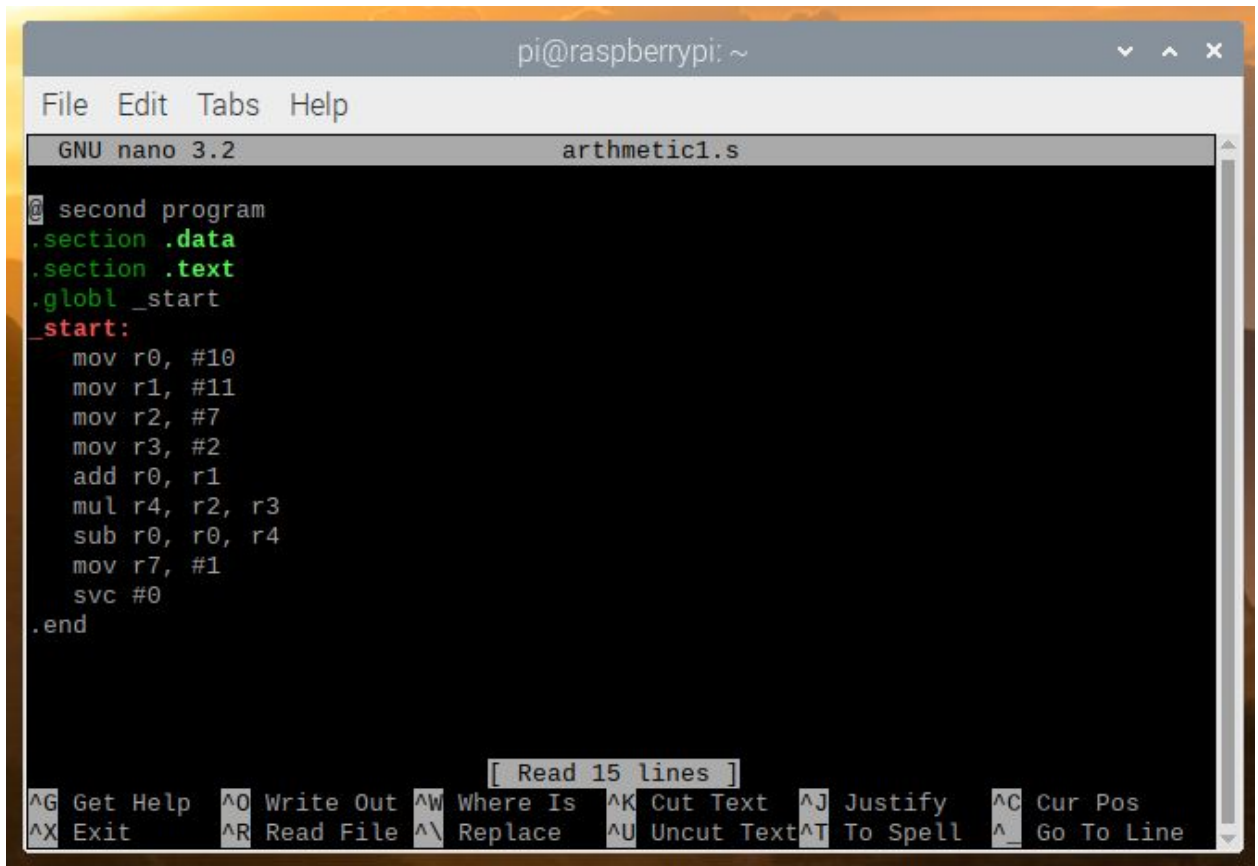
Once the debugger launched, we used the **list** command to show the first 10 lines of source code with a line number in front of each line. To see the values of our registers at the end of the program, we set the breakpoint using **b 11**, where line 11 was the last line in our program before it terminated. We then used the **run** command to have the debugger run **first** from start to the breakpoint. To see the values stored within the registers at this point in our program, we used the command **info registers**. This displays the register names, their values in hexadecimal format, and their values in decimal format. Below is a screenshot of the values stored within our registers at the end of the program:

This shows that the value stored in register 1 at the end of the program was 8. Since this was our goal, our program worked as intended and we were able to move on to the second part of the programming portion of this project.

3.2 Program for Basic Arithmetic

For the second half of the programming portion of this project, we were tasked with writing a program that does some basic arithmetic. We began the same way we did in part 1 by using **nano** to write our new program, **arthmetic1.s**. Below is a screenshot of the code:

This program begins by using **mov** to load 10 into register 0, 11 into register 1, 7 into register 2, and 2 into register 3. It then adds register 1 to register 0 using **add**, resulting in register 0 having a value of 21. It then multiplies register 2 with register 3 and stores the value into register 4 using **mul**, making register 4's value 14. While we would have liked to store this 14 into register 2, due to hardware limitations we had to use a third register. We then subtracted register 4 from register 0 using **sub**, making the final value of register 0 7. It ends the same way as **first.s**, which is to call the system to exit and wake the kernel.

Once we were finished writing our program, we saved it and then went on to assemble and link it. To do so, we used the following commands:

as -o arthmetic1.o arthmetic.s

ld -o arthmetic1 arthmetic1.o

Similarly to **first.s**, when we run the program using **./arthmetic1** we do not have any output in the terminal as we did not make the program print to the console. To ensure that our program worked as intended, we decided to debug it. To do so, we followed the same steps as in part 1 to debug **first**. That is, we used the following commands to assemble, link, and launch the program in the debugger:
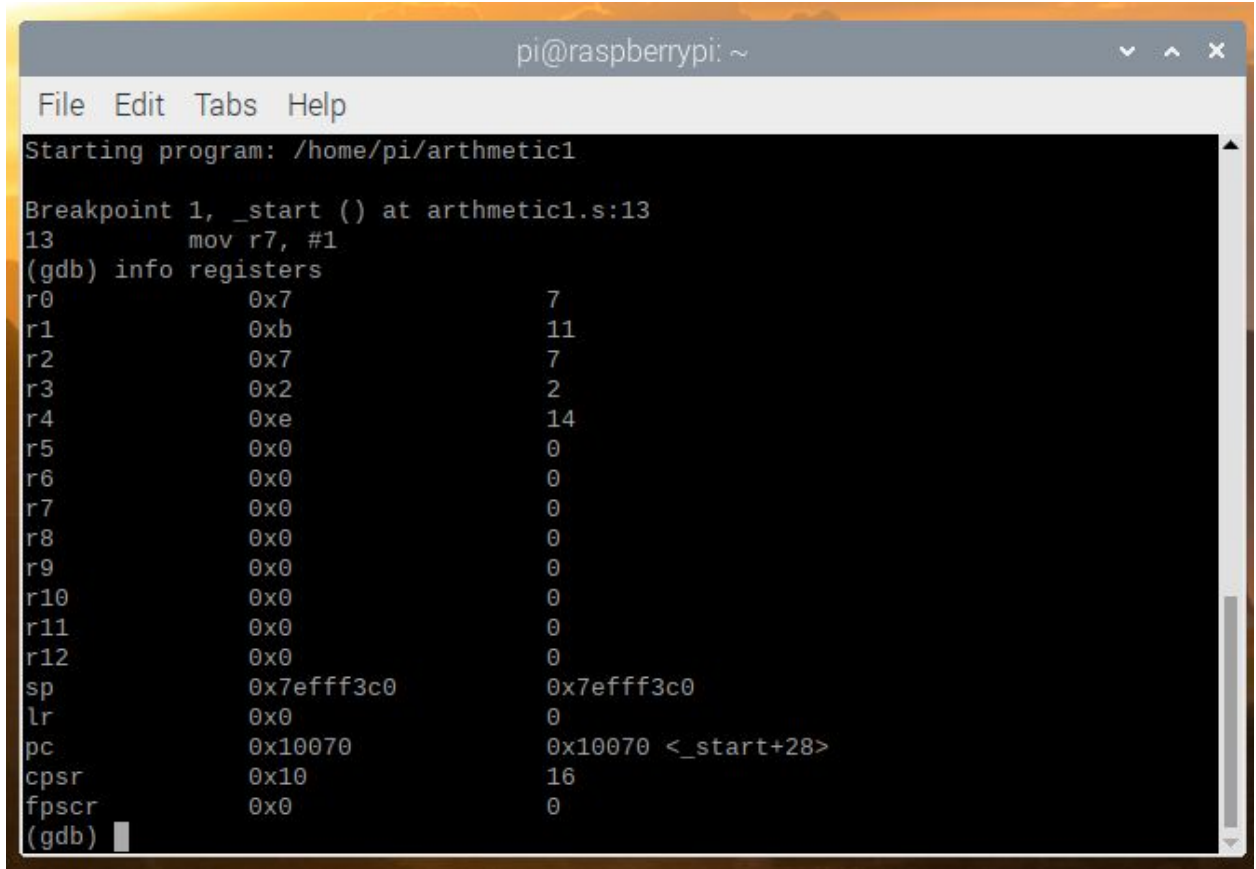
as -g -o arthmetic1.o arthmetic1.s

ld -o arthmetic1 arthmetic1.s

gnb arthmetic1

Once inside the GNU debugger, we used **list** to display the first 10 lines in the

program. We used **list** to display the next 10 lines until we found the right place to stop the program, which in this case was at line 13. To set a breakpoint at this line, we used **b 13**. Once we reached this point, we were ready to run the program, so we used the **run** command to begin debugging. Once running, we used **info registers** to display the values stored within the registers at line 13. Below are the values:

```
                              pi@raspberrypi: ~                        ⌄  ⌃  ✕

 File  Edit  Tabs  Help
Starting program: /home/pi/arthmetic1

Breakpoint 1, _start () at arithmetic1.s:13
13          mov r7, #1
(gdb) info registers
r0             0x7                      7
r1             0xb                      11
r2             0x7                      7
r3             0x2                      2
r4             0xe                      14
r5             0x0                      0
r6             0x0                      0
r7             0x0                      0
r8             0x0                      0
r9             0x0                      0
r10            0x0                      0
r11            0x0                      0
r12            0x0                      0
sp             0x7efff3c0               0x7efff3c0
lr             0x0                      0
pc             0x10070                  0x10070 <_start+28>
cpsr           0x10                     16
fpscr          0x0                      0
(gdb)
```

Through this info menu, we were able to see that the register values were what we intended them to be. At this point, we knew that our program worked as intended and that we had finished this portion of the project.

4. Appendix
4.1 Screenshots
4.1.1 Slack Introductions:

CSC 3210-Tea...
Harsh Jivani

Jump to...

Threads

Channels
arm-microprocessor
general
introductions
random

+ Add a channel

Direct Messages
♥ Slackbot
Harsh Jivani (you)
⚥ Anggiela
Binh Le
⚥ Chris Lavey
⚥ Ugonma Nnakwe

+ Invite people

Apps
● Doodle Bot

+ Install Google Drive
+ Add apps

#introductions
☆ | 👥 5 | 0 | 🔗 Add a topic

#introductions

You created this channel today. This is the very beginning of the **#introductions** channel. Purpose: All team member writes message with: name, interest, assigned task(s), and expectations from this project. (edit)

+ Add an app   👥 Add people to this channel

Today

Harsh Jivani 12:49 PM
joined #introductions along with 4 others.

Harsh Jivani 12:49 PM
Hello. My name is Harsh Jivani. I am a Junior and studying Computer Science with interests in Data Science. I am the coordinator for this Assignment 1 and my tasks are to create schedule for all members and assign the tasks for each member. I will also be doing ARM Assembly Programming for the Assignment 1. What I will get out of this project is to learn and enhance my soft skills as well as learn how to work along with my teammates. This is very important in the Computer Science industry to know how to work with others.

Chris Lavey 12:49 PM
Hello everyone. I'm Chris Lavey, a sophomore computer science student with an interest in data management and logic. For the first project, I've been tasked with compiling and summarizing the results of the project into a short technical write-up. Through this project I hope to establish foundational teamwork skills, and through future assignments I hope to develop a good understanding of assembly programming for ARM-based devices. (edited)

Ugonma Nnakwe 12:55 PM
Hello! My name is Ugonma Nnakwe, and I am a junior computer science major interested in Wed Design and UX Design. For Assignment 1, I have been tasked with setting up the Youtube channel for our group as well as filming and editing our presentation video. I hope I can learn more about the effectiveness of parallel programming, enhance my soft skills, and improve how I work with a team as we tackle all of these assignments together!

Anggiela 12:56 PM
Hello! I am Anggiela Yupanqui. I am currently a junior pursuing a computer science degree. For this project I have been assigned with the set-up of Github. I hope to develop a better understanding of this platform and learn a little more of how it works.

new messages

Binh Le 10:23 PM
Hello. My name is Binh Le. I am from Vietnam. I am a transfer student from Georgia Perimeter College. My major is Computer Science. For my personal time, I like to play with my camera. I have a Canon 6D and had taken many great portrait pictures with it. My task assigned in this project is Raspberry PI Installation and ARM Assembly Programming (Part 1). I have high expectations for this project. This is the first step for me to become familiar with the commands and the basic operations. The basic knowledge in this project will provide a solid foundation for me to complete the projects later.

Message #introductions

4.1.2 Github:

🖥 Team-Chabu / **CSC3210-CHABU**

👁 Watch ▾   1   ★ Star   0   ⑂ Fork   0

‹› Code   ⓘ Issues 0   Pull requests 0   Projects 1   Wiki   Security   Insights

No description, website, or topics provided.

🕒 **7** commits   ⑂ **1** branch   ◇ **0** releases   👥 **2** contributors

Branch: master ▾   New pull request   Create new file   Upload files   Find File   Clone or download ▾

hjivani1 Create code   Latest commit 61ba80b yesterday

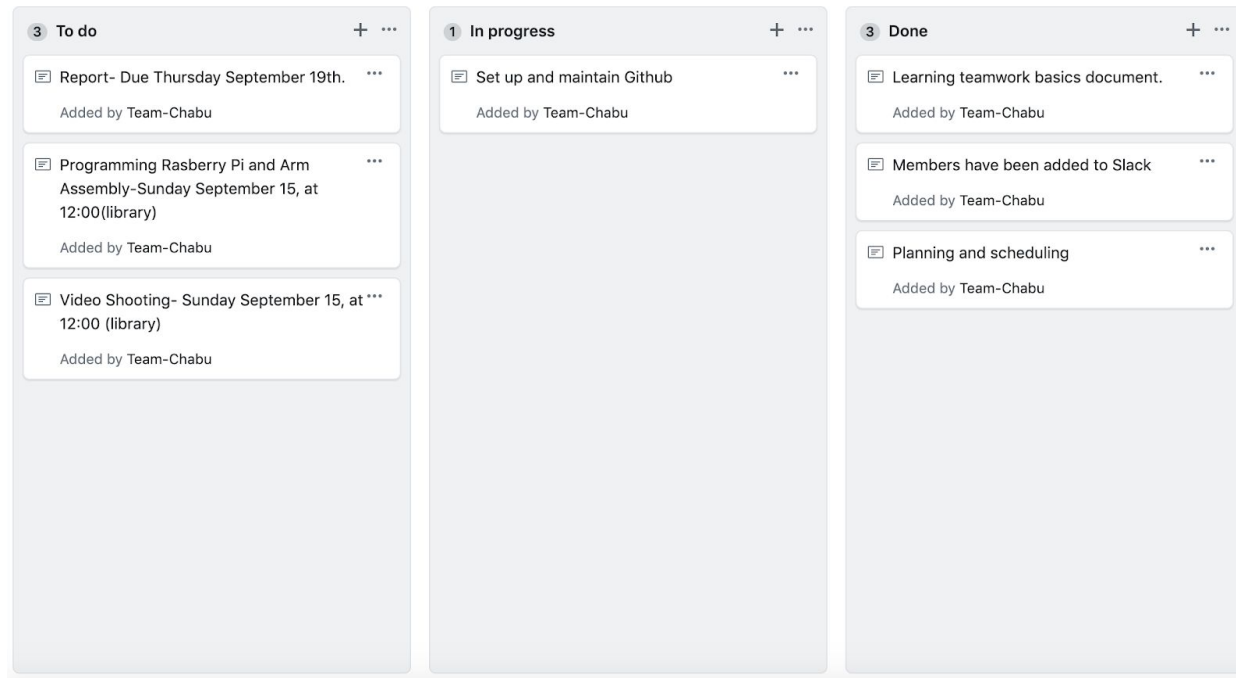📁 Assignment1   Create code   yesterday
📄 README.md   Update README.md   3 days ago

📖 README.md

CSC3210-CHABU

Team members:Anggiela Yupanqui, Chris Lavey,Harsh Jivani, Binh Ha Duy Le, Ugonma Nnakwe

3 To do	1 In progress	3 Done
📄 Report- Due Thursday September 19th. ⋯ Added by Team-Chabu	📄 Set up and maintain Github ⋯ Added by Team-Chabu	📄 Learning teamwork basics document. ⋯ Added by Team-Chabu
📄 Programming Rasberry Pi and Arm ⋯ Assembly-Sunday September 15, at 12:00(library) Added by Team-Chabu		📄 Members have been added to Slack ⋯ Added by Team-Chabu
📄 Video Shooting- Sunday September 15, at ⋯ 12:00 (library) Added by Team-Chabu		📄 Planning and scheduling ⋯ Added by Team-Chabu

4.2 Links

4.2.1 Github:

https://github.com/Team-Chabu/CSC3210-CHABU

4.2.2 Slack:

https://join.slack.com/t/csc-3210-teamchabu/shared_invite/enQtNzUyMzYwNzY
3ODI4LTQ2ZjY5NzM2ZGYwNmEzZGY2ZDc0MzAyZWIwY2M2NDA1Mzg1
MWE4MTUwMDRiNWIzYjA3ZWU3MjU5YTExYzQ2YWI

4.2.3 Youtube (Channel):

https://www.youtube.com/channel/UCp-0kVA08dU4F79t6slHtLQ

4.2.4 Youtube (Video):

https://youtu.be/3j_kYoFv5xc