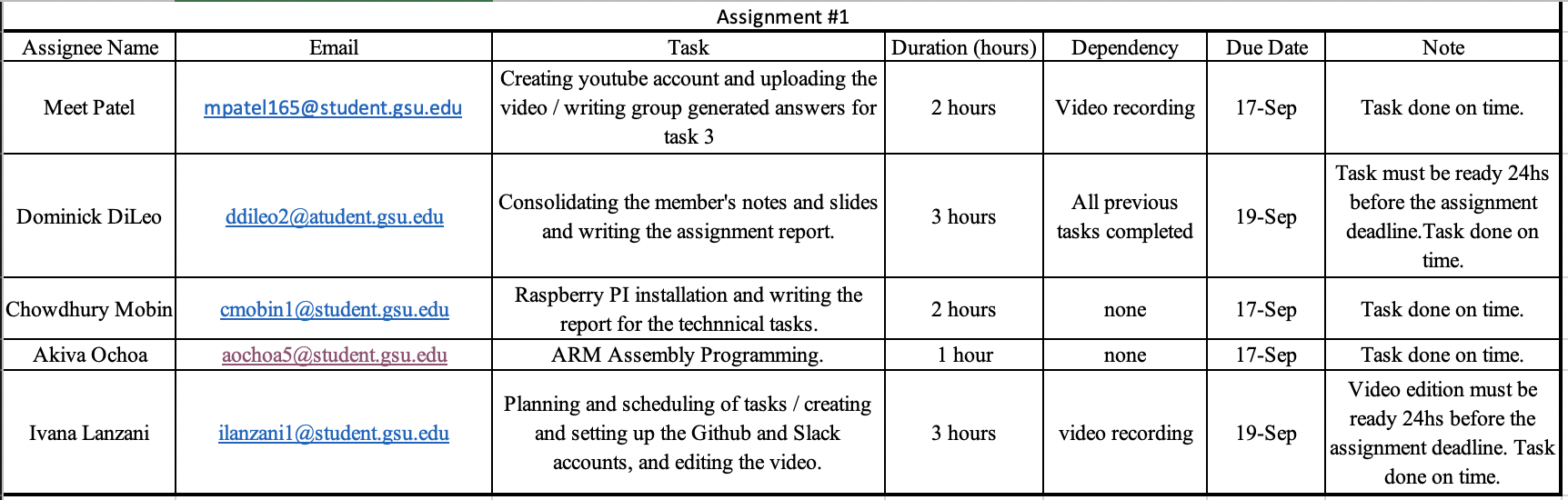
**Developing Soft and Parallel Programming Skills Using Project-Based Learning**

**Group Name: Quinary**

**Fall 2019**

**Akiva Ochoa, Chowdhury Mobin, Ivana Lanzani, Meet Patel, Dominick DiLeo**

**Planning and Scheduling**



**Teamwork Basics**

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

In order to achieve good performance and a high level of satisfaction in the team, the members should spend time to get to know each other’s strengths and preferences, agree on a set of ground rules to maintain a good working environment, use a facilitator to organize the tasks and achieve better productivity, as well as to help resolve conflicts that may arise between the members, and finally, it is necessary to keep good communication to ensure everybody’s satisfaction.

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

**Work Norms**: Work will be distributed depending on how complex and time consuming the task is. The deadlines will be set by coordinator and approved by all the team members at the beginning of the assignment. If someone doesn’t follow through on his/her commitment, then the person will be given a warning; if this behavior continues or the task is not completed within the agreed time, then another group member will take over that task and the person that did not do their part will receive a grade proportional to the amount of work he/she provided. Work will be reviewed by all members before the final submission of the assignment to guarantee it meets everybody’s expectations. If team members were to have different opinions or work habits, then those will be discussed between all the team members. The style that benefits the team the most and that the majority of members agree on will be the one that everyone will follow.

**Facilitator Norms**: Yes, we will choose a facilitator. We have decided to make the facilitator the same as the coordinator so that everyone in the group gets a chance to build up skills like keeping a task moving along smoothly between many people.

**Communication Norms**: For communication, we will all rely on the GSU email platform and a GroupMe group to set up meeting times and let the other team members know of changes.

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

**Too Quiet**: If there were to be a quiet, shy or uninvolved person in the group, the rest of the members will make efforts to include him or her in the conversation and help them understand how important their contributions are to the project.

**Complains**: If a member complains a lot, the rest of the members will set aside time to listen to the complaints and try to make that person feel more comfortable, as well as make modifications in the case that the complaint was legitimate.

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

If the team is having trouble reaching a consensus, the members would vote to select the more popular action plans. The team will work through a list of pros and cons and if it is possible, we will try to combine the plans.

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 the case someone was inclined to make rushed decisions, the other team members can control the situation by questioning if the team is ready to move on to the next task. We will see if there are any things we need to review still and if anyone needs a little extra time understanding something.

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 most people of the team aim for an “A” but someone is not willing to work hard enough to achieve it because they consider a lower grade to suffice, then the team will talk to the person to try to convince him or her to perform according to the standards that the majority of the team agreed on, otherwise his or her work will be revised and improved if necessary by another member, and that person will lose a percentage of the grade according to the extra amount of work that was needed to properly complete his/her task.

**Raspberry PI Installation and ARM Assembly Programming**

Raspberry pi is a low-cost minicomputer which was developed in the United Kingdom. It is a single board Linux based computer which can perform programs written in many languages such as C, C++, Python, JAVA etc.

Properly installing the pi was essential for this project. The Pi already had an operating system when we plugged it in and so we did not need to install anything. If we were going to have to do the installation first, we would download the zipped file Raspbian Stretch with Desktop and image flasher app called Etcher. After which we would use Etcher to flash the Raspbian image to our SD card. To physically set up the PI we went to the second floor of the library. We unplugged the usb mouse and keyboard and plugged it into the PI. Then we plugged in the hdmi into the display and attached the power cable.

See appendix picture 1

Launching the Raspberry pi for the first time, six icons were present on the top bar. The 4th icon is the terminal, which will be used to write and run our program on Raspberry Pi.

To open the text editor and write a program, we typed the command “nano first.s”. Which opened a new file named first.s using nano editor. The following commands “.selection .data”, “.selection .text” and “.globl \_start” are written for the linker. The command “\_start:” , tells the linker that this is the entry point of the program.

When we write “ mov r1, #5”, it loads the register r1 with the value of 5. The next line of command “sub r1, r1, #1”, subtracts 1 from the value of register r1, which was 5 and saves the value after the subtraction in register r1. “add r1, r1, #4”, this adds 4 to the value of the register r1 and stores the value to r1. “mov r7, #1” loads the register r7 with the value of 1 and the next line of command “svc #0”, which directs the processor to pass control of the computer to the supervisor program of the operating system. The command, “. end” marks the end of the program. Then we save the file by pressing “control w” on our keyboard.

To assemble the file the following command should be typed “as -o first.o first.s”. Here it will create an object file for our file “first.s”. Now linked our file to an executable by typing “ld -o first first.o”. After we tried to run the program by typing the “./first” command, we couldn’t see any output. To see the contents of the registers and memory we must GDB for debugging our program. To do so we added “-g” to the assembler command which now stands “as -g -o first.o first.s”. The next line for linker command will remain the same. To launch the debugger, we typed gdb followed by our executable file name which is “gdb first”. Typing “gdb list” gives us the first 10 lines of the source code. To see the next ten lines, we just must press Enter key on the keyboard.

To examine the registers, we must stop the program in its execution by setting a breakpoint. We can set a breakpoint at line 11 by typing “b 11” then pressing the Enter key. To hit the breakpoint, we must start the program execution from the beginning by typing “run”. CPU registers can be examined to see the result by typing the command “info registers”, which will show us the register name, hexadecimal contents and the contents in decimal.

See appendix picture 2

To take the screenshot we opened another terminal window and typed “scrot”. Here we can see that the value of r1 is 8 which has changed as we subtracted 1 from 5, the initial value, then added 4 to it. Register r7 has stored value 1 on it from the command “mov r1, #1”.

For part 2, we have to calculate the expression , and the values given are A=10, B=11, C=7 and D=2. To start the new program, we start the terminal and type the following command “nano arithmetic1.s”. It will start the nano editor and open our file, named arithmetic1.s. The following commands are written for the linker, “ .selection .data”, “.selection .text”, “.globl \_start”, all in separate lines. To specify an entry point of the program we typed the command “\_start”.

The next step of our program is to load values to the registers. To do so, we type the following commands:

mov r1, #10 @r1 is A

mov r2, #11 @r2 is B

mov r3, #7 @r3 is C

mov r4, #2 @r4 id D

These commands just loaded the values 10, 11, 7, 2 to r1, r2, r3 and r4 registers. After loading the value, we start adding, subtracting and multiplying by typing the following commands on terminal:

add r1, r2

mul r3, r4

sub r1, r3

The first command adds the value loaded on r1 and r2, then stores the result on r1 register. The second line of command multiplies the values loaded on r3 and r4 and loads the result on register r3. Third line of command subtracts the value loaded on r3 from r1, this command will use will the values, register r1 and r3 had after the addition and multiplication. Then we write the supervisor call command “svc #0” and end our program with “.end” command on the terminal.

See appendix picture 3

To run the program, we assemble it by typing “as -g -o arithmetic1.o arithmetic1.s”, which creates an object file. To get an executable file we type the linker command “ld -o arithmetic1 arithmetic1.o”. The command “gdb arithmetic1” will launch the GNU debugger. After we set a breakpoint “b (line number)” the program is ready to run by using the command “run”. As the program has been executed, we can now see the register values by writing “info registers” command on the terminal.

See appendix picture 4

The register r1 which was our A holds the value 7, r2 or B is 11, r3 or C is 14 and r4 or D is 2. Looking at registers we can confirm that we have successfully calculated the expression with the given values. , which is the value of register r1.

**Appendix**

**Links:**

Slack: <https://app.slack.com/client/TN3HREW9X/CN3HRGB33>

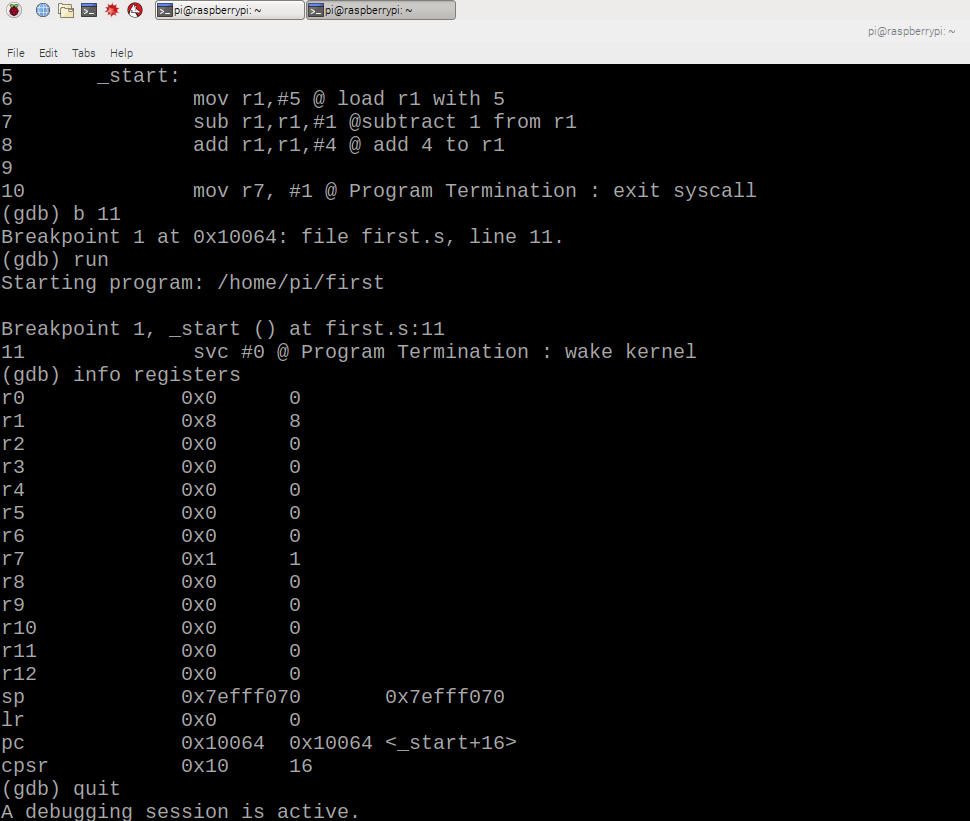
Github: <https://github.com/Quinary-GSU/Assignment1/projects/1>

**Pictures:**

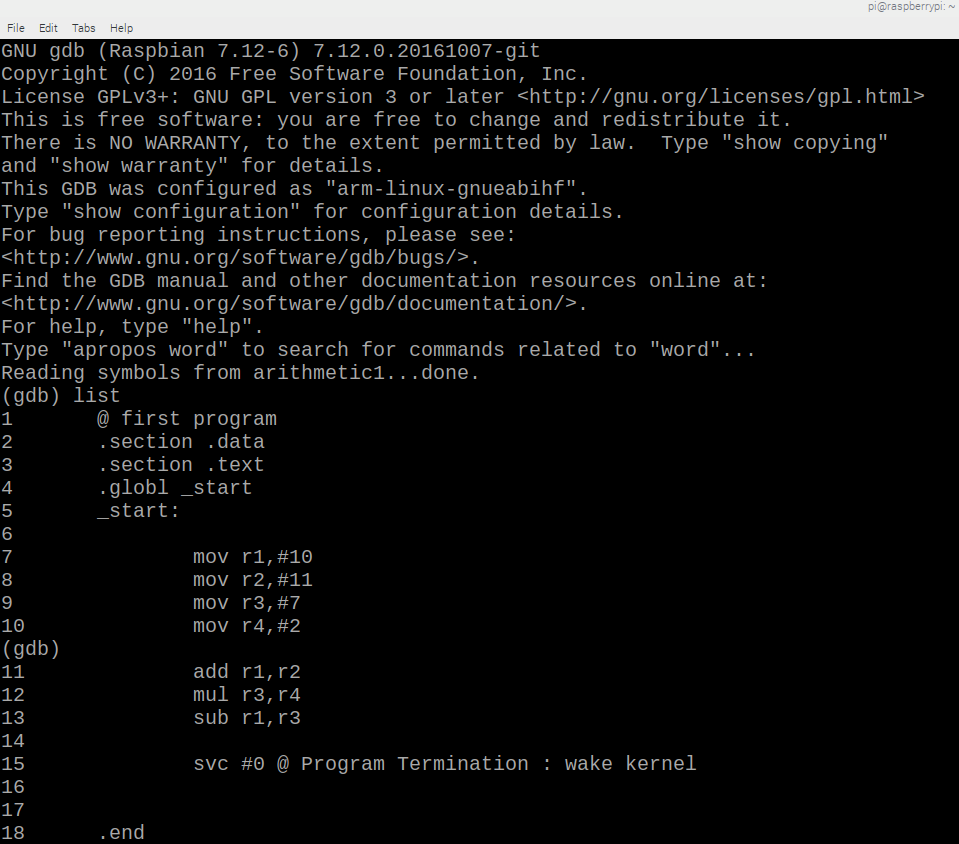
**1)**

****

**2)**



**3)**



**4)**

