Developing Soft and Parallel Programming Skills Using Project Based Learning

Fall 2019
Area 42
Members: Reza Ebrahimi, Ettione, Md Sultanul Arefin Khan, Glenn Tolbert, Billy Lewis

Planning And Scheduling: (Task 1)

Assignee Name	Email	Task	Duration (Hours)	Dependency	Due Date	Note
Reza Ebrahimi	rebrahimi1@student.gsu.edu	Raspberry Pi installation and using ARM assembly language to complete Task 4 Slack Channel Creation and Invitation	6 hours	Raspberry PI - Monitor - High Speed Internet - Monitor	09/19/19	Go over the detailed lab report thoroughly with other members and fix any mistakes
Ettione	estuckeyii1@student.gsu.edu	Utilizing all the group members to answer all the questions regarding teamwork basics for Task 3.	3 hours	Teanwork basics document on icollege.	09/17/19	Consult each member and note down opinions in order to answer teamwork basic questions as required
Md Sultanul Arefin Khan	mkhan55@student.gsu.edu	Planning and scheduling, Creating and managing Github workspace, Final Report, Video and Youtube Channel	5 hours	Github, Google docs, Gmail, Youtube and the video	09/18/19	Must be ready 36 hours before due date and then send the report to the coordinator

Glenn Tolbert	gtolbert4@student.gsu.edu	Raspberry Pi installation and using ARM assembly language to complete Task 4	6 hours	Raspberry PI - Monitor - High Speed Internet - Monitor	09/19/19	Go over the detailed lab report thoroughly with other members and fix any mistakes
Billy Lewis	tngamdee1@student.gsu.edu	Raspberry Pi installation and using ARM assembly language to complete Task 4	6 hours	Raspberry PI - Monitor - High Speed Internet - Monitor	09/19/19	Go over the detailed lab report thoroughly with other members and fix any mistakes

Teamwork Basics: (Task 3)

1. **What to do to get the task accomplished *and* the team members' satisfaction high?**

   It is important to allocate work responsibly to ensure that every group member has ample time to complete their tasks and ask for help if needed; this way, all work is completed within the appropriate time frame. Furthermore, listening to the questions, ideas, and concerns of your group members is crucial when it comes to keeping team satisfaction high.

2. **Answer all the questions in the Work Norms, Facilitator Norms, Communication Norms using your own words and your own context.**

   A) **<u>Work Norms</u>**:

-Work will be distributed based on the strengths of the group members, the amount of work needed to be done per task, and the number of members assigned per task.

-Not one single person will decide deadlines, we will communicate to each other how much time we may need per task and decide on a deadline based on that and the due date of the overall assignment.

- If someone does not follow through on a deadline, the remaining group members will either work together to finish the slackers' task or assign someone to finish that

work (if the task is simple enough). We will talk directly with that person to tell them the effect their actions are having on the group.

-The work will be uploaded to a google account (managed by the facilitator) shared by all group members and be reviewed by ALL members together before the due date. This will be done a considerable amount of time before the due date so changes can be made if need be.

-If group members have different opinions on the quality of the work, the remaining group members will weigh in on whether the task needs to be changed. If the other members feel as though the assignment is fine and an agreement can't be reached, then a member of the group will contact the professor for clarity on the assignment.

-Members having different work habits will not be an issue so long as the assignment gets done with enough time to be reviewed before the due date. If a member of the group cannot accomplish this then the remaining members will contact the professor.

B) <u>Facilitator Norms:</u>

-We will use a Facilitator.

-The position will be given to a group member who volunteers for the position. If no one volunteers, the Facilitator will be randomly chosen.

-The position of Facilitator will be rotated so ensure that everyone gets to experience the position.

-The responsibility of the Facilitator is to essentially keep the team on track. They will check on other members to ensure that we are keeping up with our tasks and responsibilities. The Facilitator will also help settle group issues.

C) **<u>Communications Norms:</u>**

-Communications will take place all throughout the project whenever needed. We will be using Google, Slack, GitHub, and GroupMe.

3. **As a team, select two cases out of the four mentioned in Handling Difficult Behavior (use your own words and your own context)**

   A) **<u>Overly Talkative:</u>** If a group member is overly talkative, we will assess what they're saying and determine if they are contributing to or causing harm to the group. If the case is the latter, we will shift the conversation in the direction of other members. If the case is the former, we will incorporate their ideas into the project as need be. In either case, the Facilitator will let the talkative member know that they are talking a bit too much.

   B) **<u>Complains:</u>** If a member is prone to complaining, we will listen to their complaints and determine if their complaints are well placed. If so, we will discuss the issues as a group in a meeting. Otherwise, the Facilitator will inform the member that a large part of this project is problem solving and urge the member to work with everyone to solve the issues at hand instead of complaining.

4. **When making decisions, if the team is having trouble reaching a consensus, what should you do? (use your own words and your own context)**

   If the team is having issues making decisions, we will identify all the different ideas and vote on the most popular ones. We will then figure out the similarities and differences in them in order to come up with an idea that everyone can appreciate.

5. **What should you do if person reach a decision more quickly than others and pressure people to move on before it is a good idea to do so?**

   The Facilitator will as everyone if they are all ok with the decision presented by the eager member to prevent the decision from being carried out too quickly.

6. **What happens if most people on the team want to get an "A" on the assignment, but another person decides a "B" will be acceptable?**

   Conversations on this matter *must* be had in advance as it is difficult to manage this issue without doing so. Group members will be totally transparent about their goals in the course to prevent these issues from arising.

Raspberry PI Installation and Arm Assembly Programming- Lab Report (Task 4):

Observations with Installing ARM Assembly Programming on Raspberry Pi:

To access the program we created, we need to first type “nano” to be able to Read/Write in the file.

To assemble the code in:

line 1 we use the instruction code, mov, to move the immediate, 5, to the general purpose register, r1.

In line 2, the instruction code, sub, is used to subtract the immediate, 1, from general purpose register, r1, and the result is stored in the general purpose register, r1.
Line 3, the instruction code, add, is used to add the immediate, 4, to the general purpose register, r1, and the result is stored in the general purpose register, r1.
Line 4 we use the instruction code, mov, to move the immediate, 1, to the general purpose register, r7.

To make comments, we use the “@” sign in order for other team members to know what each line of code does.

To save the file, we need to use “Ctrl W” and “Ctrl X” to exit the editor.

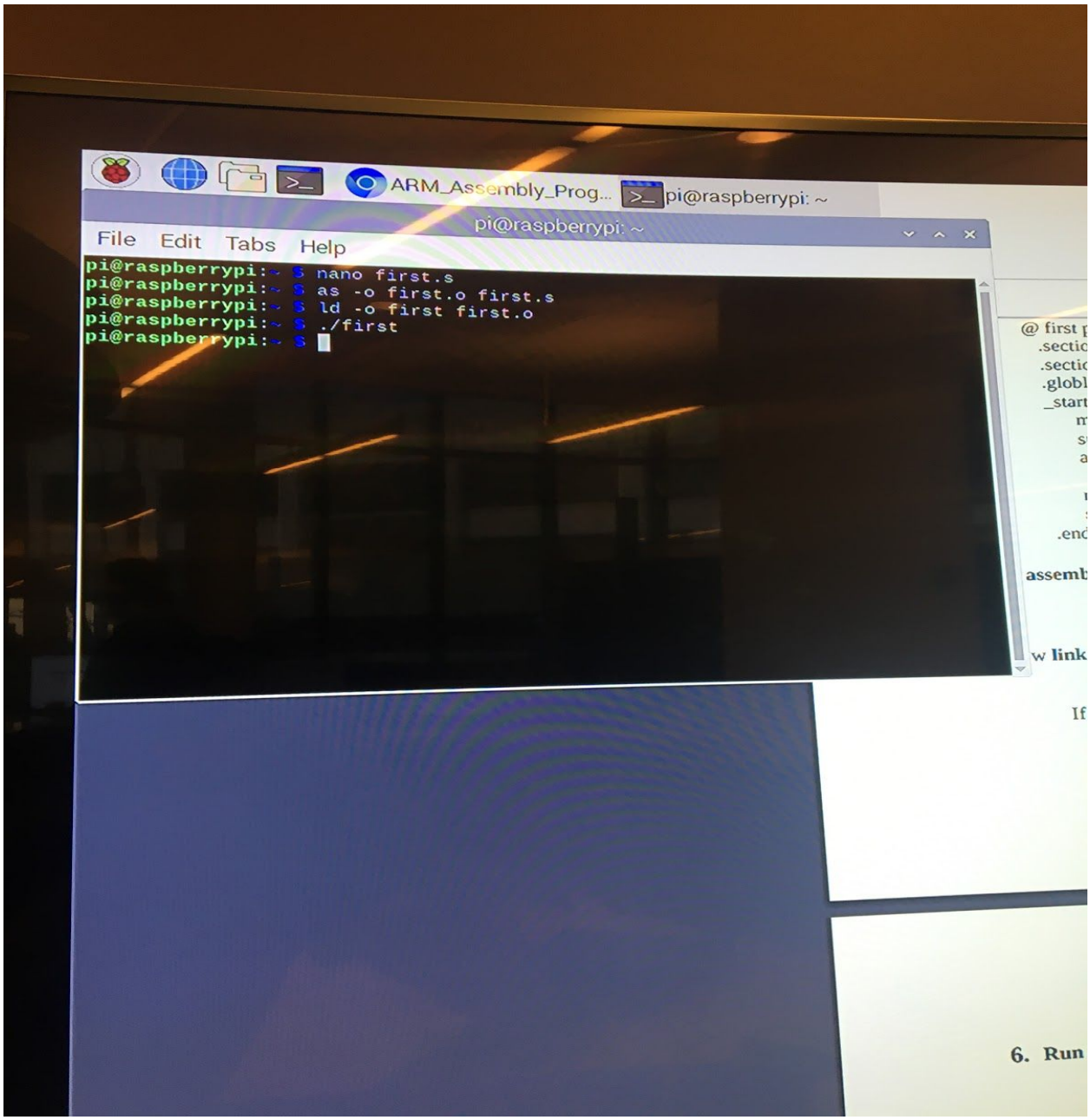
At first when we tried to run the code we did not see an output because in our program we did not have a line of code to print the contents of the general purpose register to the monitor. There is no output because we did not use the GDB debugger to step through the program and examine the contents of the registers and memory.

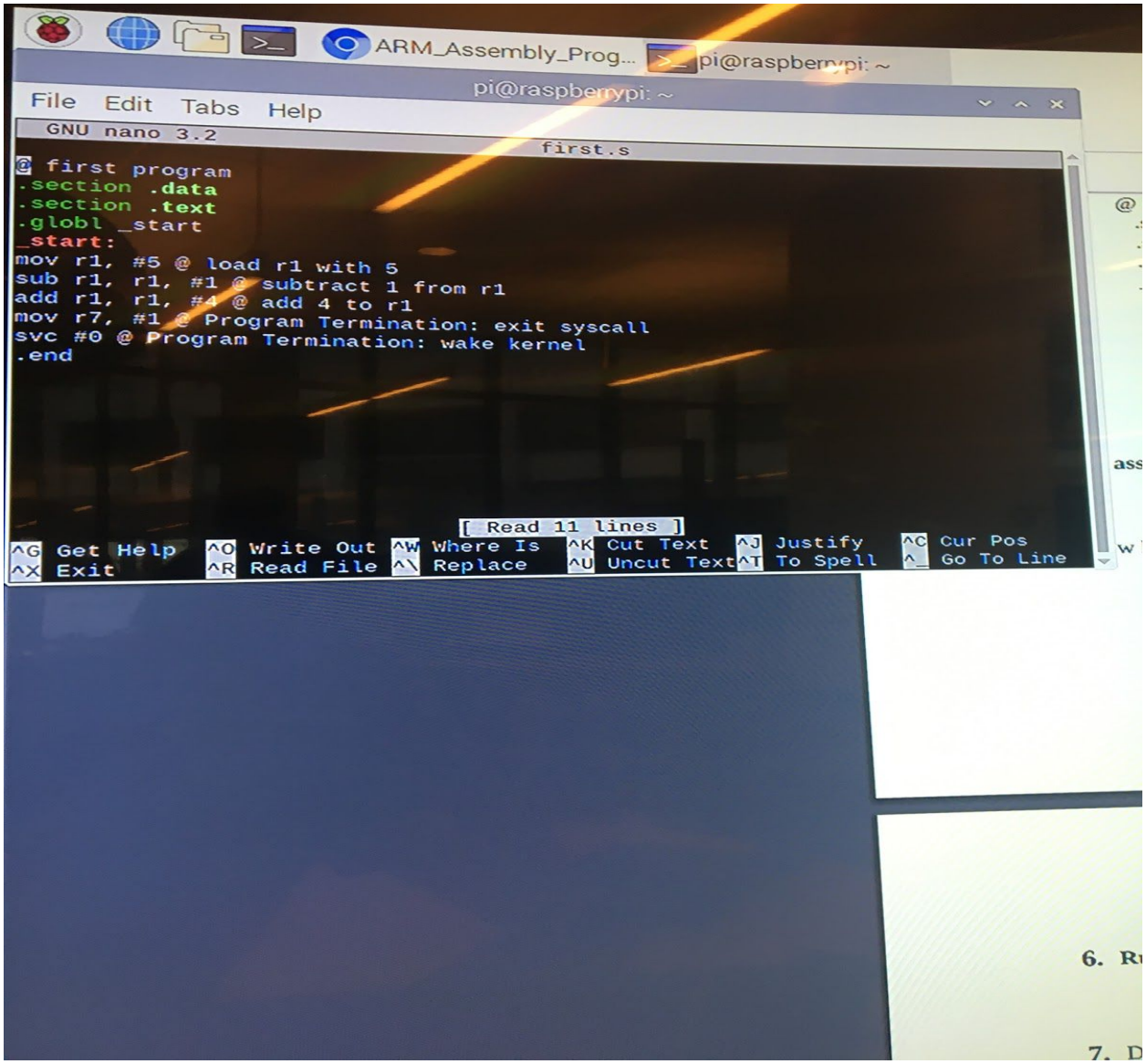
When debugging the code we used the GDB debugger to set a breakpoint at the end of the code and we looked at the contents of the registers. We saw that the general purpose register, r1, held the value 8 and the general purpose register, r7, held the value 1. The first row held the information about what register we were looking at and the second row held the values of the hexadecimal number while the third row held the values of the decimal number in each register.

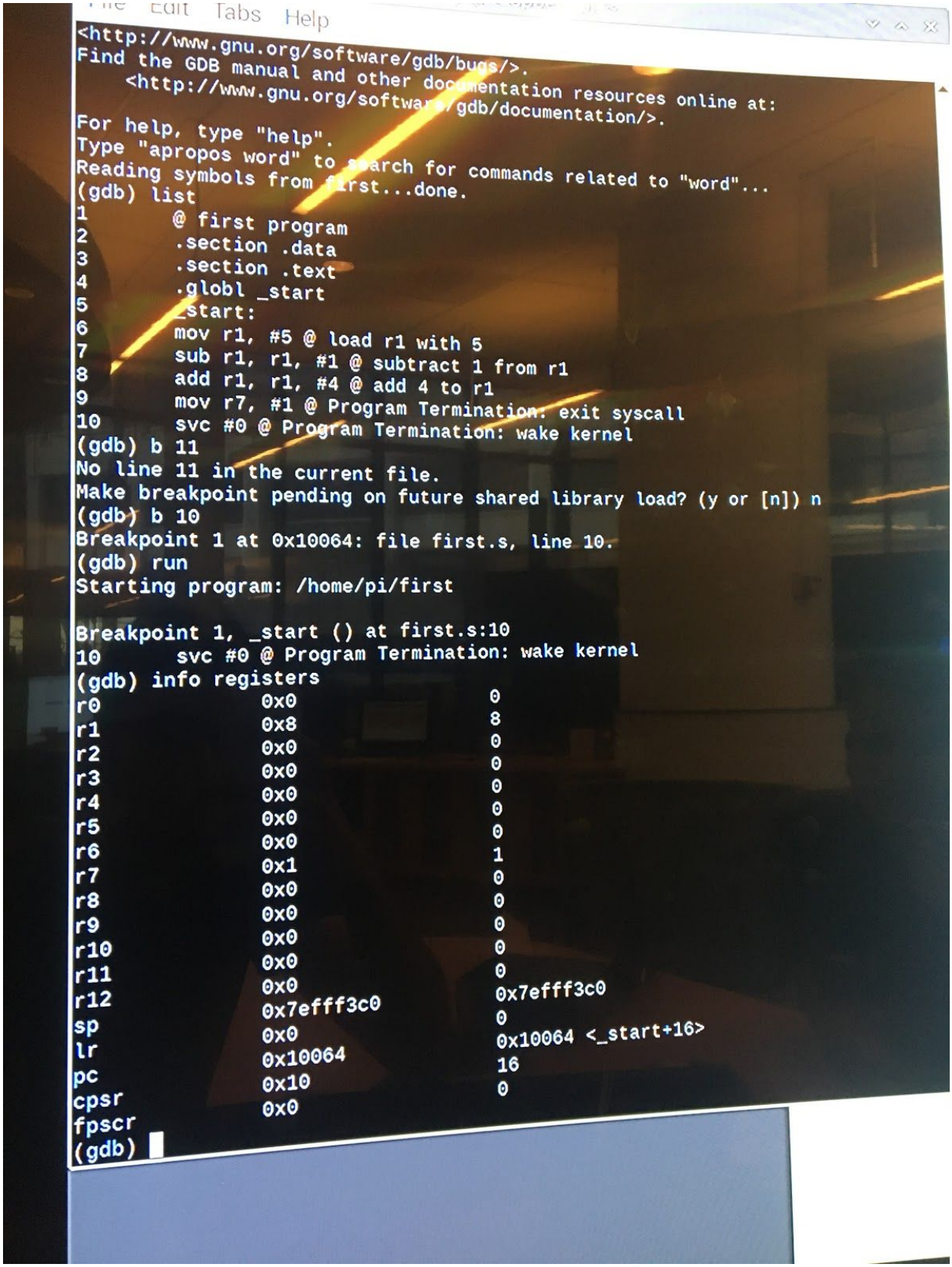
Part 2:
In our code we used the instruction code, mov, to move the immediates, 10; 11; 7; 2, into the general purpose registers, r1; r2; r3; r4, respectively. We manipulated the values in the registers by adding the contents of general purpose register, r2, to general purpose register, r1 using the instruction code, add; multiplying the contents of general purpose register, r4, to general purpose register, r3 using the instruction code, mul; and then subtracted general purpose register, r3, from general purpose register, r1 using the instruction code, sub. Then we moved the immediate, 1, to the general purpose register, r7, using the instruction code, mov.

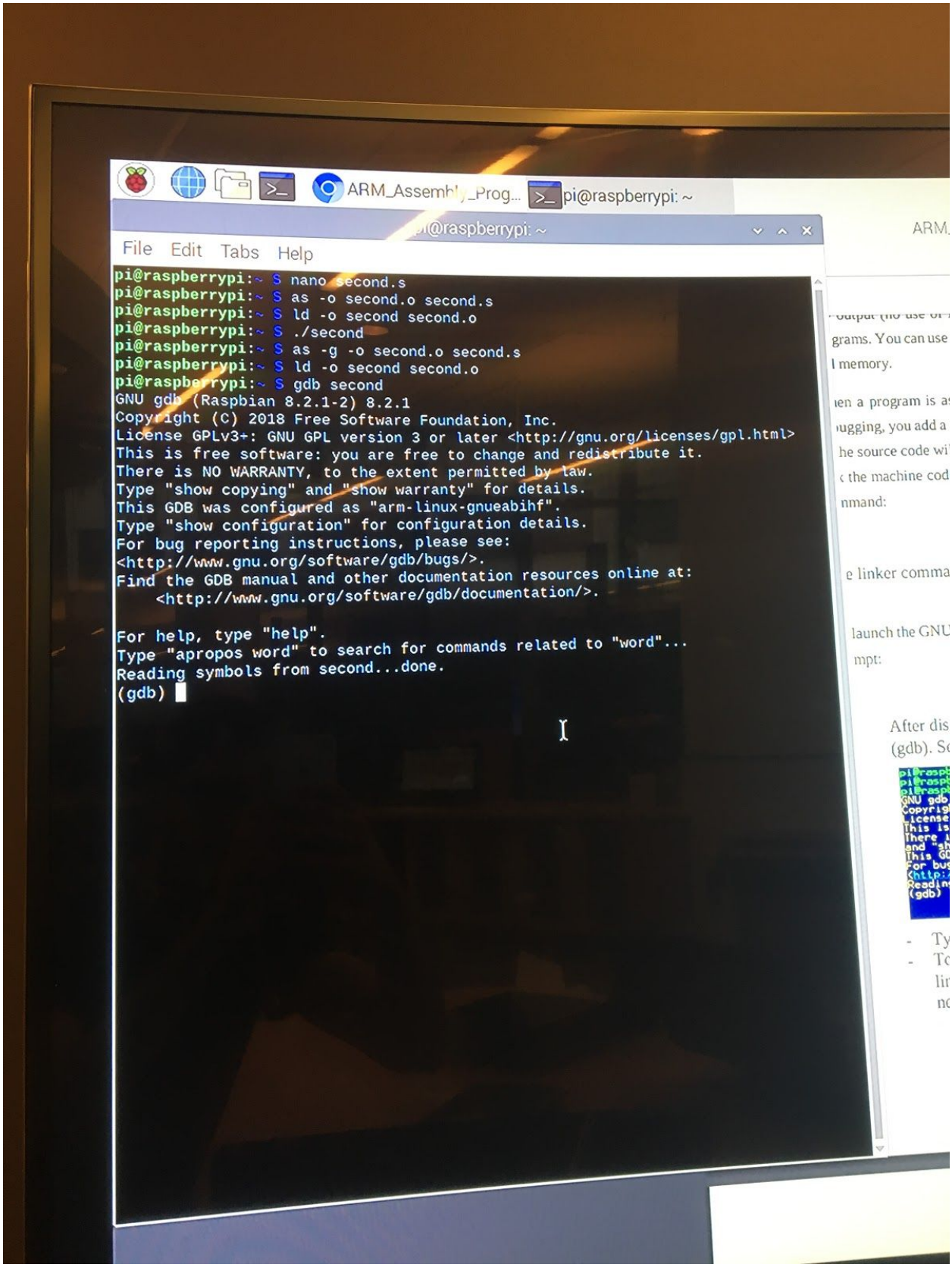
We debugged the program and saw that general purpose register, r1, held the value 7; general purpose register, r2, held the value 11; general purpose register, r3, held the value 14; general purpose register, r4 held the value 2; and general purpose register, r7, held the value 1.

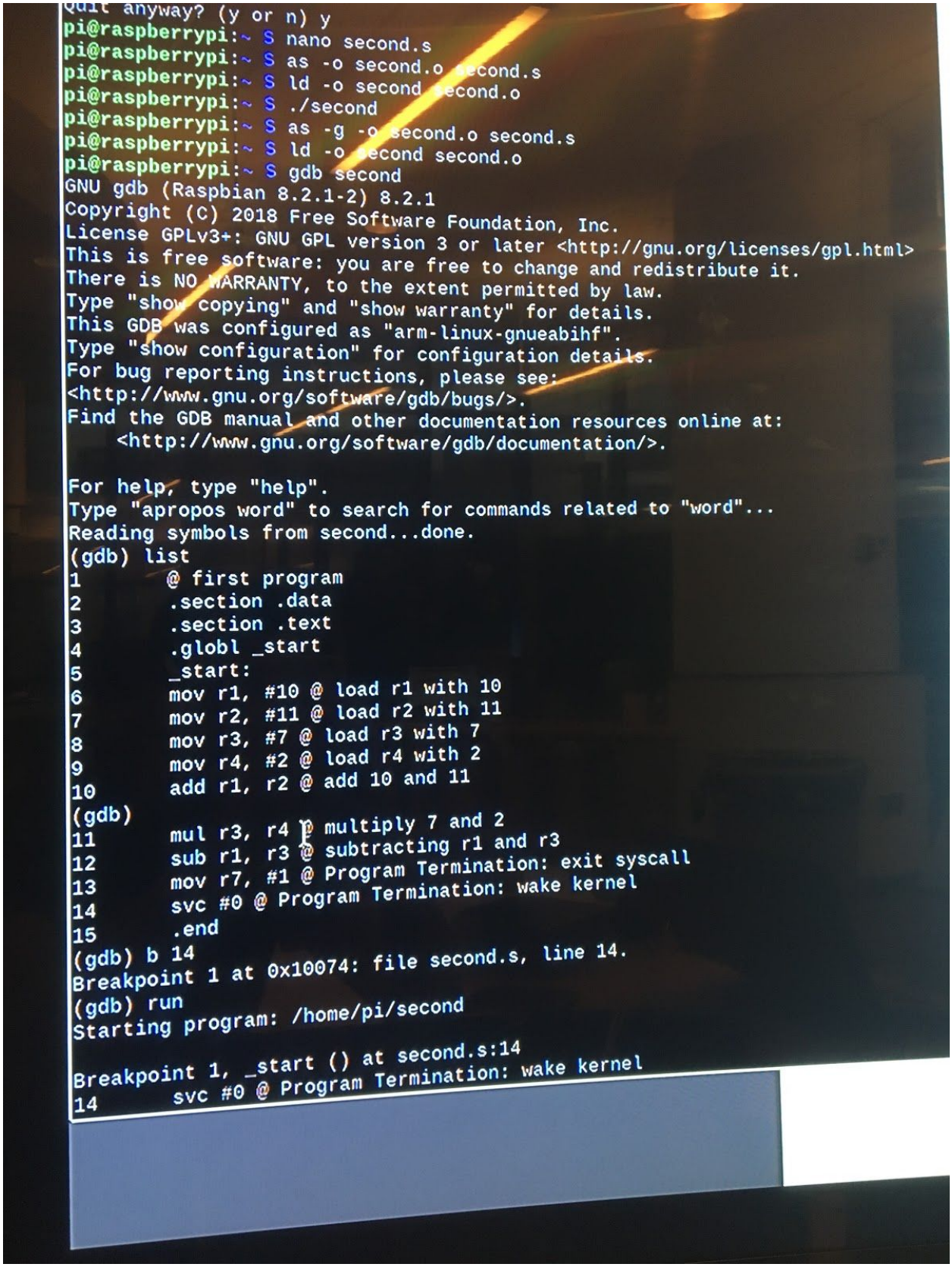
Next few pages contains screenshots of the actual program and answer.

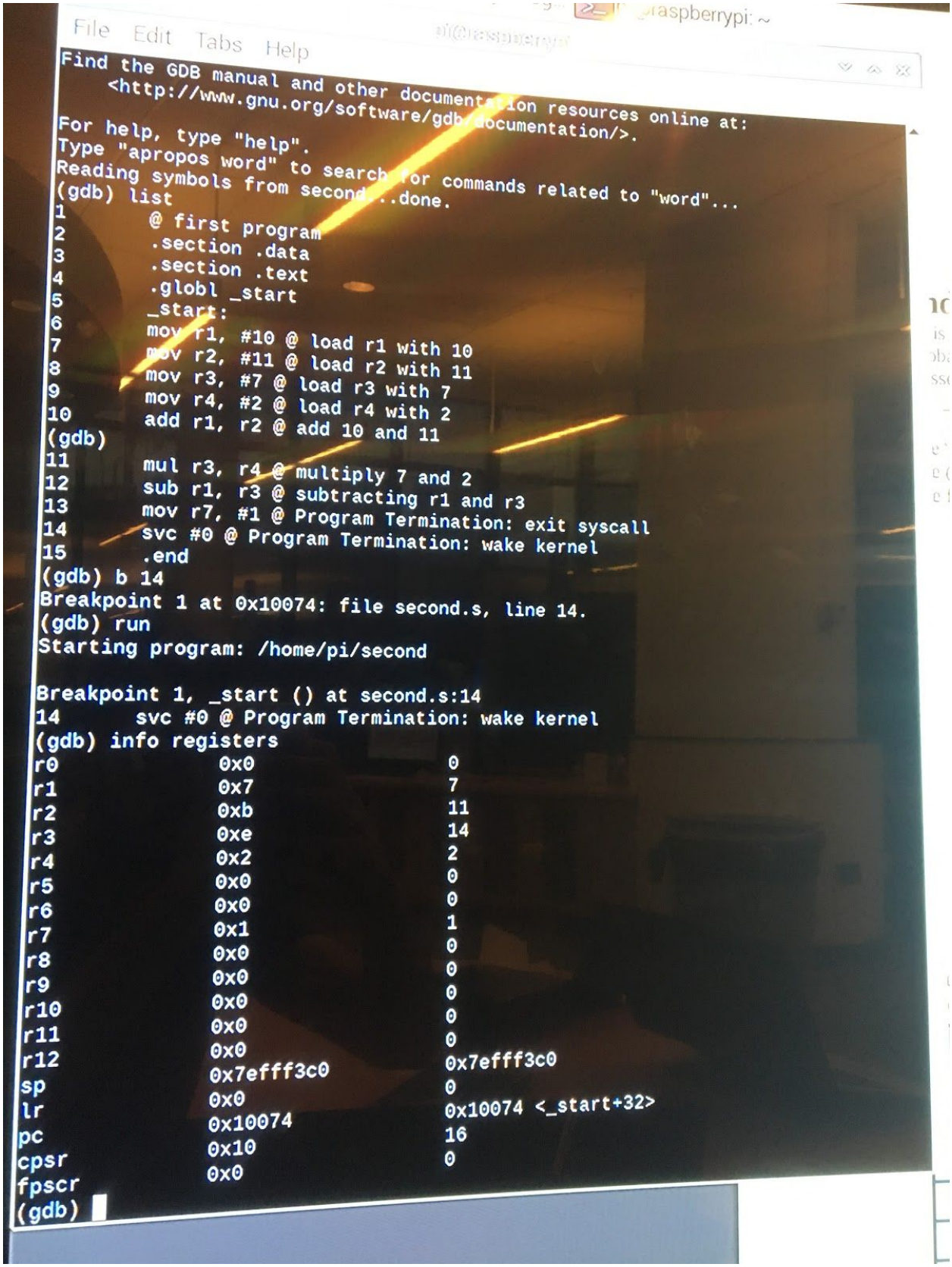

ARM_Assembly_Prog...
pi@raspberrypi: ~
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ nano first.s
pi@raspberrypi:~ $ as -o first.o first.s
pi@raspberrypi:~ $ ld -o first first.o
pi@raspberrypi:~ $ ./first
pi@raspberrypi:~ $
6. Run


ARM_Assembly_Prog...
pi@raspberrypi: ~
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2
first.s
@ 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
[Read 11 lines]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line


File Edit Tabs Help
<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
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 r1
9 mov r7, #1 @ Program Termination: exit syscall
10 svc #0 @ Program Termination: wake kernel
(gdb) b 11
No line 11 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) b 10
Breakpoint 1 at 0x10064: file first.s, line 10.
(gdb) run
Starting program: /home/pi/first
Breakpoint 1, _start () at first.s:10
10 svc #0 @ Program Termination: wake kernel
(gdb) info registers
r0 0x0 0
r1 0x8 8
r2 0x0 0
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 1
r7 0x1 0
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0x7efff3c0
sp 0x7efff3c0 0
lr 0x0 0x10064 <_start+16>
pc 0x10064 16
cpsr 0x10 0
fpscr 0x0
(gdb)


ARM_Assembly_Prog...
pi@raspberrypi: ~
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ nano second.s
pi@raspberrypi:~ $ as -o second.o second.s
pi@raspberrypi:~ $ ld -o second second.o
pi@raspberrypi:~ $ ./second
pi@raspberrypi:~ $ as -g -o second.o second.s
pi@raspberrypi:~ $ ld -o second second.o
pi@raspberrypi:~ $ gdb second
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 second...done.
(gdb)

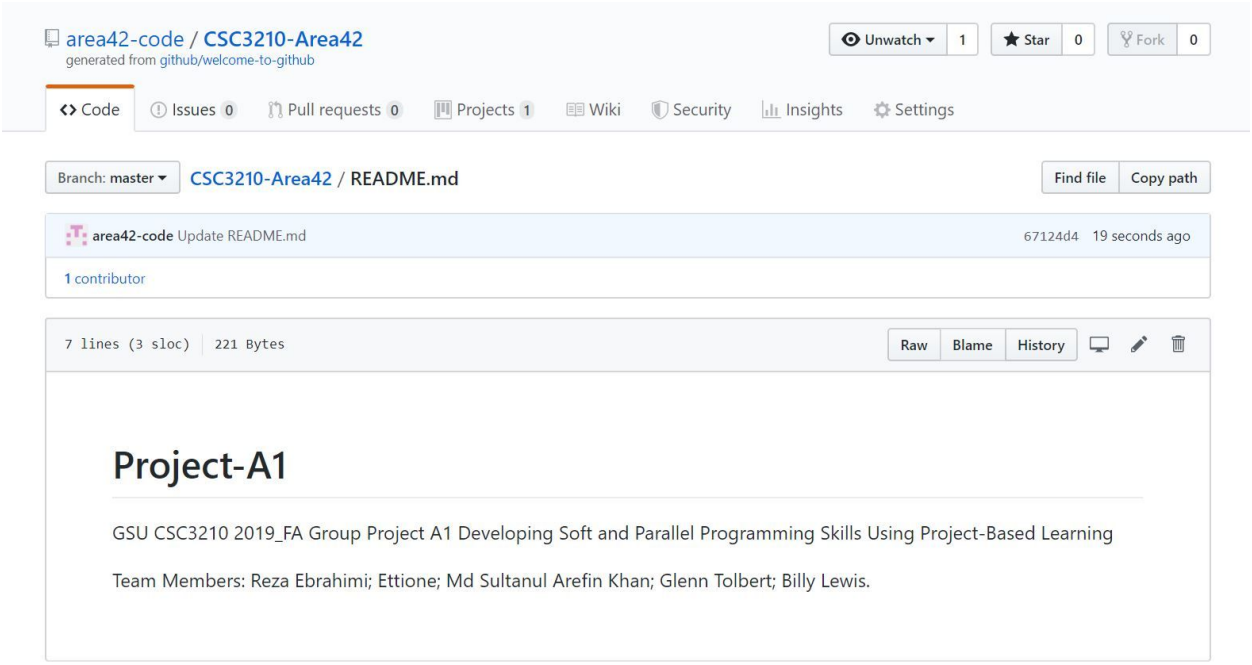

Quit anyway? (y or n) y
pi@raspberrypi:~ $ nano second.s
pi@raspberrypi:~ $ as -o second.o second.s
pi@raspberrypi:~ $ ld -o second second.o
pi@raspberrypi:~ $ ./second
pi@raspberrypi:~ $ as -g -o second.o second.s
pi@raspberrypi:~ $ ld -o second second.o
pi@raspberrypi:~ $ gdb second
GNU gdb (Raspbian 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 second...done.
(gdb) list
1 @ first program
2 .section .data
3 .section .text
4 .globl _start
5 _start:
6 mov r1, #10 @ load r1 with 10
7 mov r2, #11 @ load r2 with 11
8 mov r3, #7 @ load r3 with 7
9 mov r4, #2 @ load r4 with 2
10 add r1, r2 @ add 10 and 11
(gdb)
11 mul r3, r4 @ multiply 7 and 2
12 sub r1, r3 @ subtracting r1 and r3
13 mov r7, #1 @ Program Termination: exit syscall
14 svc #0 @ Program Termination: wake kernel
15 .end
(gdb) b 14
Breakpoint 1 at 0x10074: file second.s, line 14.
(gdb) run
Starting program: /home/pi/second
Breakpoint 1, _start () at second.s:14
14 svc #0 @ Program Termination: wake kernel

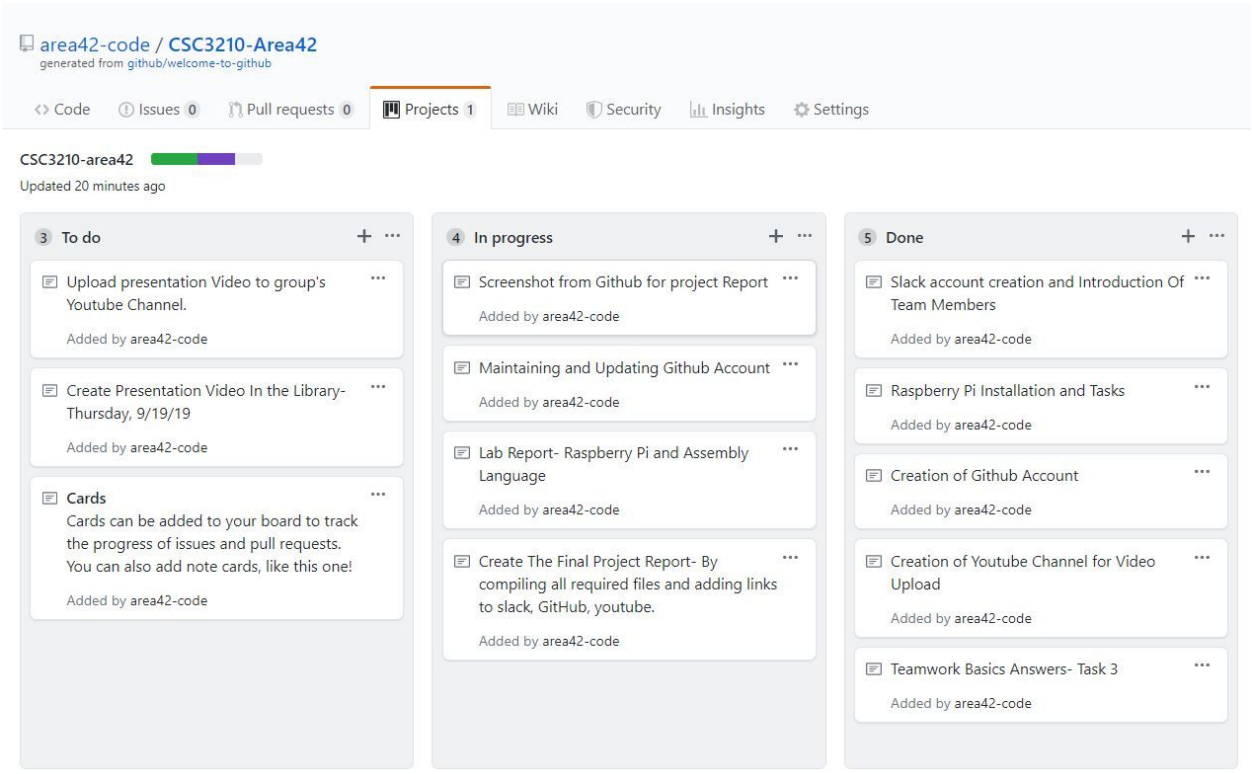

File Edit Tabs Help
raspberrypi: ~
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from second...done.
(gdb) list
1 @ first program
2 .section .data
3 .section .text
4 .globl _start
5 _start:
6 mov r1, #10 @ load r1 with 10
7 mov r2, #11 @ load r2 with 11
8 mov r3, #7 @ load r3 with 7
9 mov r4, #2 @ load r4 with 2
10 add r1, r2 @ add 10 and 11
(gdb)
11 mul r3, r4 @ multiply 7 and 2
12 sub r1, r3 @ subtracting r1 and r3
13 mov r7, #1 @ Program Termination: exit syscall
14 svc #0 @ Program Termination: wake kernel
15 .end
(gdb) b 14
Breakpoint 1 at 0x10074: file second.s, line 14.
(gdb) run
Starting program: /home/pi/second
Breakpoint 1, _start () at second.s:14
14 svc #0 @ Program Termination: wake kernel
(gdb) info registers
r0 0x0 0
r1 0x7 7
r2 0xb 11
r3 0xe 14
r4 0x2 2
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 0x7efff3c0 0x7efff3c0
lr 0x0 0
pc 0x10074 0x10074 <_start+32>
cpsr 0x10 16
fpscr 0x0 0
(gdb)

Appendix:

Github: https://github.com/area42-code

Required Screenshots:

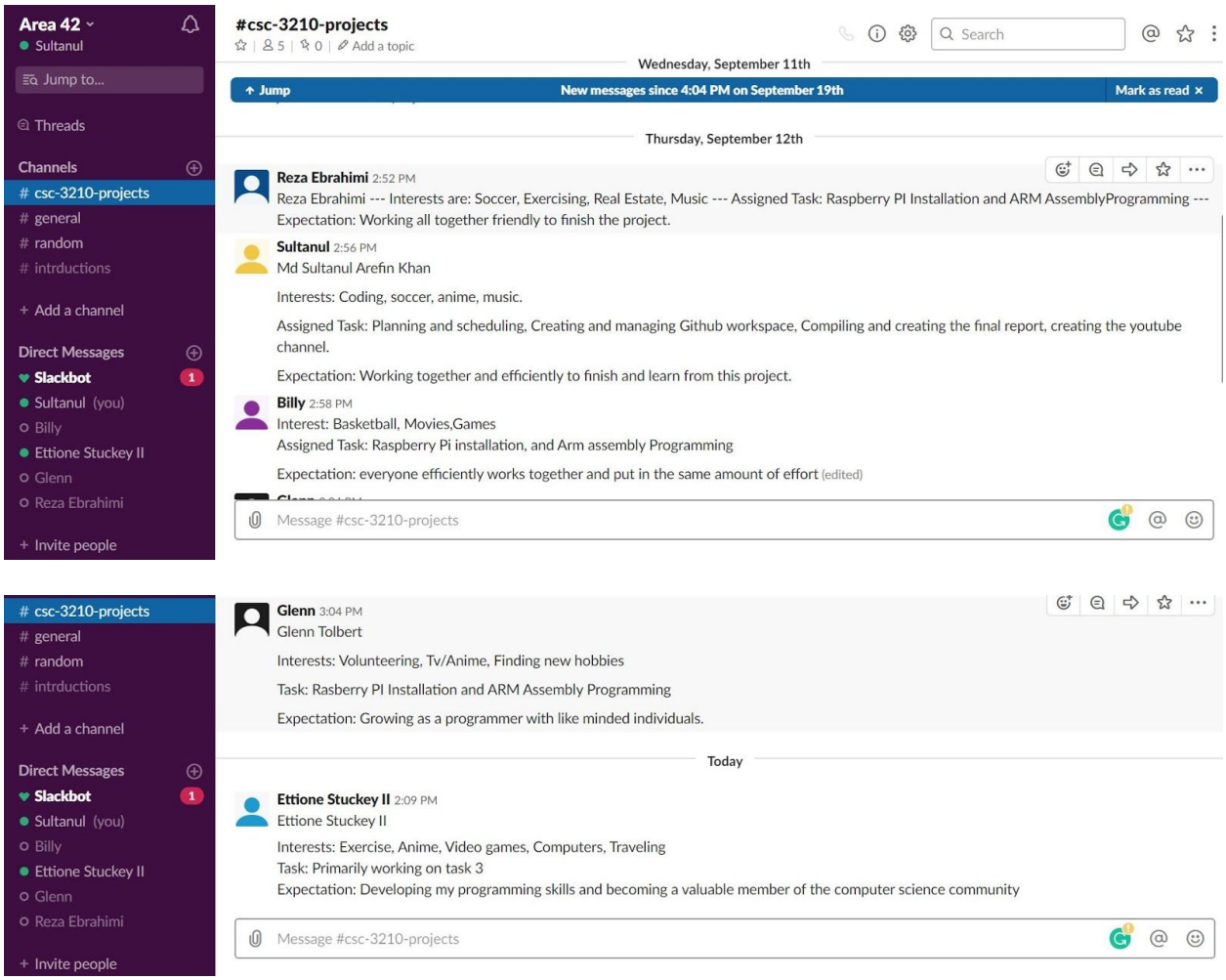




Slack:

Link to channel: https://app.slack.com/client/TN68KEVL7/CN4CHL9UG

Screenshot of main screen:



YouTube Video link to Presentation Video (Task 6):

https://youtu.be/Zobq5x8rUwk