

Developing Soft and Parallel Programming Skills

Using Project – Based Learning

CSC 3210 Spring – 2020

8 – Bit

Akash Dansinghani

Landon Wang

Raejae Sandy

Tony Ngo

### Work Breakdown Structure

Assignee Name	Email (studentID@student.gsu.edu)	Task	Duration (Hours)	Dependency	Due Date	Note
Akash Dansinghani	adansinghani1	Gathered and assembled the reports	2 hours on task, 11 hours on other project tasks	GitHub, Slack, tasks 3, and task 4 (all have to be done first to make the report)	2/07/20	Be ready 5 hours before due date
Landon Wang	lwang51	Group coordinator, and helped with ARM Assembling program	3 hours on task, 9 hours on other project tasks	None	2/07/20	Plan meet up times, and look over tasks
Raejae Sandy	Rsandy2	Set up GitHub and Slack accounts, and helped connect Pi to GitHub	7 hours on task, 9 hours on other project tasks	None	2/07/20	Send everyone the Slack and GitHub information
Shawn Martin	smartin93	WITHDRAWN FROM CLASS	--	--	--	--
Tony Ngo	tngo23	Made the YouTube Channel, recorded and edited the video	2 hours on task, 8 hours on other project tasks	None	2/07/20	Send everybody the YouTube account information

### **Task 3: Learning Teamwork Basics**

By: Akash Dansinghani

#### Work Norms:

Work was picked based upon first come basis (All tasks were available to everybody, but the team leader had first pick and the rest of the tasks were available to everyone else). The team leader will set the deadlines, but we all pick the deadlines collectively. We setup deadlines where if someone is behind the deadline, they have to submit it at their earliest convenience. Essentially, if we have a checkpoint due on Friday, we will set the deadline on Thursday so we have time to put everything together and get it to the team leader. We also check on each other to make sure everyone is on the same pace. We will review our own work, but we have a location where we put all of our assignments that are due. The team leader will ultimately check the assignment as well to make sure we did not make any mistakes, but we as a team, will look over each of the assignments that will be turned in. If we have differences in opinions on how to work on an assignment, we will resolve the issue by finding some sort of middle ground. If we are not able to hash out our problem and find a solution to our differences; we are forced to let the individual either go with solving the problem or task their own and learn from their mistakes or being able to compromise the situation beforehand.

#### Team Leader/Facilitator Norms:

For our project we are assigning a team leader or a facilitator for each of our checkpoints for the project. He or she will be responsible as well as everyone in the group to work on assigned tasks but will mainly focus on helping others who need help, assisting them where files are placed, and also outlining the parameters of each assignment to make sure each task is done to the best of the individuals ability. The facilitator is rotated for each checkpoint so everyone has the chance to help each other out and use their skills of a team leader to ensure everyone is completing their assignments in a timely manner as well as completing it properly. The responsibilities of a group facilitator include: Communicating with team members to assure assignments are being completed before the deadline, gathering all the required documents and assignments and making sure they meet the requirements of the rubric, contacting the TA responsible for receiving the assignments for the checkpoint, observing the assignments ahead of time, preferably when the project starts and having an outline for the expectations of each of the team members as well as the facilitator him or herself, and overall getting input from each of the team members and making sure decisions are made collectively rather than solely on the team leader.

#### Communication Norms:

Our group started using a group message using our numbers but we resorted to using GroupMe as it will be easier to communicate and send any form of documents especially since some of us are on different platforms (Android, iOS, Windows, etc.). We use our student email to contact each other for any other reason such as sending a document or communicating, but most of our files and tasks will be stored on GitHub. We may call each other if we are meeting outside of class or if someone has not arrived on time, we can contact them through their phone to confirm they are meeting the group on time.

### Scheduling/Meeting Norms:

We all gave our schedules when the project started and we are meeting during school hours as well as outside of class. We all live close to each other and if someone lives out of the radius of other team members, we are responsible for accommodating the individual(s) to find a place where we can meet at ease. During school hours, we do not have an issue since we all are in the vicinity, however if we are meeting outside of class, we are responsible collectively as a group to find a place (library or coffee shop of some sort) to discuss any issues that arise. We plan our meeting ahead of time and make sure each team member has it marked on their calendar to meet at the scheduled time. If for any reason the individual has any issues meeting, he or she should let the group know ahead of time to accommodate them or he or she should make time to contact the group to be up to date on deadlines and expectations for the assignments.

### Meeting Constrictions:

During our meeting we mainly focus on our project and our tasks that pertain to the project or class as a whole. We should eat before we meet or at the start of the meeting but we will not during the meeting unless we get a big part of the project done for the checkpoint and we all take a break together.

### Additional Questions:

- 1) What should you do to get the task accomplished and the team members' satisfactory high?

As mentioned previously, we should work on the task and make sure if we have any questions to discuss with our team members. If we start on a task before hand, we will be able to find our mistakes and correct them compared to if we were working on our task last minute.

- 2) As a team, select two cases out of the four mentioned in Handling Difficult Behavior.

**Argues:** If two or more people are arguing about a particular issue, then we will have to find some middle ground so we are able to find a resolution to the problem. Worst case scenario, if this involves an individual(s) who won't comply and thinks he is right and everyone else is wrong, then that individual would have to do it their own way and we would have to inform the instructor of the difficulties and the person will receive the grade he or she gets. After all, we are working as a team so we should succeed together.

**Complains:** If someone complains about someone else, they should let someone know, preferably the team coordinator and resolve the issue. We as a team should resolve problems, we may have but ideally, there should be no complaining, just coding.

- 3) If a team is having an issue resolving a problem, what should they do?

If we have differences in opinions on how to work on an assignment, we will resolve the issue by finding some sort of middle ground. If we are not able to hash out our problem and find a solution to our differences; we are forced to let the

individual either go with solving the problem or task their own and learn from their mistakes or being able to compromise the situation beforehand

- 4) Is it a good idea to pressure your team members if they do not work as fast you do?

Personally, its not the best idea. Everyone works at their own pace, everyone has a different schedule, and everyone has more or less work tolerance. That being said, there should be some sort of way where everybody can work efficiently and reach deadlines. If someone is known to turn in things late, give them an extra push and make sure they finish the task because you want to turn in a quality assignment rather than an assignment that was done, but could have lots of improvements.

- 5) What about if most people in the group want an “A” and one person wants a “B”?

The best way to motivate people is by them seeing everyone is working, so if everyone is putting effort into the project when they are working on it, then they will start to do the same. We also have a team coordinator who double checks the assignments and the team members can help as well. Ultimately, if the student if having a “B mindset”, we would have to communicate with the professor to let him know that the rest of the members are putting in more work than one individual.

## Learning Teamwork Basics

By: Landon Wang

**1) What to do to get the task accomplished *and* the team member's satisfaction high?**

In order to get the project tasks done and keep the satisfactions high, we, as a group must get to know everyone, and communicate with each and every member. Along with this, we also have to have rules set, know how to avoid/solve problems together, and have a facilitator assigned for each project (different facilitator for different project).

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

**a. Work Norms**

For these projects, work will be equally distributed among every member of the group. The tasks will also be rotated so that no one will continuously be doing the easy/hard or favorable tasks. The deadline of the tasks will be set by group coordinators, and members who continuously fail to abide by the deadline will be noted on the peer review. Final result of each task will be reviewed by everyone in the group. If any member has any disagreements over the quality of the task, the group will resolve the problem as a whole and in a professional way. If the final product of a task is not what everyone was expecting, the group coordinator will step in. It will not be a major problem if group members have different work habits as long as they get the work done by the set deadline with an acceptable quality.

**b. Facilitator Norms**

Our group have agreed to have the group coordinator to also be the facilitator of the group. The facilitator will have the power to set expectations and task deadlines (along with all the responsibilities stated in the Teamwork Basics document). Since the group coordinators are also the facilitators, it will be assigned when a new project rolls out.

**c. Communication Norms**

We have agreed as a group that our mode of communication will be through GroupMe. On GroupMe, we discuss assignments and make sure that every member is on the same page. We also have a set time for weekly in-person meetups, and individual meetups when requested.

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

**i. Argues**

In our group, feedbacks/critiques/counterarguments are always welcomed if they are expressed in an appropriate and respectful way. However, if any disrespect or inappropriate responses are made by any group members, the group coordinator and/or the professor will step in.

**ii. Complains**

Complains can be brought up to group coordinators if any member has concerns that they believe will affect the group productivity. It is then up

to the group coordinator to decide how or whether or not to address the issue with the group.

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

We agreed as a group to have the group coordinator to decide what is the best for the whole group when the group has trouble reaching an agreement. For example, if conflict arises during task assignments, the group coordinator will then be the one who breaks up the conflict and assign the tasks.

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

In our group, all decision is agreed upon every member. However, if this does become an issue, the group coordinator will step in to resolve it. If that does not work, we will turn to the professor.

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

If any member in the group decides to turn in a “B” quality assignment while everyone else turns in an “A” quality assignment, the group will discuss as a whole to figure out what needs to be done (with the group coordinator having the final say). That group member who turned in a “B” quality assignment will also be noted on the report if they fail to improve their work.





## Learning Teamwork Basics

By: Raejae Sandy

**1) What to do to get the task accomplished *and* the team member's satisfaction high?**  
Make sure the work presented is up to quality requested and have effective communication amongst group members.

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

**a. Work Norms**

Within this project we will see to it that work is distributed in a fair manner to the best of our abilities. Each task should have a complexity to it that requires an equal workload if enough effort is put forth, and our main goal is to develop together as a team regardless of individual experience. We are allowing the coordinator to oversee the group as a leader should and shall discuss disputes in an organized manner to effectively carry out each task. We're striving to allow different work styles as long as communication is present.

**b. Facilitator Norms**

Our group has designated the facilitator as the coordinator. This person will set the expectations and deadlines, and continuously push updates to ensure proper completion.

**c. Communication Norms**

We have designated days to be met equal to everyone's schedule; however, our main communication is set on group-me.

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

i. Argues: If arguing is present its potentially the failure of the group to truly mediate things, therefore a step back should be taken; however, anything outside of the norm should be taken to higher sources.

ii. Complains: This is something that should be handled by the coordinator to mediate and make judgement based off.

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

Everyone should be working to promote communication, and to get a conclusion; however, we've settled on allowing our coordinator to have the leadership role. Therefore, we will allow his judgement to take precedence.

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

Bring up such conflicts or complaints to the coordinator or other group members. We are prioritizing each group member being on the same page so our workflow is more uniformed.

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

Such things would be documented; however, it's been explicitly explained that such behavior must be taken up by the group to ensure the work to distribute be re-distributed to ensure proper work is completed.

### **CSC 3210 – Task 3: Learning Teamwork Basics**

By: Tony Ngo

#### **Task Accomplished & Team Members Satisfaction**

In order to get the task accomplished and get the team members' satisfaction high, we must get to know everybody in our group, set ground rules, use a facilitator, keep lines of communication open, and know how to avoid, or solve, common problems

#### **Work Norms**

Work will be distributed amongst all of our peers evenly through rotating tasks each assignment. The team leader will be setting the deadlines for our assignments. If someone does not follow through with their part of the commitment, we will penalize them in the peer review portion of the assignment. The work will be reviewed by all members of the team and if someone has different opinions about the quality of said work, then it will be handled amongst the peers in a professional manner; if the work is worse than expected then the group coordinator can get involved. If people have different work habits, I believe as long as you have a set-in stone timeframe when to complete assignments, it should not matter as long as the quality is great.

#### **Facilitator Norms**

Our facilitator will be our group coordinator, so they will set our expectations and deadlines along with everything else. The facilitator will also double as our group coordinator so it will be chosen when a new assignment comes out.

#### **Communication Norms**

Our group is communicating through GroupMe and we are constantly discussing as we do each part of our assignments and making sure we are all on the same page. We meet weekly also in person to discuss.

### Meeting Norms

Tony's Schedule:

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Not available	Work from 7 AM – 2 PM	Not available	Available after 1 PM	Available	Available	Available

Raejae's Schedule:

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Not available	Not available in PM b/c work	Available after 3	Available after 3	Available	Available	Available

Landon's Schedule:

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Available after 5	Available	Available after 5	Available after 3	Not Available	Available	Available

Aakash's Schedule:

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Available from 3-5	Available	Available from 3-5	Available after 3	Not Available	Available	Available

The group coordinator is responsible for choosing when the meeting is. Our preferences have varied so far since we have different things occurring every week. We usually hold meetings at the Georgia State Library. It is okay if someone is late to a meeting or meetings IF they let us know ahead of time and have a valid reason for missing the meeting. If someone misses multiple meetings in a row, we will ask the professor for guidance on what to do.

### Consideration Norms

People are able to eat at meetings but should not be able to smoke since we are in the school library. It should be fine if someone is dominating the conversation as long as it contributes positively to what we are trying to accomplish at the moment. All complaints with how everything is going can be brought up with the group coordinator at the time and changes will be made accordingly.

### Handling Difficult Behavior

1. Argues: If a team member brings up a counterpoint towards another team member respectfully and appropriately, it will be fine, but if a team member uses inappropriate manners towards another team member for a disagreement, we will bring it up to the professor.
2. Complains: A team member can complain to the group coordinator if they have a valid concern that they believe is affecting the productivity of our group and the group coordinator will decide if it is an appropriate measure to implement. If they are complaining about something that is not valid, then it will be up to the discretion of the group coordinator to decide what to do.

### Reaching a Consensus

We believe that it is up to the current group coordinator to decide what is best for the group. For example, if two members are conflicted between which task to do for the current assignment, the group coordinator will step in and decide which one they will do.

### Decision Making

This will not be an issue with our group because all of our decisions will have to be a unanimous decision amongst all of us. If this becomes an issue with a singular team member, we will go to the professor and ask for guidance.

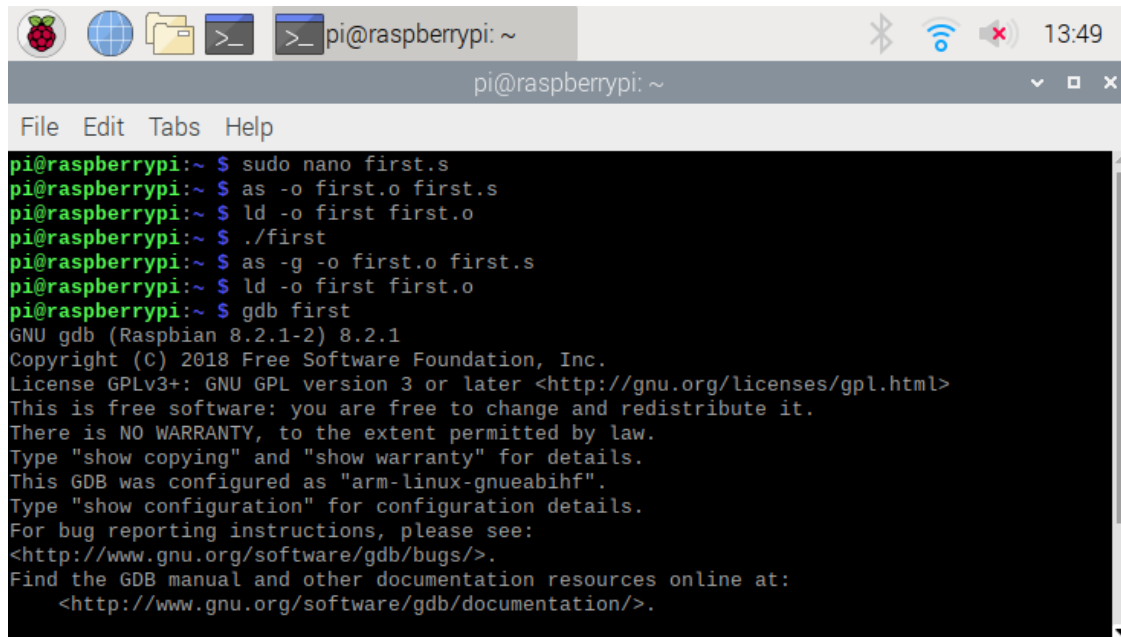
### “A” vs. “B”

If the singular team member turns in a “B” worthy assignment and the other members turn in an “A” worthy assignment, there is discussion that needs to be had amongst the group members to the group coordinator to figure out what is going on because that is not fair to the other 4-5 members that everybody gets to fall because of it.

## ARM Assembly Program Report

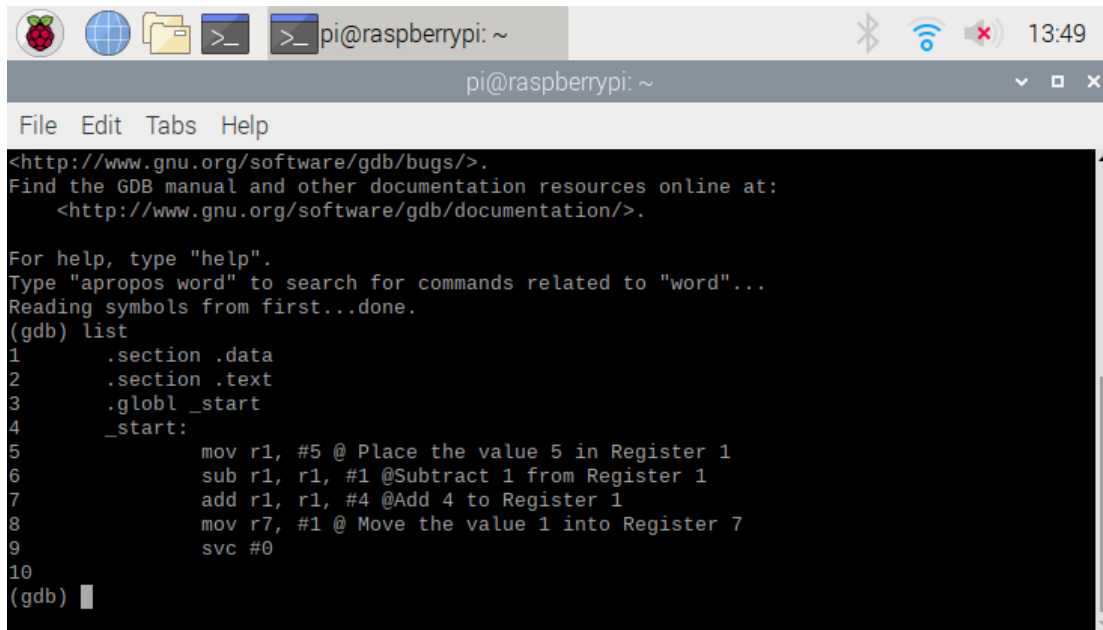
By: Akash Dansinghani

### Part 1: First Code



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo 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-gnueabi".  
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/>.
```

To preface, I used sudo before entering the code into the GNU because I was not able to run the code, but this allowed it to gain access. Also, this is what I wrote before entering the GDB for debugging. I started with “sudo nano first” which allowed me to input the code and allowing me to run the first program. After I saved the first.s file, I assembled the program and made the first.o program. After that, I linked the first file to first.o. Once this was complete, I ran the code “first”. Once no errors showed up I decided to assemble (using as -g -o first.o first.s) and link the program (ld -o first first.o) once again, but this time to be able to be used in the debugger. Then I ran the code in the debugger using “gdb first”.



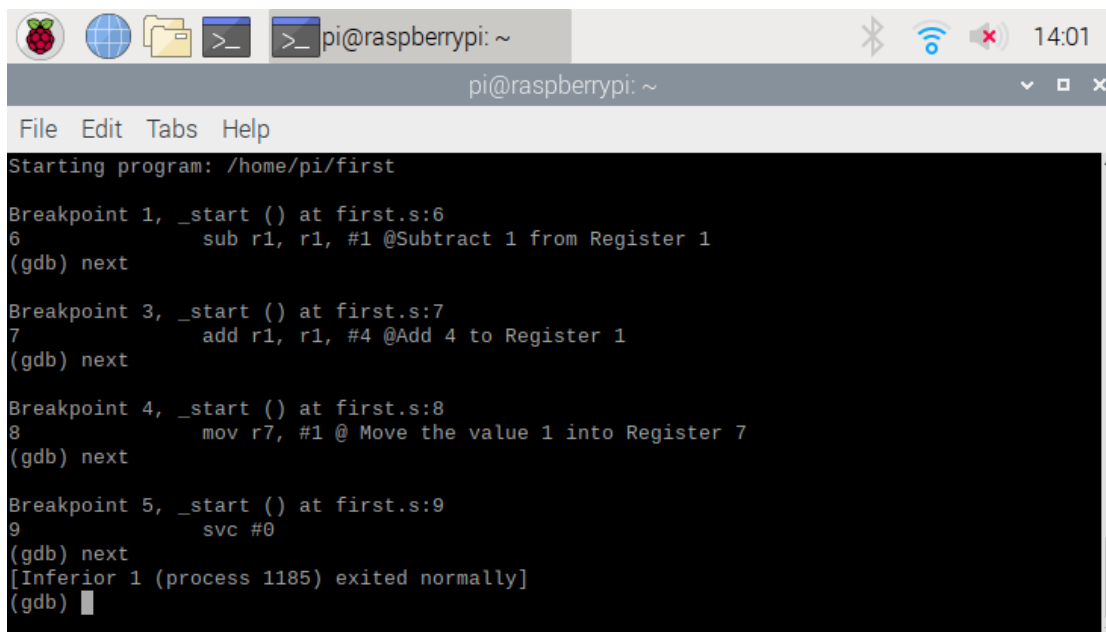
```

pi@raspberrypi: ~
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      .section .data
2      .section .text
3      .globl _start
4      _start:
5          mov r1, #5 @ Place the value 5 in Register 1
6          sub r1, r1, #1 @Subtract 1 from Register 1
7          add r1, r1, #4 @Add 4 to Register 1
8          mov r7, #1 @ Move the value 1 into Register 7
9          svc #0
10
(gdb)

```

This is the code that was inputted in the beginning and this was where I was able to set breakpoints to be able to run each line of code and see the result.

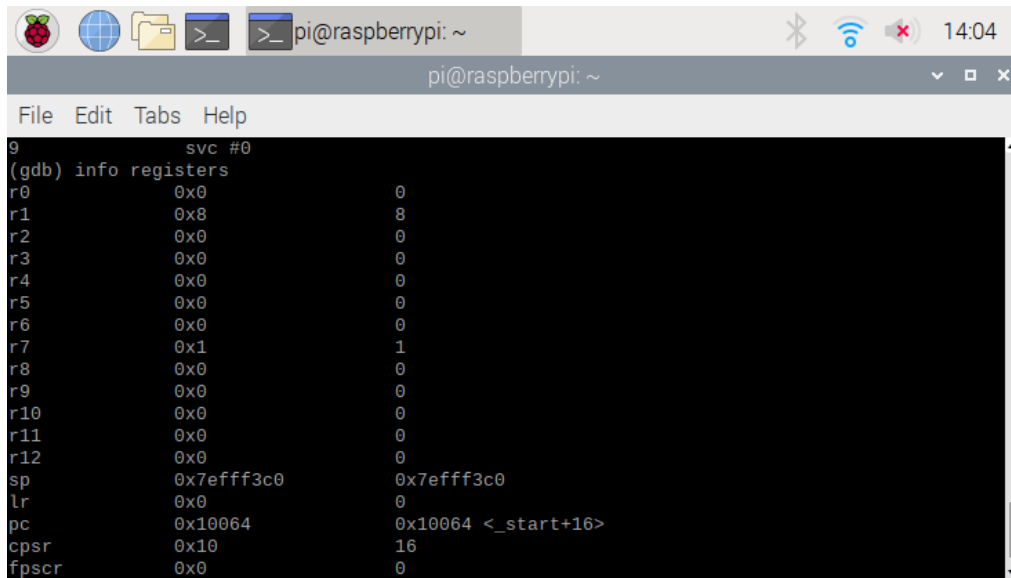


```

pi@raspberrypi: ~
File Edit Tabs Help
Starting program: /home/pi/first
Breakpoint 1, _start () at first.s:6
6          sub r1, r1, #1 @Subtract 1 from Register 1
(gdb) next
Breakpoint 3, _start () at first.s:7
7          add r1, r1, #4 @Add 4 to Register 1
(gdb) next
Breakpoint 4, _start () at first.s:8
8          mov r7, #1 @ Move the value 1 into Register 7
(gdb) next
Breakpoint 5, _start () at first.s:9
9          svc #0
(gdb) next
[Inferior 1 (process 1185) exited normally]
(gdb)

```

These are the breakpoints I set up and made sure each one ran before looking at the result of the registers.



The screenshot shows a terminal window titled "pi@raspberrypi: ~" with a menu bar (File, Edit, Tabs, Help). The terminal output is as follows:

```
9          svc #0
(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          0x7efff3c0    0x7efff3c0
lr          0x0          0
pc          0x10064       0x10064 <_start+16>
cpsr        0x10         16
fpscr       0x0          0
```

This was the result of the registers and was seen by typing in “info registers” in the debugger. As you can see in the code above, 5 was placed in Register 1, 5 was subtracted by 1 equalling 4 and finally 4 was added into Register 1 resulting in the value of 8 being stored in the register. Finally, the value 1 was put into Register 7 and we can see above that register is shows the same information.



## Part 2 : Arithmetic Code

```

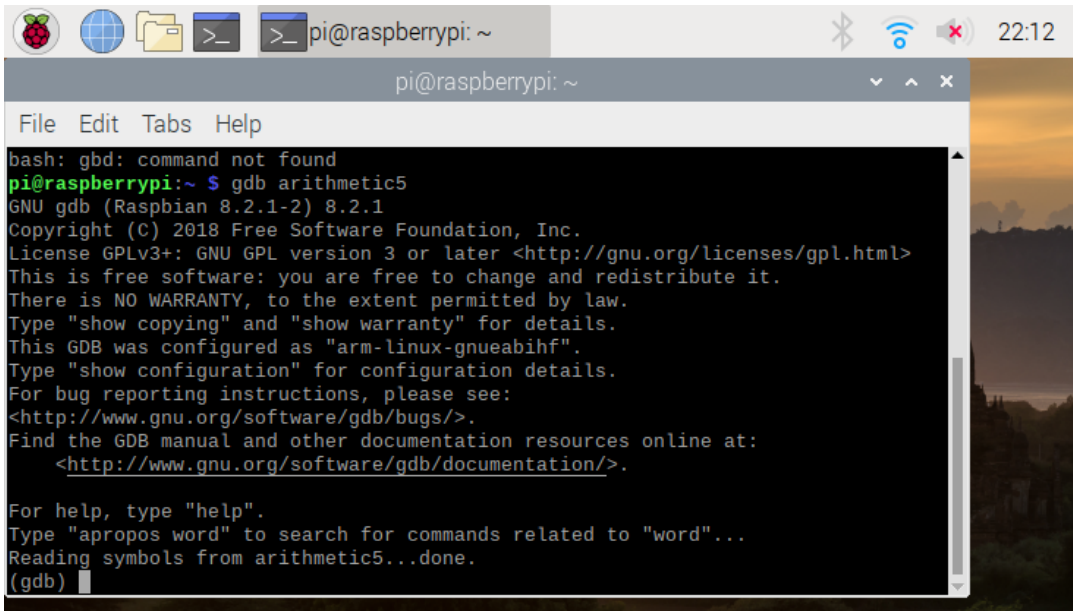
@arithmetic5 program
.section .data
.section .text
.global _start
_start:
    mov r1, #10 @ moving 10 to Register 1
    mov r2, #11 @ moving integer 11 to Register 2
    mov r3, #7 @ moving integer 7 to Register 3
    mov r4, #2 @ moving integer 2 to Register 4
    add r5, r1, r2 @ adding Registers 1 and Registers 2 and placing them in$
    mul r6, r3, r4 @ multiplying Registers 3 and Registers 4 and placing th$
    sub r1, r5, r6 @ subtracting Registers 6 and Registers 5 and placing th$
    mov r7, #1
    svc #0
  
```

This is the starter code for the file “arithmetic5.s” that was inserted in the GNU nano. As you can see, I also put comments to describe each line of code.

```

pi@raspberrypi:~$ sudo i
sudo: i: command not found
pi@raspberrypi:~$ sudo arithmetic5.s
sudo: arithmetic5.s: command not found
pi@raspberrypi:~$ sudo nano arithmetic5.s
pi@raspberrypi:~$ as -o arithmetic5.o arithmetic5.s
arithmetic5.s: Assembler messages:
arithmetic5.s:13: Error: bad arguments to instruction -- `mov r7#1'
pi@raspberrypi:~$ nano arithmetic5.s
pi@raspberrypi:~$ as -o arithmetic5.o arithmetic5.s
pi@raspberrypi:~$ ld -o arithmetic5 arithmetic5.o
pi@raspberrypi:~$ ./arithmetic5
pi@raspberrypi:~$ as -g -o arithmetic5.o arithmetic5.s
pi@raspberrypi:~$ ld -o arithmetic5 arithmetic5.o
pi@raspberrypi:~$ gdb arithmetic5
bash: gdb: command not found
pi@raspberrypi:~$ gdb arithmetic5
  
```

this is what I wrote before entering the GDB for debugging. I started with “sudo nano arithmetic5.s” which allowed me to input the code and allowing me to fun the first program. After I saved the arithmetic5.s file, I assembled the program and made the arithmetic5.o program. After that, I linked the first file to arithmetic5.o. Once this was complete, I ran the code “arithmetic5”. Once no errors showed up I decided to assemble (using as -g -o arithmetic5.o arithmetic5.s) and link the program (ld -o arithmetic5 arithmetic5.o) once again, but this time to be able to be used in the debugger. Then I ran the code in the debugger using “gdb arithmetic5”.



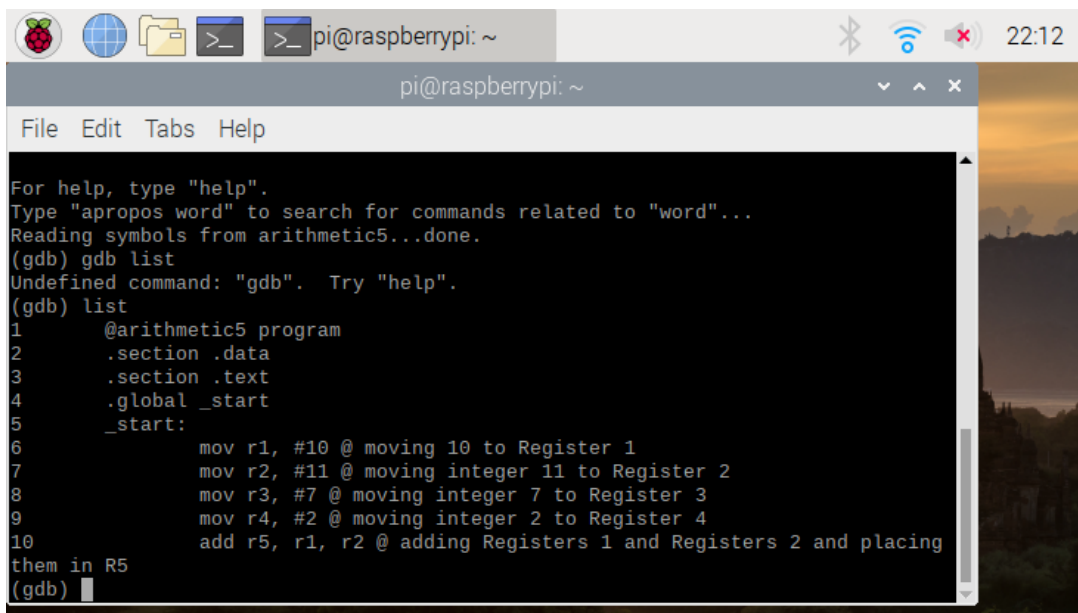
```

pi@raspberrypi: ~
File Edit Tabs Help
bash: gbd: command not found
pi@raspberrypi:~ $ gdb arithmetic5
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-gnueabi".
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 arithmetic5...done.
(gdb)

```

This is the screen when I typed in “gdb arithmetic5” and launched the debugger.



```

pi@raspberrypi: ~
File Edit Tabs Help
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from arithmetic5...done.
(gdb) gdb list
Undefined command: "gdb". Try "help".
(gdb) list
1      @arithmetic5 program
2      .section .data
3      .section .text
4      .global _start
5      _start:
6          mov r1, #10 @ moving 10 to Register 1
7          mov r2, #11 @ moving integer 11 to Register 2
8          mov r3, #7 @ moving integer 7 to Register 3
9          mov r4, #2 @ moving integer 2 to Register 4
10         add r5, r1, r2 @ adding Registers 1 and Registers 2 and placing
them in R5
(gdb)

```

This is where the code is listed so we are able to add breakpoints for each line.

```

pi@raspberrypi: ~
File Edit Tabs Help
6      mov r1, #10 @ moving 10 to Register 1
7      mov r2, #11 @ moving integer 11 to Register 2
8      mov r3, #7 @ moving integer 7 to Register 3
9      mov r4, #2 @ moving integer 2 to Register 4
10     add r5, r1, r2 @ adding Registers 1 and Registers 2 and placing
them in R5
(gdb) b 5
Breakpoint 1 at 0x10058: file arithmetic5.s, line 7.
(gdb) b 8
Breakpoint 2 at 0x1005c: file arithmetic5.s, line 8.
(gdb) b 9
Breakpoint 3 at 0x10060: file arithmetic5.s, line 9.
(gdb) b 10
Breakpoint 4 at 0x10064: file arithmetic5.s, line 10.
(gdb) b 11
Breakpoint 5 at 0x10068: file arithmetic5.s, line 11.
(gdb) b 12
Breakpoint 6 at 0x1006c: file arithmetic5.s, line 12.
(gdb)

```

As you can see the breakpoints are being added so we can run each line of code.

```

pi@raspberrypi: ~
File Edit Tabs Help
(gdb) info registers
13      mov r7, #1
(gdb) info registers
r0      0x0          0
r1      0x7          7
r2      0xb         11
r3      0x7          7
r4      0x2          2
r5      0x15         21
r6      0xe         14
r7      0x0          0
r8      0x0          0
r9      0x0          0
r10     0x0          0
r11     0x0          0
r12     0x0          0
sp      0x7efff3b0   0x7efff3b0
lr      0x0          0
pc      0x10070      0x10070 <_start+28>
cpsr    0x10        16
fpscr   0x0          0

```

This is the result of the code. Each one of the lines of code were executed where the value 10 was placed in Register 1, the value 11 was placed in Register 2, the value 7 was placed in Register 3, the value 2 was placed in Register 4, and adding the contents of Register 1 and Register 2 which is 21 and that value was placed in Register 5. Finally, Registers 2 and Register 4 were multiplied and the value of 14 was stored in Register 6.

### Overview of Assignment and What I Learned:

Connecting my Raspberry Pi was difficult and since I have very little experience using the device and never using GitHub in general. With the help of my team, I was able to finally connect my Raspberry Pi to GitHub and access the repository. Lastly, when I was creating the code for the arithmetic part of the program, I ran into an issue being able to run the program, but with the help on my team members, who were able to assist me on what I needed to fix in my code in the nano.

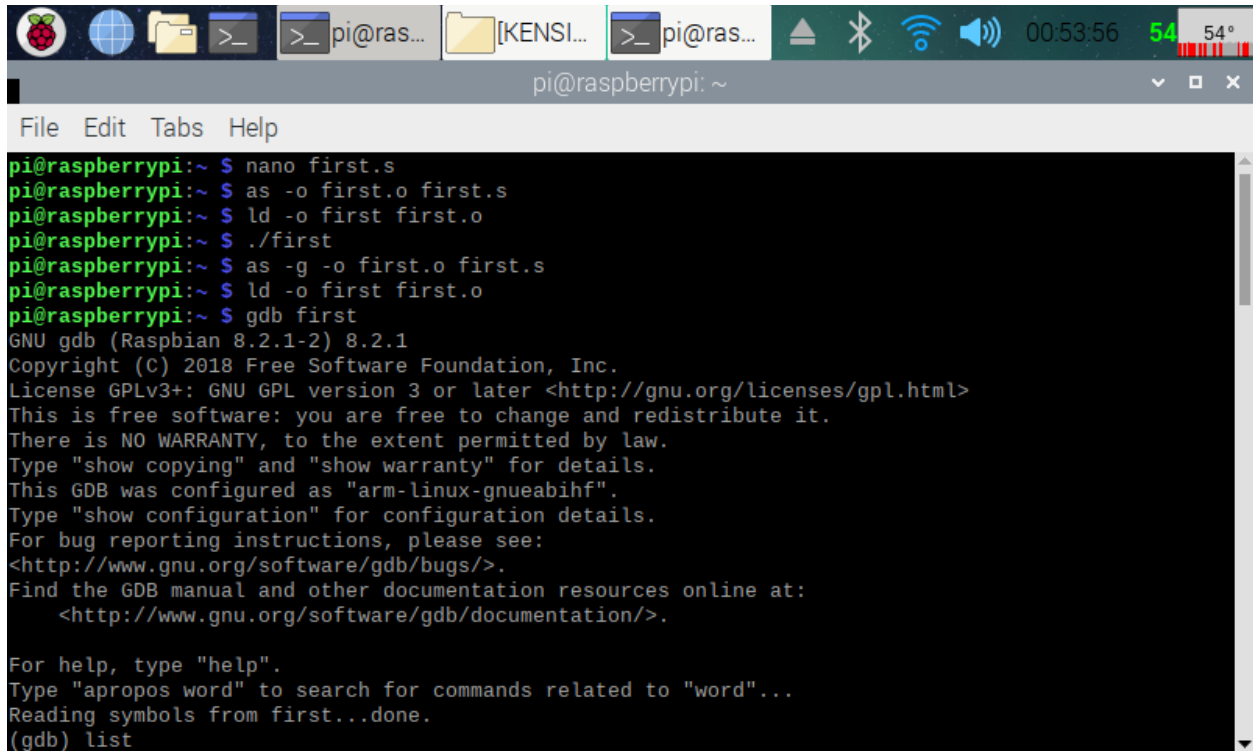
Each of the pictures have a description of what processes occurred between each code. I was able to put values into assigned registers, perform operations between two register, and move the final value in the assigned register. To be able to do this, there is a somewhat of a long and lengthy process of inputting the code, assembling the file, linking the file, and finally running the file. After this to start debugging, you how to assemble the file again, link the file again, and load it into the debugger which was done by putting it in the GDB. After this, you are able to assign breakpoints and run through each line of code (or at least where the breakpoints were placed) and be able to see the out by typing “info registers”. After we made sure each of the values were in the proper register, we were able to leave the debugger.

I learned many things while completing the assignment. For instance, I was able to navigate through the Raspberry Pi, link GitHub through the terminal, execute different programs through the terminal, place files that were stored on the Raspberry Pi and upload them on GitHub. Lastly, I learned a different form of coding where you can add the values of two registers and place them in a different register all in one line of code.

## ARM Assembler in Raspberry Pi

By: Landon Wang

### Part 1: First Program:



```

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-gnueabi".
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

```

Here (in the screenshot above), I have created the First assembly program (nano first.s). I then assembled the program (as -o first.o first.s) and linked it (ld -o first first.s). After that, I ran the program (./first), and no output was displayed, because any program that manipulated data between the CPU registers and memory will not use the IO. To see what the program is doing, I used the GDB (GNU Debugger). To do this, I used the debugger to link the machine code to the source code (as -g -o first.o first.s), then assembled it like I did before. I then ran the code with the debugger (gdb first), which displayed some of the debugger information before continuing.

```

(gdb) list
1      @ first program
2      .section .data
3      .section .text
4      .global _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
10         mov r7, #1      @Program Termination: exit syscall
(gdb)
11         svc #0          @Program Termination: wake kernel
12     .end
(gdb)
Line number 13 out of range; first.s has 12 lines.
(gdb) b 6
Breakpoint 1 at 0x10058: file first.s, line 7.
(gdb) b 7
Note: breakpoint 1 also set at pc 0x10058.
Breakpoint 2 at 0x10058: file first.s, line 7.
(gdb) b 87
No line 87 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) n

```

Here (in the screenshot above), I displayed my code in the debugger ((gdb) list), and added breakpoints in between the lines of the codes ((gdb) b “a line number”). This way I can see what is happening in the registry later.

```

Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) b 8
Breakpoint 3 at 0x1005c: file first.s, line 8.
(gdb) b 9
Breakpoint 4 at 0x10060: file first.s, line 10.
(gdb) b 10
Note: breakpoint 4 also set at pc 0x10060.
Breakpoint 5 at 0x10060: file first.s, line 10.
(gdb) b 11
Breakpoint 6 at 0x10064: file first.s, line 11.
(gdb) run
Starting program: /home/pi/first

Breakpoint 1, _start () at first.s:7
7       sub r1, r1, #1 @subtract 1 from r1
(gdb) b 5
Note: breakpoints 1 and 2 also set at pc 0x10058.
Breakpoint 7 at 0x10058: file first.s, line 7.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pi/first

Breakpoint 1, _start () at first.s:7

```

Here (in the two screenshots above), I was playing around with the breakpoints, and after I was done adding breakpoints to the lines that I wanted, I ran the debugger ((gdb) run). The debugger then goes through the first breakpoint. I then pulled up the register information ((gdb) info registers).

```

Breakpoint 1, _start () at first.s:7
7       sub r1, r1, #1 @subtract 1 from r1
(gdb) b 5
Note: breakpoints 1 and 2 also set at pc 0x10058.
Breakpoint 7 at 0x10058: file first.s, line 7.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pi/first

Breakpoint 1, _start () at first.s:7
7       sub r1, r1, #1 @subtract 1 from r1
(gdb)
(gdb) info registers
r0             0x0             0
r1             0x5             5
r2             0x0             0
r3             0x0             0
r4             0x0             0
r5             0x0             0
r6             0x0             0
r7             0x0             0
r8             0x0             0
r9             0x0             0

```

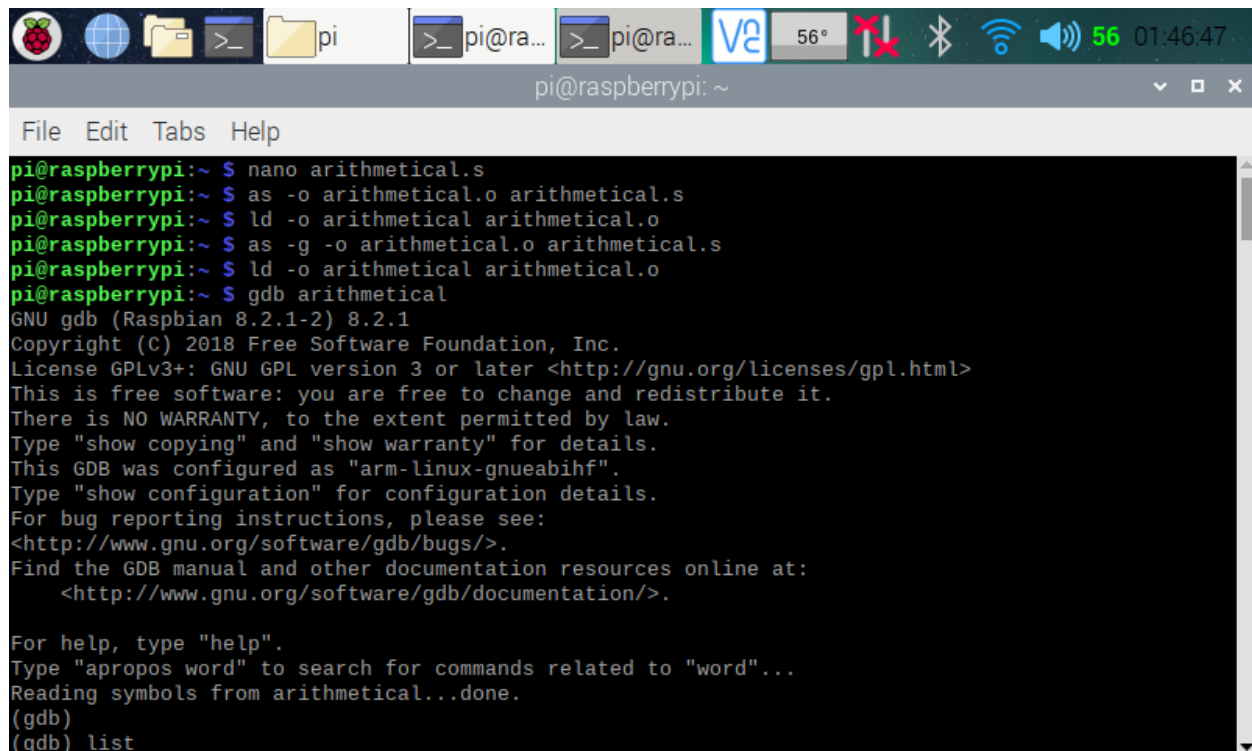
```

Breakpoint 1, _start () at first.s:7
7          sub r1, r1, #1 @subtract 1 from r1
(gdb)
(gdb) info registers
r0             0x0             0
r1             0x5             5
r2             0x0             0
r3             0x0             0
r4             0x0             0
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             0x10058         0x10058 <_start+4>
cpsr           0x10           16
fpscr          0x0             0
(gdb)

```

Here (in the screenshot above), I know that the code is working, because after going through the first breakpoint, you can see that 5 was indeed added into register 1 (r1) by the code (mov r1, #5). This is the end of part one.

## **Part 2: Arithmetical Program:**



```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ nano arithmetical.s
pi@raspberrypi:~ $ as -o arithmetical.o arithmetical.s
pi@raspberrypi:~ $ ld -o arithmetical arithmetical.o
pi@raspberrypi:~ $ as -g -o arithmetical.o arithmetical.s
pi@raspberrypi:~ $ ld -o arithmetical arithmetical.o
pi@raspberrypi:~ $ gdb arithmetical
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-gnueabi".
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 arithmetical...done.
(gdb)
(gdb) list

```

Here (in the screenshot above), I made the arithmetical program (nano arithmetical.s). I then assembled, linked and ran it with the GDB debugger like I did with the First program.



```

(gdb) list
1      @arithmetical program
2      .section .data
3      .section .text
4      .global _start
5      _start:
6          mov r1, #10
7          mov r2, #11
8          mov r3, #7
9          mov r4, #2
10         add r5, r1, r2
(gdb) b 6
Breakpoint 1 at 0x10058: file arithmetical.s, line 7.
(gdb) b 8
Breakpoint 2 at 0x1005c: file arithmetical.s, line 8.
(gdb) b 9
Breakpoint 3 at 0x10060: file arithmetical.s, line 9.
(gdb) b 10
Breakpoint 4 at 0x10064: file arithmetical.s, line 10.
(gdb) b 11
Breakpoint 5 at 0x10068: file arithmetical.s, line 11.
(gdb) b 12
Breakpoint 6 at 0x1006c: file arithmetical.s, line 12.
(gdb)

```

Here (in the screenshot above), I pulled up the code on the debugger and added in the breakpoints like I did in part 1.

Here (in the screenshot above), I ran the code through the debugger and pulled up the register information. In the register, you see that, after executing line 6 (with the breakpoint moving on to line 7 [(gdb) next]) of the code, the number 10 has been added into register one

```

(gdb)
Note: breakpoint 6 also set at pc 0x1006c.
Breakpoint 7 at 0x1006c: file arithmetical.s, line 12.
(gdb) run
Starting program: /home/pi/arithmetical

Breakpoint 1, _start () at arithmetical.s:7
7          mov r2, #11
(gdb) info registers
r0             0x0             0
r1             0xa            10
r2             0x0             0
r3             0x0             0
r4             0x0             0
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             0x7efff3b0      0x7efff3b0
lr             0x0             0

```

(mov r1, #10). Register one shows 0xa, with “a” representing 10 in hexadecimal format. On the column over, 10 being showed in decimal format.

```

lr          0x0          0
pc          0x10058      0x10058 <_start+4>
cpsr        0x10        16
fpscr       0x0          0
(gdb) next

Breakpoint 2, _start () at arithmetical.s:8
8          mov r3, #7
(gdb) info registers
r0          0x0          0
r1          0xa          10
r2          0xb          11
r3          0x0          0
r4          0x0          0
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          0x7efff3b0    0x7efff3b0
lr          0x0          0

```

Here (in the screenshot above), in the register, you see that, after executing line 7 (with the breakpoint moving on to line 8) of the code, the number 11 has been added into register two (mov r2, #11). Register two shows 0xb, with “b” representing 11 in hexadecimal format. On the column over, 11 is being displayed in decimal format.

```

lr          0x0          0
pc          0x1005c      0x1005c <_start+8>
cpsr        0x10        16
fpscr       0x0          0
(gdb) next

Breakpoint 3, _start () at arithmetical.s:9
9          mov r4, #2
(gdb) info registers
r0          0x0          0
r1          0xa          10
r2          0xb          11
r3          0x7          7
r4          0x0          0
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          0x7efff3b0    0x7efff3b0
lr          0x0          0

```

Here (in the screenshot above), in the register, you see that, after executing line 8 (with the breakpoint moving on to line 9) of the code, the number 7 has been added into register three (mov r3, #7). Register three shows 0x7, with “7” representing 7 in hexadecimal format. On the column over, 7 is being displayed in decimal format.

```

lr      0x0      0
pc      0x10060   0x10060 <_start+12>
cpsr    0x10     16
fpscr   0x0      0
(gdb) next

Breakpoint 4, _start () at arithmetical.s:10
10      add r5, r1, r2
(gdb) info registers
r0      0x0      0
r1      0xa      10
r2      0xb      11
r3      0x7      7
r4      0x2      2
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      0x7efff3b0 0x7efff3b0
lr      0x0      0

```

Here (in the screenshot above), in the register, you see that, after executing line 9 (with the breakpoint moving on to line 10) of the code, the number 2 has been added into register four (mov r4, #2). Register four shows 0x2, with “2” representing 2 in hexadecimal format. On the column over, 2 is being displayed in decimal format.

```

lr      0x0      0
pc      0x10064   0x10064 <_start+16>
cpsr    0x10     16
fpscr   0x0      0
(gdb) next

Breakpoint 5, _start () at arithmetical.s:11
11      mul r6, r3, r4
(gdb) info registers
r0      0x0      0
r1      0xa      10
r2      0xb      11
r3      0x7      7
r4      0x2      2
r5      0x15     21
r6      0x0      0
r7      0x0      0
r8      0x0      0
r9      0x0      0
r10     0x0      0
r11     0x0      0
r12     0x0      0
sp      0x7efff3b0 0x7efff3b0
lr      0x0      0

```

Here (in the screenshot above), in the register, you see that, after executing line 10 (with the breakpoint moving on to line 11) of the code, the number 21 has been added into register five (add r5, r1, r2). Register four shows 0x15, with “15” representing 21 in hexadecimal format. On the column over, 21 is being displayed in decimal format. We are getting 21 because we added r2 (11) to r1 (10), which gave us 21, and stored it into r5.

```

lr          0x0          0
pc          0x10068      0x10068 <_start+20>
cpsr        0x10        16
fpscr       0x0         0
(gdb) next

Breakpoint 6, _start () at arithmetical.s:12
12      sub r1, r5, r6
(gdb) info registers
r0          0x0          0
r1          0xa          10
r2          0xb          11
r3          0x7          7
r4          0x2          2
r5          0x15         21
r6          0xe          14
r7          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x0          0
r12         0x0          0
sp          0x7efff3b0    0x7efff3b0
lr          0x0          0

```

Here (in the screenshot above), in the register, you see that, after executing line 11 (with the breakpoint moving on to line 12) of the code, the number 14 has been added into register six (mul r6, r3, r4). Register six shows 0xe, with “e” representing 14 in hexadecimal format. On the column over, 14 is being displayed in decimal format. We are getting 14 because we multiplied r3 (7) by r4 (2), which gave us 14, and stored it into r6.

```

lr          0x0          0
pc          0x1006c      0x1006c <_start+24>
cpsr        0x10        16
fpscr       0x0         0
(gdb) next
0x00010070 in ?? ()
(gdb) info registers
r0          0x0          0
r1          0x7          7
r2          0xb          11
r3          0x7          7
r4          0x2          2
r5          0x15         21
r6          0xe          14
r7          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x0          0
r12         0x0          0
sp          0x7efff3b0    0x7efff3b0
lr          0x0          0
pc          0x10070      0x10070
cpsr        0x10        16

```

Here (in the screenshot above), in the register, you see that, after executing line 12 (with the breakpoint moving on to line 13) of the code, the number 7 has been added into register one (sub r1, r5, r6). Register one shows 0x7, with “7” representing 7 in hexadecimal format. On the column over, 7 is being displayed in decimal format. We are getting 7 because we subtracted r6 (14) from r5 (21), which gave us 7, and stored it into r1.

```

r2      0xb      11
r3      0x7      7
r4      0x2      2
r5      0x15     21
r6      0xe      14
r7      0x0      0
r8      0x0      0
r9      0x0      0
r10     0x0      0
r11     0x0      0
r12     0x0      0
sp      0x7efff3b0 0x7efff3b0
lr      0x0      0
pc      0x10070   0x10070
cpsr    0x10     16
fpscr   0x0      0
(gdb) quit
A debugging session is active.

```

Inferior 1 [process 2929] will be killed.

```

Quit anyway? (y or n) y
pi@raspberrypi:~ $

```

Here (in the screenshot above), after knowing that the program runs correctly, I left the debugger ((gdb) quit).

```

GNU nano 3.2      arithmetical.s      Modified
@arithmetical program
.section .data
.section .text
.global _start
_start:
    mov r1, #10    @load registry 1 with 10
    mov r2, #11    @load registry 2 with 11
    mov r3, #7     @load registry 3 with 07
    mov r4, #2     @load registry 4 with 02
    add r5, r1, r2 @add values in registries 1 and 2 together
                    @and store it into registry 5
    mul r6, r3, r4 @multiply values in registries 3 and 4 together
                    @and store it into registry 6
    sub r1, r5, r6 @subtract values in registries 5 and 6 and store
                    @it into registry 1
.end

```

17 items (20 hidden) Free space: 21.0 GiB (Total: 27.4 GiB)

Here (in the screenshot above), I went back to the arithmetical program and commented through my code. This is the end of part two.

## ARM Assembly Report

By: Raejae Sandy

```

pi@raspberrypi: ~/Raejae/8-Bit/Raejae
File Edit Tabs Help
pi@raspberrypi:~/Raejae/8-Bit/Raejae $ ld -o first first.o
pi@raspberrypi:~/Raejae/8-Bit/Raejae $ 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-gnueabi".
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
1      @first program
2      .section data
3      .section .text
4      .globl _start
5      _start:
6
7      mov r1, #5 @load r1 with 5
8      sub r1, r1, #1 @subtract 1 from r1
9      add r1, r1, #4 @add 4 to r1
10
11     mov r7,#1    @Program Termination : exit syscall
(gdb) b 11
Breakpoint 1 at 0x10064: file first.s, line 11.
(gdb) info registers
The program has no registers now.
(gdb) run
Starting program: /home/pi/Raejae/8-Bit/Raejae/first

Breakpoint 1, _start () at first.s:11
11     svc #0      @Program Termination : wake kernel
(gdb) b 6
Breakpoint 2 at 0x10058: file first.s, line 7.
(gdb) run

```

Basic Initialization of code.

```

pi@raspberrypi: ~/Raejae/8-Bit/Raejae
File Edit Tabs Help
(gdb) list
1      @first program
2      .section data
3      .section .text
4      .globl _start
5      _start:
6
7      mov r1, #5 @load r1 with 5
8      sub r1, r1, #1 @subtract 1 from r1
9      add r1, r1, #4 @add 4 to r1
10
11     mov r7,#1    @Program Termination : exit syscall
(gdb) b 11
Breakpoint 1 at 0x10064: file first.s, line 11.
(gdb) info registers
The program has no registers now.
(gdb) run
Starting program: /home/pi/Raejae/8-Bit/Raejae/first

Breakpoint 1, _start () at first.s:11
11     svc #0      @Program Termination : wake kernel
(gdb) b 6
Breakpoint 2 at 0x10058: file first.s, line 7.
(gdb) run

```

This is applying breakpoints. By adding a breakpoint at beginning and end it allows us to continuously run through our code in debugger line by line.

The image consists of two screenshots of a Raspberry Pi terminal window, showing the execution of a program named 'first.s' using GDB. The terminal window has a title bar that reads 'pi@raspberrypi: ~/Raejae/8-Bit/Raejae'. The top panel shows the initial state of the program, and the bottom panel shows the state after a few instructions.

**Top Screenshot:**

```

Breakpoint 2 at 0x10058: file first.s, line 7.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pi/Raejae/8-Bit/Raejae/first

Breakpoint 2, _start () at first.s:7
7      sub r1, r1, #1 @subtract 1 from r1
(gdb) info registers
r0          0x0          0
r1          0x5          5
r2          0x0          0
r3          0x0          0
r4          0x0          0
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          0x7efff360    0x7efff360

```

**Bottom Screenshot:**

```

(gdb) next
8      add r1, r1, #4 @add 4 to r1
(gdb) info registers
r0          0x0          0
r1          0x4          4
r2          0x0          0
r3          0x0          0
r4          0x0          0
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          0x7efff360    0x7efff360
lr          0x0          0
pc          0x1005c      0x1005c <_start+8>
cpsr       0x10         16
fpscr       0x0          0
(gdb) next
10     mov r7, #1 @Program Termination : exit syscall

```

On this line we load 5 to the Register (r1) : and afterwards we subtract 1 from the same register which lands us on 4 for the first register.

```

(gdb) next
10      mov r7, #1      @Program Termination : exit syscall
(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      0x0            0
r8      0x0            0
r9      0x0            0
r10     0x0            0
r11     0x0            0
r12     0x0            0
sp      0x7efff360     0x7efff360
lr      0x0            0
pc      0x10060        0x10060 <_start+12>
cpsr    0x10          16
fpscr   0x0            0
(gdb) next

```

```

(gdb) next
Breakpoint 1, _start () at first.s:11
11      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            0
r7      0x1            1
r8      0x0            0
r9      0x0            0
r10     0x0            0
r11     0x0            0
r12     0x0            0
sp      0x7efff360     0x7efff360
lr      0x0            0
pc      0x10064        0x10064 <_start+16>
cpsr    0x10          16
fpscr   0x0            0

```

On the lines above we add 4 to the R1 register which updates this register and then we load to R7 #1.



The image consists of two screenshots of a Raspberry Pi terminal window, showing the GDB debugger interface. The terminal window has a title bar with the Raspberry Pi logo and the path `pi@raspberrypi: ~/Raejae/8-Bit/Raejae`. The window contains a file explorer on the right side with files like `.bashrc` and `.xsession-errors.old`.

**Top Screenshot:** The GDB prompt is at line 9. The registers are listed as follows:

```

9          @B : R2
10         @C : R3
(gdb)
11         @D : R4
12         @A : 10
13         @B : 11
14         @C : 7
15         @D : 2
16
17         mov r1, #10
18         mov r2, #11
19         mov r3, #7
20         mov r4, #2
(gdb)
21         @Register's Loaded
22
23         add r5, r1, r2 @Add's Registers (A + B) To R1 interact with R1 &
24         R2
25
26         mul r6, r3, r4
27
28         sub r1, r5, r6
29

```

**Bottom Screenshot:** The GDB prompt is at line 22. The registers are listed as follows:

```

22
23         add r5, r1, r2 @Add's Registers (A + B) To R1 interact with R1 &
24         R2
25
26         mul r6, r3, r4
27
28         sub r1, r5, r6
29
30         mov r7, #1 @Program Termination : exit syscall
31         svc #0 @Program Termination : wake kernel
(gdb)
32         .end
(gdb) b 17
Breakpoint 1 at 0x10058: file aritmetic1.s, line 18.
(gdb) b 18
Note: breakpoint 1 also set at pc 0x10058.
Breakpoint 2 at 0x10058: file aritmetic1.s, line 18.
(gdb) b 19
Breakpoint 3 at 0x1005c: file aritmetic1.s, line 19.
(gdb) b 20
Breakpoint 4 at 0x10060: file aritmetic1.s, line 20.
(gdb) b 21

```

In the two snaps above, after loading in Arithmetic similarly to “First.” After setting breakpoints manually I realized that you could set the initial and end and the program will run through. Unfortunately, I realized too late and found myself taking a long route.

```

pi@raspberrypi: ~/Raejae/8-Bit/Raejae
File Edit Tabs Help
Breakpoint 5, _start () at arithmetic1.s:23
23      add r5, r1, r2 @Add's Registers (A + B) To R1 interact with R1 &
R2
(gdb) info registers
r0          0x0          0
r1          0xa          10
r2          0xb          11
r3          0x7          7
r4          0x2          2
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          0x7efff360   0x7efff360
lr          0x0          0
pc          0x10064      0x10064 <_start+16>
cpsr        0x10        16
fpscr       0x0          0
(gdb) next
2020-02-06-183748_800x480_screenshot.png (36.6 KiB) PNG image
Free space: 21.8 GiB (Total: 27.4 GiB)

```

The first few lines of code are simply dedicated to moving numbers in the registers but an important thing to notice is that in r1 and r2 there is a 0xa / 0xb which lets us know this is hexadecimal representation.

```

pi@raspberrypi: ~/Raejae/8-Bit/Raejae
File Edit Tabs Help
Breakpoint 10, _start () at arithmetic1.s:29
29      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          0x15         21
r6          0xe          14
r7          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x0          0
r12         0x0          0
sp          0x7efff360   0x7efff360
lr          0x0          0
pc          0x10070      0x10070 <_start+28>
cpsr        0x10        16
fpscr       0x0          0
(gdb) next
2020-02-06-183748_800x480_screenshot.png (36.6 KiB) PNG image
Free space: 21.8 GiB (Total: 27.4 GiB)

```

This is the result after the function calls ADD SUB and MUL which manipulate the registers and updates each answer to a new register below its calling

## CSC 3210 Project A1 Task 4

By: Tony Ngo

### Task A: Connecting to GitHub via SSH

My personal experience while trying to install GitHub onto the Raspberry Pi was seamless except that my Pi would not connect to SSH at first, but you can change that in the settings.

### Task B: ARM Assembly Programming

#### Part 1: First Program

```

root@raspberrypi:~/test.git# nano first.s
root@raspberrypi:~/test.git# as -o first.o first.s
root@raspberrypi:~/test.git# ld -o first first.o
root@raspberrypi:~/test.git# ./first
root@raspberrypi:~/test.git# as -g -o first.o first.s
root@raspberrypi:~/test.git# ld -o first first.o
root@raspberrypi:~/test.git# 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-gnueabi".
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)

```

Picture 1: Assembly, Link,

Run, & Beginning of debugging

```

(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
10         mov r7, #1      @ Program Termination: exit syscall
(gdb)
11         svc #0          @ Program Termination: wake kernel
12     .end
(gdb) b 11
Breakpoint 1 at 0x10064: file first.s, line 11.
(gdb) run
Starting program: /root/test.git/first

Breakpoint 1, _start () at first.s:11
11         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           0
r7          0x1           1
r8          0x0           0
r9          0x0           0
r10         0x0           0
r11         0x0           0
r12         0x0           0
sp          0x7efff740    0x7efff740
lr          0x0           0
pc          0x10064       0x10064 <_start+16>
cpsr       0x10         16
fpscr      0x0           0
(gdb)

```

Picture 2: Debugging of First program

## Part 2: Arithmetic Program

```

root@raspberrypi:~/test.git# nano arithmetic1.s
root@raspberrypi:~/test.git# as -o arithmetic1.o arithmetic1.s
root@raspberrypi:~/test.git# ld -o arithmetic1 arithmetic1.o
root@raspberrypi:~/test.git# ./arithmetic1
root@raspberrypi:~/test.git# as -g -o arithmetic1.o arithmetic1.s
root@raspberrypi:~/test.git# ld -o arithmetic1 arithmetic1.o
root@raspberrypi:~/test.git# gdb arithmetic1

```

Picture 1: Assembly, Link, Run, &amp; Beginning

of debugging

```

(gdb) list
1      @ Arithmetic 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
(gdb)
11          add r1, r2 @ add r2 to r1
12          mul r3, r4 @ multiply r4 to r3
13          sub r1, r3 @ subtract r3 from r1
14
15          mov r7, #1 @ Program Termination: exit syscall
16          svc #0     @ Program Termination: wake kernel
17      .end
(gdb) b 15
Breakpoint 1 at 0x10070: file arithmetic1.s, line 15.
(gdb) run
Starting program: /root/arithmetic1

Breakpoint 1, _start () at arithmetic1.s:15
15          mov r7, #1 @ Program Termination: exit syscall
(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          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x0          0
r12         0x0          0
sp          0x7efff750    0x7efff750
lr          0x0          0
pc          0x10070       0x10070 <_start+28>
cpsr       0x10         16
fpscr      0x0          0

```

Picture 2: Debugging of Arithmetic1

### Part 1: First Program Explanation

The first part of this assignment was program was made simply just to add and subtract ARM code as an example. The command “as” assembles the code that you wrote, while the command “ld” is the linker and creates the executable of the file. Adding “-g” in the code assembles the code and also allows for the GDB debugger to be used. The GDB Debugger serves as a way to check where specifically your code has bugs, but in our case we want to see what values were added to each register. We used the command “b <number>” to see specifically when we wanted to see the state of each register and “info registers” to see the registers themselves. In our example, we used “b 11” to see what our registers would look like post-operations and “info registers” to execute, which in our example register r2 has 8 because  $5 + 4 - 1 = 8$ .

I did not have difficulties running the original “first” program. I believe that running “./first” doesn’t have any output because we are only adding values to each register, and the output itself does not show the register. I was able to run the debugger seamlessly without any error and register r1 has the correct number in the entry ( $5 - 1 + 4 = 8$ ). When I did “list”, I did not press enter so it only shows 10 lines.

### Part 2: Arithmetic Program Explanation

The second part of this assignment we were supposed to write a program that had the arithmetic sequence  $A = (A + B) - (C * D)$  where  $A = 10$ ,  $B = 11$ ,  $C = 7$ , and  $D = 2$ . I assembled the code using “as”, linked the code and created the .exe using “ld” and re-assembled the code for debugging in GDB uses using “as -g” vs “as”. I ran the debugger using “gdb arithmetic1” and set the breakpoint to be at “b 15” because that was after I had completed my operations. I was able to see that my registers were set correct because the arithmetic was right.

While writing the code, I did experience some difficulties but that was due to user error. When I originally compiled this code, I had sub “r1, r4” but that would be incorrect since the value of  $(C * D)$  is stored in C (r3 in this case), so the correct line would be “sub r1, r3”, which is seen in the screenshot.

That error caused me to run into issues while I was trying to debug it, because if you just reassemble the line “as -o Assembly1.o Assembly1.s”, it will not affect the line in the debugger, so I had to reassemble “as -g -o Assembly1.o Assembly1.s” to get the correct output in r1, which is 7.

## Appendix

Slack: [https://app.slack.com/client/TTK19C222/CTGQG7H7A/user\\_profile/UTGB5U71S](https://app.slack.com/client/TTK19C222/CTGQG7H7A/user_profile/UTGB5U71S)

GitHub: <https://github.com/Rsandy2/8-Bit>

Presentation on YouTube:

<https://www.youtube.com/watch?v=O8H1UMMoN3w&feature=youtu.be>

Screenshot of Introductions on Slack:

