DEVELOPING SOFT AND PARALLEL PROGRAMMING SKILLS USING PROJECT-BASED LEARNING,

(FALL 2019) ***THE 6-PACK:***

TEAM MEMBERS:

JENNIFER VU

ABRAHAM MAMMEN

RACHID BODSON

BRYAN RUDY GONZALES

HAZEL SANTIAGO

LAUREN JAMES

Planning and scheduling (T-1):

Work Breakdown Structure

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Note
Jennifer Vu **(coordinator)**	jvu@student.gsu.edu	-Make work breakdown structure -Proofread and print out copy of report to hand in to the instructor -Raspberry PI Installation & ARM Assembly Programming Participation	2	none	-9/18/2019 -9/20/2019 -9/17/2019	
Abraham Mammen	amammen1@student.gsu.edu	-Report -Raspberry PI Installation & ARM Assembly Programming Participation	1	Teamwork and lab reports needed before putting together final report.	-9/19/2019 -9/17/2019	
Rachid Bodson	rbodson1@student.gsu.edu	-Typing up the answers to teamwork basics questions -Raspberry PI Installation & ARM Assembly Programming Participation	2	none	-9/15/2019 -9/17/2019	Make a rough draft of answers and ask team members for input on what needs to be changed or added.
Byran Gonzales	bgonzales1@student.gsu.edu	-Video -Raspberry PI Installation & ARM Assembly Programming Participation	5	Team members should prepare what they are going to say	-9/19/2019 -9/17/2019	

				beforeh and.		
Hazel Santiago	hsantiago1@student.gsu.edu	-Creating the Slack and Github -Raspberry PI Installation & ARM Assembly Programming Participation	1	Slack, Github	-9/18/2019	Send everyone the Slack and Github links. Ask everyone to send a message over Slack with their introductio n.
Lauren James	ljames26@student.gsu.edu	-Raspberry PI Installation & ARM Assembly Programming Participation - Raspberry PI Lab Report	6	none	-9/17/2019 -9/18/2019	Schedule and reserve meeting rooms in the library

Teamwork basics(T-3):

1. We will scan through each task and assign each team member a task based on their level of understanding of the material demanded in that task. We will be having a meeting at least twice a week for an estimation of two hours each meeting to discuss the progress of each team member's assigned task. Each team member will present their assigned task and walk us through the process of what they have accomplished so far. We will then together give a feedback and introduce new ideas to better the task. We will make sure that each team member is highly satisfied with their task and the task of the other team members.

2. ***Work Norms:***

We will distribute the work based on each member's understanding of the task's demands. The deadline will be set by the team coordinator. Twenty-four hours before the deadline of each task, the team coordinator will notify each person about the deadline and request a screenshot of progress of the task to ensure that the deadline will be met with the completion of the task. However, if the deadline is missed and the due date for the assignment is near, then a meeting will be held to finish up the task. The team member will be put on probation, however, if that happen again, then points may be deducted from their participation grade. We will be holding at least two meeting every week to review each team member's task. Everyone will be given a chance to give a feedback of each team member's task and the overall tasks combined. All opinions will be put together and generate a final say, but if there is still a confusion then a majority vote will be the final say. Each assigned task will be given enough time to be completed. This will help avoid the team member from rushing to complete the task and avoid mistakes. Everybody will be advised to think about the task before diving into it and to avoid the pressure of a deadline.

Facilitator Norms:

Yes, a facilitator will be used. The facilitator will be chosen based on the understanding of the project. It will be rotated at each project. The facilitator will be responsible for focusing the team on a task, getting participation from all team members, keeping the team on time, helping team members confront problems, helping clarify team's decisions.

Communication Norms:

Communication will take place at every time to ensure that everyone is up to date on the project. Communication will be through e-mail, a text and a phone call.

3. As a team, we will have a hard time completing an assignment if we have a team member who argues or complains. Every case of argument or complaint will be taken seriously and we will find a way to solve it. If the same person continues to argue and complain over everything, and we find the complaint to be irrelevant, then points may be deducted from their participation grade to avoid future arguments and complaints.

4. If the team is having a trouble reaching a consensus, then the majority vote rule will be put into effect.

5. There will not be any pressure put on anybody. If a team member reaches a decision quicker than the rest, then that team member will have to help the others to reach their decisions.

6. On each assignment, an "A" will be the targeted grade, however, if anybody decides to get a "B", then we will walk that person through the benefits of having an "A". We will try convincing that person to think of "B" as a mediocre accomplishment, but "A" is what the team is aiming for.

Raspberry PI Installation and ARM Assembly Programming(T-4):

Part 1:

First, we ensured that **scrot** was installed so we could take screenshots of our progression using the Raspberry Pi for our reports.

```
pi@raspberrypi:~ $ nano first.s
pi@raspberrypi:~ $ sudo apt-get install scrot
Reading package lists... Done
Building dependency tree       ano first.s
Reading state information... Done
scrot is already the newest version (0.8-18).
scrot set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

There was no output after attempting to run the program using **./first** command because there was an error. While debugging, we discovered this error was caused due to the absence of breakpoints.

```
pi@raspberrypi:~ $ as-o first.o first.s
bash: as-o: command not found
pi@raspberrypi:~ $ as -o first.o first.s
pi@raspberrypi:~ $ ld -o first first.o
pi@raspberrypi:~ $ scrot
```

gdb first opened the file for us to debug and displayed the copyright and licensing information.

```
pi@raspberrypi:~ $ gdb first
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 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) list
```

(gdb) list displayed the first 10 lines of the program.

```
(gdb) list
1       @first program
2         .section .data
3         .section .text
4         .globl _start
5         _start:
6           mov r1,#5  @load r1 with 5
7           sub r1,r1,#1  @subtract 1 from r1
8           add r1,r1,#4 @add 4 to 1
9
10          mov r7,#1 @Program Termination:exit syscall
```

(gdb) b 11 told the program to stop on that line at the breakpoint.

```
(gdb) b 11
Breakpoint 1 at 0x10064: file first.s, line 11.
```

(gdb) run started the program before stopping at the assigned breakpoint.

```
(gdb) run
Starting program: /home/pi/first

Breakpoint 1, _start () at first.s:11
11          svc #0 @Program Termination:wake kernel
```

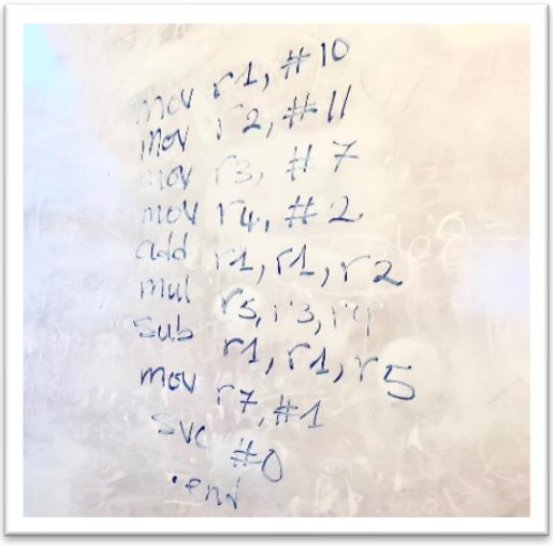
(gdb) info registers displayed the register's memory locations and the values stored in them. r1 had a value of 8, which shows that our program worked as 5-1+4=8.

```
(gdb) info registers
r0             0x0      0
r1             0x8      8
r2             0x0      0
r3             0x0      0
r4             0x0      0
r5             0x0      0
r6             0x0      0
r7             0x1      1
r8             0x0      0
r9             0x0      0
r10            0x0      0
r11            0x0      0
r12            0x0      0
sp             0x7efff060       0x7efff060
lr             0x0      0
pc             0x10064  0x10064 <_start+16>
cpsr           0x10     16
```

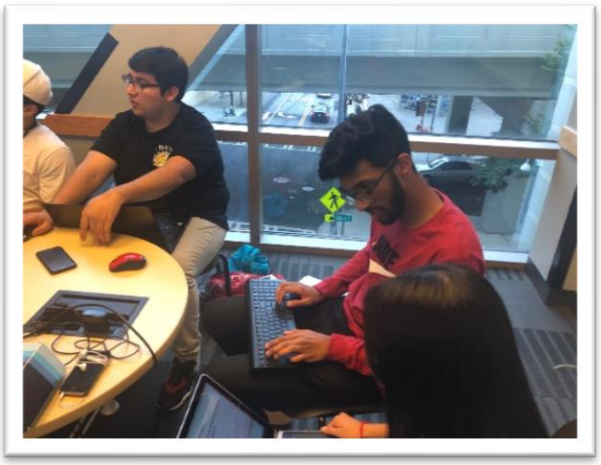
Part 2:

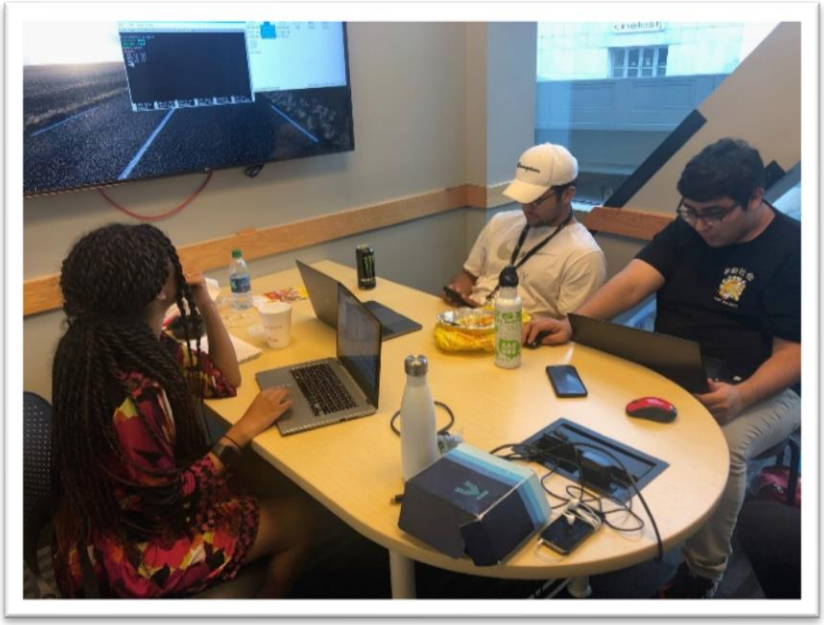
We created a program called arithmetic1.s using the same instructions as Part 1 to guide. We whiteboarded our code before inserting it in the program to see how we planned on executing the calculations.

A = (A + B) − (C *D), where A=10, B=11, C=7, and D=2



In our program, r1 loaded the value 10, r2 loaded the value 11, r3 loaded the value 7, and r4 loaded the value 2.





We assembled and linked the file to have an executable program before debugging using GNU creating a breakpoint.

```
 GNU nano 2.7.4                File: arithmetic1.s

@ arithmetic1 program
.section .data
.section .text
.globl _start
_start:
  mov r1, #10  @load r1 with 10
  mov r2, #11  @load r2 with 11
  mov r3, #7   @load r3 with 7
  mov r4, #2   @load r4 with 2

  add r1,r1,r2  @add r1 and r2
  mul r5,r3,r4  @multiply r3 and r4
  sub r1,r1,r5  @ subtract r5 from r1
  mov r7,#1     @ Program Termination: exit syscall
  svc #0        @Program Termination: wake kernel
 .end
```

```
bash: gbd: command not found
pi@raspberrypi:~ $ nano arithmetic1.s
pi@raspberrypi:~ $ as -o arithmetic1.o arithmetic1.s
pi@raspberrypi:~ $ ld -o arithmetic1 arithmetic1.o
pi@raspberrypi:~ $ ./arithmetic1
pi@raspberrypi:~ $ as -g -o arithmetic1.o arithmetic1.s
pi@raspberrypi:~ $ ld -o arithmetic1 arithmetic1.o
pi@raspberrypi:~ $ gbd arithmetic1
bash: gbd: command not found
pi@raspberrypi:~ $ gdb arithmetic1
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 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"...
```

We had created a breakpoint using (gdb) b 14 command, so the program will stop executing before line 14 (it is no longer 11 as this program is longer than first.s). Since (10+11)-(7*2)=(21)-(14)=7, r1 should have 7 stored after running the program. We entered the (gdb) info registers command, to prove that our program had the correct output and made the right computations.

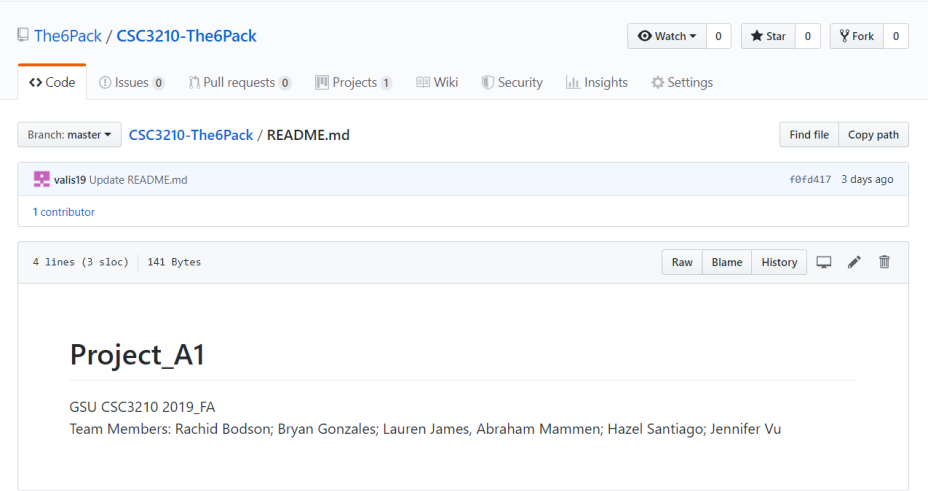
```
(gdb) run
Starting program: /home/pi/arithmetic1

Breakpoint 1, _start () at arithmetic1.s:14
14          mov r7,#1       @ Program Termination: exit syscall
(gdb) info registers
r0             0x0      0
r1             0x7      7
r2             0xb      11
r3             0x7      7
r4             0x2      2
r5             0xe      14
r6             0x0      0
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0x0      0
r11            0x0      0
r12            0x0      0
sp             0x7efff050       0x7efff050
lr             0x0      0
pc             0x10070  0x10070 <_start+28>
cpsr           0x10     16
(gdb)
```

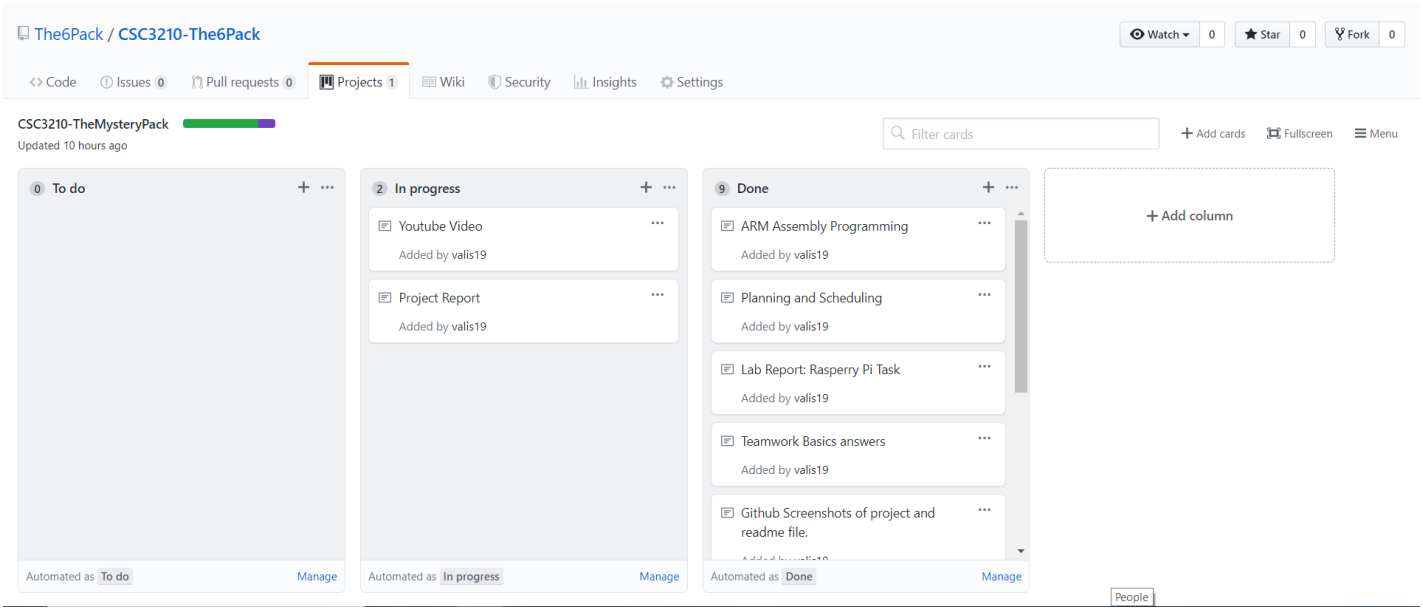
Appendix:

GitHub- https://github.com/The6Pack

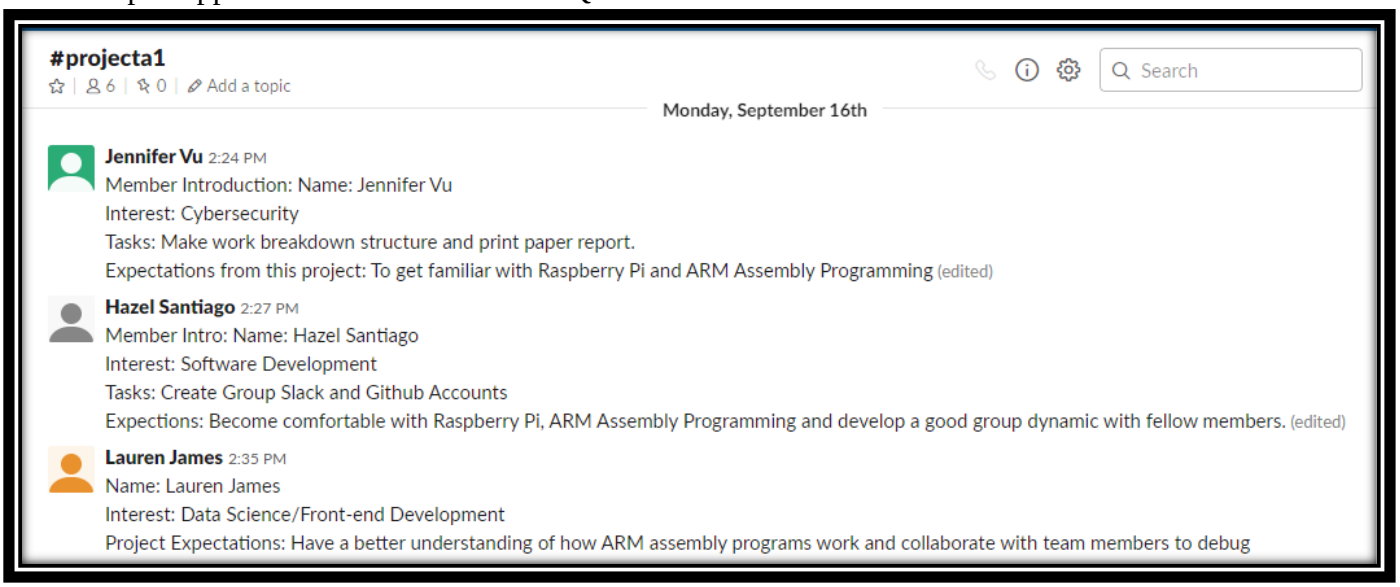
ReadMe Page:

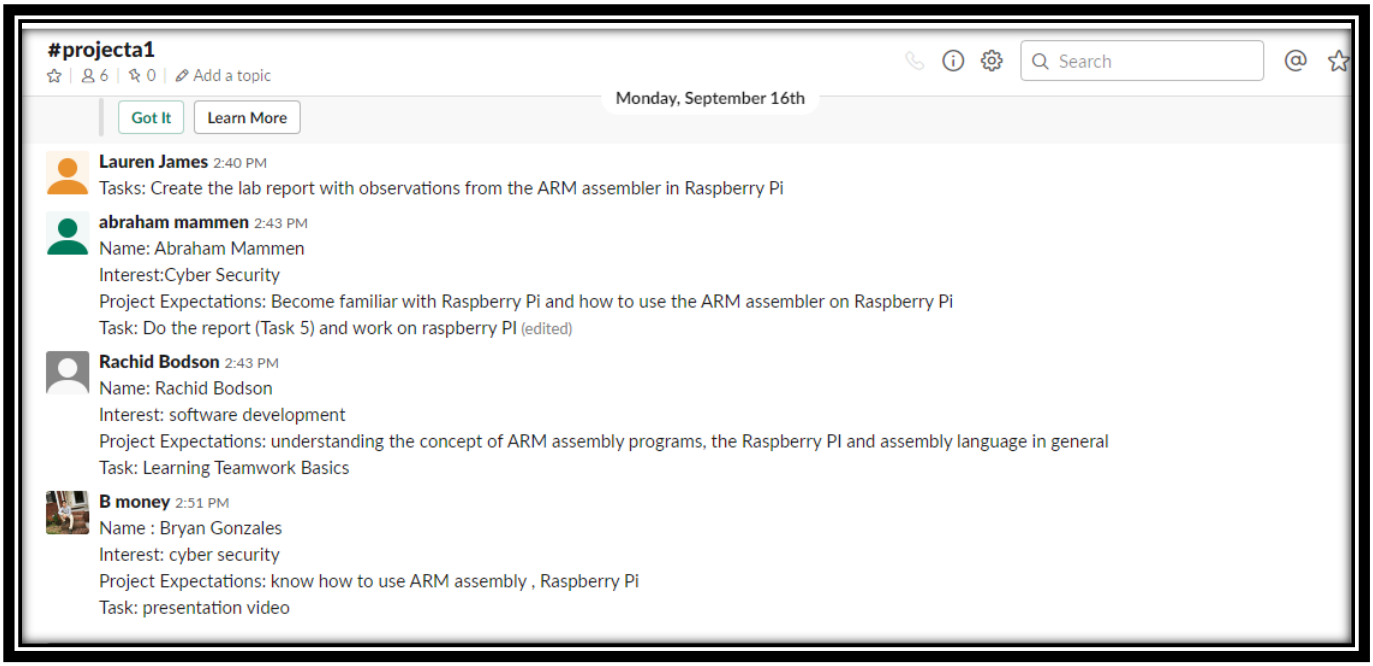


Project Page:



Slack- https://app.slack.com/client/TN9D41Q1K/CN8UPBFB6





Youtube-https://www.youtube.com/channel/UC7dMmLnGrmjZv3d8c-39qJg?view_as=subscriber