

Developing Soft and Parallel Programming Skills Using Project-Based Learning

Spring-2019 **JupitorInk**

Nhi Ong, Alejandro Herbener, Alec Buchinski, Johnathan Moore

Table of Contents

I.	Project Description.....	2
	1.1 Purpose of the Project.....	2
II.	Introduction to the Team.....	2
	2.1 Members Roles and Assignments.....	2
III.	Planning and Scheduling.....	2-4
	3.1 Role as Coordinator.....	3
	3.2 Additional Tasks.....	3-4
IV.	Teamwork Basics.....	4-7
	4.1 Teamwork Basics Responses.....	4
	4.2 Teamwork Basics Compiled Report.....	4-7
	4.2.a Work Norms.....	4-5
	4.2.b Facilitator Norms.....	5-6
	4.2.c Communication Norms.....	6-7
V.	Parallel Programming Skills.....	7-12
	5.1 Foundation.....	7-8
	5.2 Parallel Programming Basics Report.....	9-12
VI.	ARM Assembly Programming.....	12-14
VII.	Appendix.....	15

I. Project Description

1.1 Purpose of the Project

The purpose of this project is to 1) become familiar with unified, team-based project management tools and communication applications in order to develop soft skills, 2) identify Raspberry Pi 3+ components and architecture; explain basic multicore architectures and programming using OpenMP and C language 3) gain more practice with ARM assembly programming (Mussa, Assignment 2 Worksheet).

II. Introduction to the Team

2.1 Members' Roles and Assignments

Coordinator: Alejandro Herbener

He is the coordinator for this assignment. In addition to his responsibilities as a coordinator, she was also responsible for tasks one, two, and six. He created a new assignment task chart, sent an invitation to our Slack workspace to the TA, created a new GitHub repository and projects page, and edited our video and uploaded it to the YouTube channel.

Team Member: Alec Buchinski

This team member is responsible for task three. Alec answered the questions that for the Parallel Programming Foundation and completed the programming portion of this task. He also took screenshots and created a written report of his findings.

Team member: Nhi Ong

This team member is responsible for task five. I created this report after compiling each member's work from their corresponding tasks.

Team Member: Johnathan Moore

This team member is responsible for task four. Johnathan completed the ARM Assembly Programming for this task. He took screenshots and included them in his write up for this task.

III. Planning and Scheduling

3.1 Role as Coordinator

As the coordinator, Alejandro was responsible for assigning us with our tasks, updating our assignment table, creating a new repository for the assignment, and adding the TA to our Slack workspace. He first contacted us to discuss our preferences for our roles in this assignment, which we each quickly selected and promptly began working on our tasks. He updated the assignment task table soon thereafter and notified us of our responsibilities, dependencies, and due date. (see Figure 1.1)

	A	B	C	D	E	F	G	H	I
	Assignee Name	Email	Task	Duration (hours)	Dependency	Due date			
1	Alejandro Herbener	aherbener2@student.gsu.edu	Task 1: Planning and Scheduling; Task 2: Collaboration	1~	Group members' input on preferred tasks	2/16/2019			
3	Alec Buchinski	abuchinski1@student.gsu.edu	Task 3: Parallel Programming skills	1-2	None	2/22/2019			
4	Johnathon Moore	jmoore180@student.gsu.edu	Task 4: ARM Assembly Programming	1-2	Completion of task 3 to use the Raspberry Pi	2/22/2019			
5	Nhi Ong	nong1@student.gsu.edu	Task 5: Report	0-1	Completion of all other tasks	2/22/2019			
6									
7									
8									
9									
10									
11									
12									

Figure 1.1 – Task One, Assignment Chart

3.2 Additional Tasks

Alejandro was responsible for creating a new repository for this assignment with a readme page. He added a projects page which details our tasks and allows for any member to update their status on their role. (see Figure 1.2) In addition to this, he also edited our video and uploaded it to YouTube. (see Appendix A, for link to [video](#))

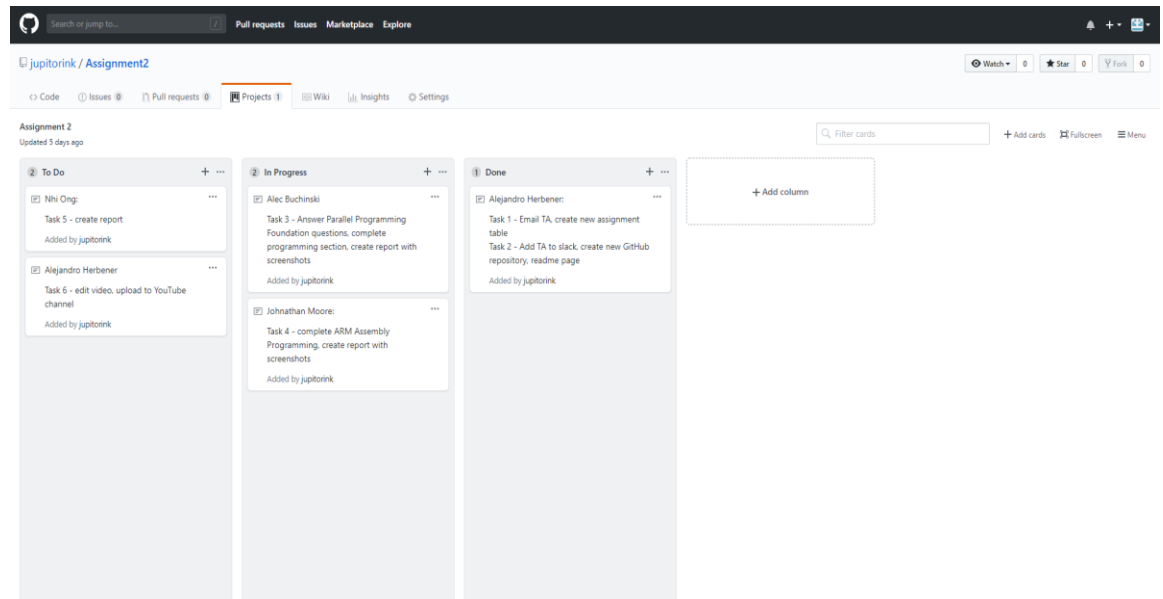


Figure 1.2 – Task Two, GitHub Projects Page Screenshot

IV. Teamwork Basics

4.1 Teamwork Basics Responses

Due to issues we encountered with finding a proper time to meet in order to complete the list of questions we had to get creative. Alejandro decided that it would be most convenient for each member to read over the teamwork building document individually and promptly submit their responses to the provided questions on the Learning Teamwork Basics section of the assignment. He created a poll for us to enter our responses in order to compile them all into a neat document. The earlier issues with scheduling time restricted our options when it comes to meeting up for the “group” sections of the assignment, therefore Alejandro’s idea gave us far more breathing room for other tasks such as the video.

4.2 Teamwork Basics Compiled Report

4.2.a Work Norms:

What to do to get the task accomplished and the team member’s satisfaction high?

We all agree that the key to successful teamwork and project outcome is having good communication within the team and working through each task

together. We must also set ground rules that each member should follow and consulting our designated facilitator should we be unable to reach a compromise when an issue arises.

How will work be distributed?

The work will be distributed based on a combination of what each team members' preference and strengths as well as the facilitator's decision.

Who will set deadlines?

The facilitator is responsible for setting deadlines.

What happens if someone doesn't follow through on his/her commitment (for example, misses a deadline)?

The task will be transferred over to a member(s) who has the time and resources to complete the task. The professor will be notified in order for the member(s) to receive the appropriate credit.

How will the work be reviewed?

Tasks are uploaded to the appropriate repository on GitHub and Google Drive, allowing all members to check each other's work to ensure it is of sufficient quality. Every member has access to the GitHub and Google Drive, therefore having access to each other's work.

What happens if people have different opinions about the quality of the work?

A compromise is arranged so that work on the project can progress.

What happens if people have different work habits (e.g., some people like to get assignments done right away; others work better with the pressure of a deadline)?

The facilitator sets deadlines convenient for each member, so that everyone can work at their own pace knowing when their work needs to be done.

4.2.b Facilitator Norms:

Will you use a facilitator? How will the facilitator be chosen?

Yes. The first one was chosen by the professor.

Will you rotate the position? What are the responsibilities of the facilitator?

Yes. The facilitator must ensure that each member completes their task in a timely and appropriate manner.

4.3.c Communication Norms:

When should communication takes place and through what medium (e.g., do some people prefer to communicate through e-mail while others would rather talk on the phone)?

Our group has decided to use a medium that everybody has easy access to and would check frequently enough to be an effective medium of communication.

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

If a person is being too quiet, they could, for example, not be reaching out to other members of the group, which would impede the productivity of the group. Drawing this person out would help them to become more comfortable with the group and not be discouraged from communicating. If a person is complaining, they would probably be complaining about the assignment of tasks, to which the proper response would be to listen to their response, resolve the situation if their argument is legitimate, or if it's not legitimate, explain to them why.

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

Try and find a compromise that everybody can be happy with.

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?

Ask them to slow down and work with the pace of the team and encourage them to use the extra time they have to double check their work.

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 that person can’t be convinced that putting in full effort is acceptable, then we let the professor know so that that person gets a grade corresponding to the effort they put in.

V. Parallel Programming Skills

5.1 Foundation

Identifying the components on the raspberry PI B+

The most important component of the Pi is the CPU (which also happens to be the RAM in this case) which is the largest square chip located on the left half of the board underneath the heat sink. Along the sides there are connectors for USB (4 ports) HDMI, power, camera and display. There is also an Ethernet port next to the USB’s that has a corresponding controller nearby.

How many cores does the Raspberry Pi B+ CPU have?

As the CPU is labeled as a Quad-Core CPU, it is safe to assume that there are four cores.

List three main differences between X86 (CISC) and ARM Raspberry PI (RISC), Justify your answers and use your own words (do not copy and paste).

- 1) ARM’s RISC instruction set is far more basic and can only access memory to Load/Store. This means that ARM cannot use instructions other than Load/Store on memory directly leading to moving memory to registers and back to memory for simple changes which can feel redundant to someone writing the code.
- 2) ARM has the potential to be faster and more efficient, while writing X86 is much simpler due to the removal of some redundancy found in ARM’s simple instructions.
- 3) X86 has fewer total registers due to the tradeoff for complex instructions in dealing with memory, which may affect the scope of some larger codes.

What is the difference between sequential and parallel computation and identify the practical significance of each?

In sequential computation, the code is broken down into parts and is executed one at a time through a single processor. Parallel computation sends its parts of code and solves them simultaneously on different processors. Parallel is important as it can execute code more efficiently than sequential, while Sequential’s significance

lies with ease of access especially when coding for hardware that may only have one core.

Identify the basic form of data and task parallelism in computational problems.

- 1) Data Parallelism refers to problems that require the same computation to be used on multiple data items. Imagine how many computations will need to be done if a large group of data elements had to all be individually modified. The potential parallelism could be huge.
- 2) Task Parallelism refers to problems where the focus of the parallelism is on the functions and not the data itself.

Explain the differences between processes and threads?

Threads are used for “lighter” tasks that allows a process to be broken up into independent parts, as well as the fact that they share the common memory of the process they are a part of, while processes are abstractions of a program that run with separate memory and are used primarily for heavier tasks. Processes can only be done one at a time with a single-core CPU

What is OpenMP and what is OpenMP pragmas?

OpenMP is an API which is used for parallel programming within a multi-core CPU. The OpenMP pragmas are compiler directives that will cause the compiler to generate threaded code in order to carry the work out using parallelism.

What applications benefit from multi-core (list four)?

Compilers
Database Servers
Multimedia Applications
Web Servers

Why Multicore? (why not single core, list four)

- 1) It is used to bypass clock frequency limits of single-core
- 2) Several newer applications are multithreaded which essentially requires multi-cores.
- 3) General computer architecture trends are shifting toward parallelism.
- 4) Deeply pipelined circuits (Problems with more sub-problems) would be unnecessarily complicated with single-core

5.2 Parallel Programming Basics

2/18 - Alec handed off the Raspberry Pi to me; however, due to unforeseen circumstances, I was unable to begin working on the assignment until Wednesday night.

2/20 - I began working with the Pi.

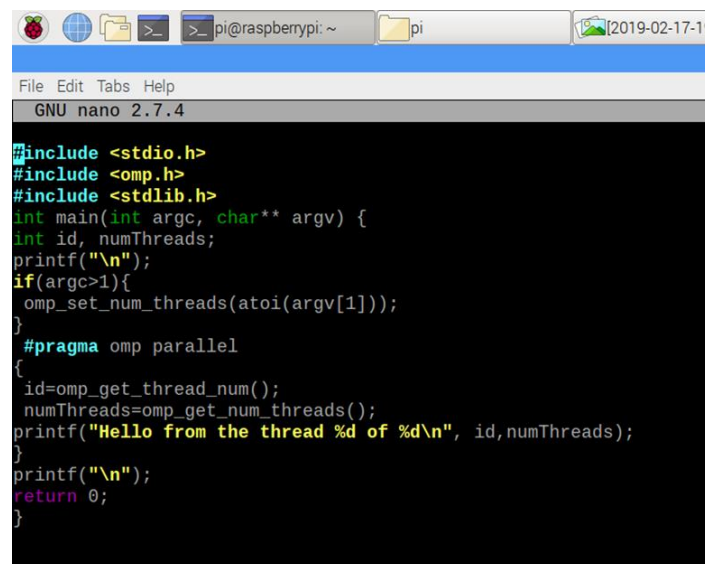
The main learning outcome for this assignment is Parallel Programming, which meant that I had to write a code that utilizes multiple cores of the CPU after finishing the initial “Foundation” section and answering the questions. Afterwards, I began working with the Pi directly.

I ran into my first obstacle with the Pi – I had no prior experience. I spent 30 minutes playing around with it to familiarize myself with the device.

The instructions were fairly simple for the assignment. After reading the basic tutorial for programming, I analyzed the code provided in an attempt to understand OpenMP and how it allows programs to easily run blocks of code concurrently.

This is my process through the programming section step by step:

2.2 Copying the code was giving me issues so I typed it in manually and found a missing “>” in the given code. I created `spmd2` and made the executable. Below is the completed code:



```

GNU nano 2.7.4
#include <stdio.h>
#include <omp.h>
#include <stdlib.h>
int main(int argc, char** argv) {
    int id, numThreads;
    printf("\n");
    if(argc>1){
        omp_set_num_threads(atoi(argv[1]));
    }
    #pragma omp parallel
    {
        id=omp_get_thread_num();
        numThreads=omp_get_num_threads();
        printf("Hello from the thread %d of %d\n", id,numThreads);
    }
    printf("\n");
    return 0;
}

```

Figure 5.1 – Task Three, Raspberry Pi Screenshot A

I then ran the executable to get an output. I ran it a couple of times with different values.

```

pi@raspberrypi:~ $ ./spmd2 4
Hello from the thread 3 of 4
Hello from the thread 2 of 4
Hello from the thread 2 of 4
Hello from the thread 2 of 4

pi@raspberrypi:~ $ ./spmd2 4
Hello from the thread 0 of 4
Hello from the thread 3 of 4
Hello from the thread 2 of 4
Hello from the thread 0 of 4

pi@raspberrypi:~ $ ./spmd2 3
Hello from the thread 0 of 3
Hello from the thread 2 of 3
Hello from the thread 0 of 3

pi@raspberrypi:~ $ ./spmd2 16
Hello from the thread 1 of 16
Hello from the thread 13 of 16
Hello from the thread 6 of 16
Hello from the thread 14 of 16
Hello from the thread 1 of 16
Hello from the thread 1 of 16
Hello from the thread 9 of 16
Hello from the thread 8 of 16
Hello from the thread 11 of 16
Hello from the thread 4 of 16
Hello from the thread 10 of 16
Hello from the thread 12 of 16
Hello from the thread 15 of 16
Hello from the thread 5 of 16
Hello from the thread 7 of 16
Hello from the thread 0 of 16

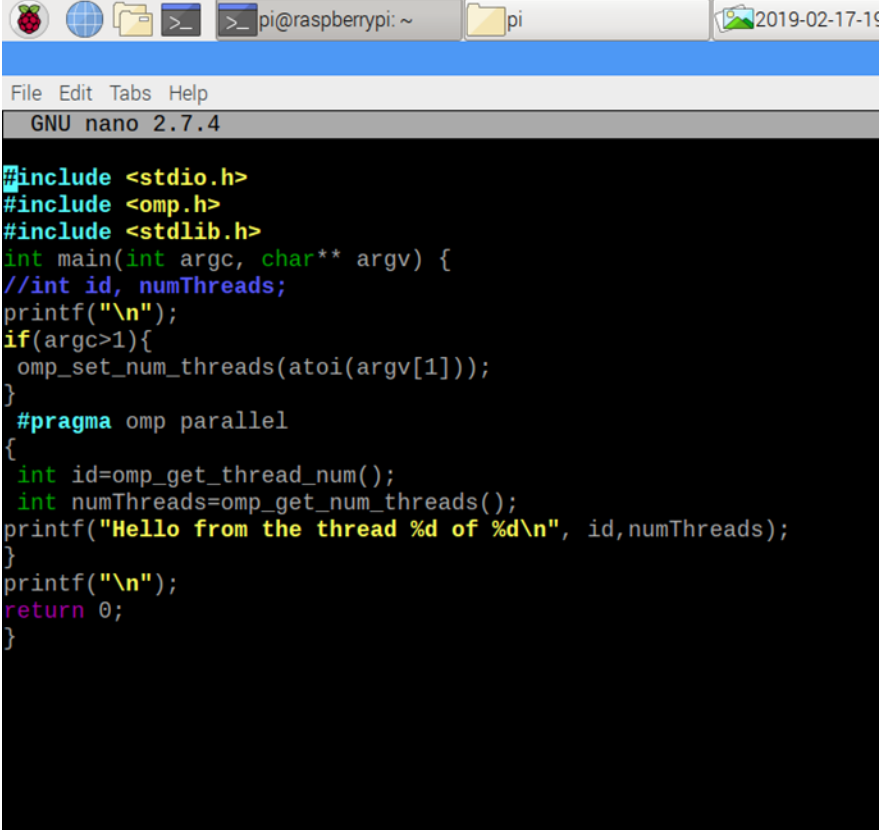
```

Figure 5.2 – Task Three, Raspberry Pi Screenshot B

The number after the command to run `spmd2` represents how many instances of the print block will be run (how many threads to fork) and the desired goal is to have no repeating threads. As you can see, however, there were issues to look into causing repeats.

2.3 The order being out of place is not an issue in terms of this assignment. What is an issue, however, is that there are repeating numbers within the threads. This seems to be an issue with the variables being initialized outside of the block that will be forked. The issue was surprisingly easy to understand after reading the corresponding text within the assignment.

2.4 After commenting out the declaration line using `*/` and editing the initialization lines to have the declaration within them, the code looked like this:



```

File Edit Tabs Help
GNU nano 2.7.4

#include <stdio.h>
#include <omp.h>
#include <stdlib.h>
int main(int argc, char** argv) {
//int id, numThreads;
printf("\n");
if(argc>1){
    omp_set_num_threads(atoi(argv[1]));
}
#pragma omp parallel
{
    int id=omp_get_thread_num();
    int numThreads=omp_get_num_threads();
printf("Hello from the thread %d of %d\n", id,numThreads);
}
printf("\n");
return 0;
}

```

Figure 5.2 – Task Three, Raspberry Pi Screenshot C

As you can see, only lines 5, 12, and 13 were changed while moving the declarations. The result of these changes is shown below:

```

pi@raspberrypi:~ $ nano spmd2.c
pi@raspberrypi:~ $ gcc spmd2.c -o spmd2 -fopenmp
pi@raspberrypi:~ $ ./spmd2 4
bash: ./spmd2: No such file or directory
pi@raspberrypi:~ $ ./spmd2 4

Hello from the thread 1 of 4
Hello from the thread 3 of 4
Hello from the thread 0 of 4
Hello from the thread 2 of 4

pi@raspberrypi:~ $ ./spmd2 4

Hello from the thread 0 of 4
Hello from the thread 1 of 4
Hello from the thread 2 of 4
Hello from the thread 3 of 4

pi@raspberrypi:~ $ ./spmd2 6

Hello from the thread 1 of 6
Hello from the thread 0 of 6
Hello from the thread 2 of 6
Hello from the thread 3 of 6
Hello from the thread 5 of 6
Hello from the thread 4 of 6

```

Figure 5.2 – Task Four, Raspberry Pi Screenshot D

The code worked and showed no repeats no matter what size I chose! The fix was successful. With the code corrected and the desired output, the task is complete!

2/21 I went back into the Pi to retake screenshots for a higher font size.

VI. ARM Assembly Programming

In assignment four, I started by copying the listed program, assembling and linking it. When I ran the program, there was no output as nothing was printed out, and register/memory change should not be visible outside of a debugger. When following the `x/3xw` command for the known register `0x1008`, three addresses are shown in hexadecimal.

This is the results of our changes on the register. This location is where the breakpoint was. It shows the three addresses visible at that time. (see Figure 6.1)

```

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) b 15
Breakpoint 1 at 0x1008c: file second.s, line 15.
(gdb) run
Starting program: /home/pi/second

Breakpoint 1, _start () at second.s:15
15      str r1, [r2]    @store r1 into memory c
(gdb) stepi
17      mov r7, #1      @Program Termination: exit syscall
(gdb) stepi
18      svc #0          @Program Termination: wake kernel
(gdb) info registers
r0      0x00000000
r1      0x00000007
r2      0x200000ac     131244
r3      0x00000000
r4      0x00000000
r5      0x00000000
r6      0x00000000
r7      0x10000001
r8      0x00000000
r9      0x00000000
r10     0x00000000
r11     0x00000000
r12     0x00000000
sp      0x7efff060     0x7efff060
lr      0x00000000
pc      0x10094      0x10094 <_start+32>
cpsr    0x10000000
(gdb) p&a
$1 = (<data variable, no debug info> *) 0x200a4
(gdb) x/3xw 0x200a4
0x200a4: 0x00000002 0x00000005 0x00000007

```

```

File Edit Search Options Help
second program c = a + b
.section .data
a: .word 2 @32-bit variable in memory
b: .word 5 @32-bit variable in memory
c: .word 0 @32-bit variable in memory
.section .text
.globl _start
_start:
    ldr r1, =a @load the memory address of a into r1
    ldr r1, [r1] @load the value a into r1
    ldr r2, =b @load the memory address of b into r2
    ldr r2, [r2] @load the value b into r2
    add r1, r1, r2 @add r1 to r2 and store into r1
    ldr r2, =c @load the memory address of c into r2
    str r1, [r2] @store r1 into memory c
    mov r7, #1 @Program Termination: exit syscall
    svc #0 @Program Termination: wake kernel
.end

```

Figure 6.1 – Task Four, Raspberry Pi Screenshot A

For part two, I created a program for the listed equation, Register = val2 + 9 + val3 – val 1. I created three words named val1 val2 and val3 and then loaded them into registers r1 r2 and r3. (see Figure 6.2)

```

File Edit Search Options Help
5      val3: .word 16 @32-bit variable val3 in meory
6      .section .text
7      .globl _start
8      _start:
9          ldr r1, =val1 @load the memory address of val1 into r1
10         ldr r1, [r1] @load the value of val1 into r1
(gdb) b 17
Breakpoint 1 at 0x10094: file arithmetic2.s, line 17.
(gdb) run
Starting program: /home/pi/arithmetic2

Breakpoint 1, _start () at arithmetic2.s:17
17      sub r3, r3, r1 @subtract r1(val1) from r3(val3) and store in val3
(gdb) stepi
18      add r1, r2, r3 @add r2(val2+9) and r3(val3-val1) and store in r1
(gdb) stepi
20      mov r7, #1 @Program Termination: exit syscall
(gdb) stepi
21      svc #0 @Program Termination: wake kernel
(gdb) info registers
r0      0x00000000
r1      0x10000030
r2      0x10000020
r3      0xa0000010
r4      0x90000009
r5      0x00000000
r6      0x00000000
r7      0x10000001
r8      0x00000000
r9      0x00000000
r10     0x00000000
r11     0x00000000
r12     0x00000000
sp      0x7efff060     0x7efff060
lr      0x00000000
pc      0x100a0      0x100a0 <_start+44>
cpsr    0x10000000
(gdb)

```

```

File Edit Search Options Help
SecondProjectProgram Register = val2 + 9 + val3 - val1
.section .data
val1: .word 6 @32-bit variable val1 in memory
val2: .word 11 @32-bit variable val2 in memory
val3: .word 16 @32-bit variable val3 in memory
.section .text
.globl _start
_start:
    ldr r1, =val1 @load the memory address of val1 into r1
    ldr r1, [r1] @load the value of val1 into r1
    ldr r2, =val2 @load the memory address of val2 into r2
    ldr r2, [r2] @load the value of val2 into r2
    ldr r3, =val3 @load the memory address of val3 into r3
    ldr r3, [r3] @load the value of val3 into r3
    mov r4, #9 @load the value of 9 into r4
    add r2, r2, r4 @add r2(val2) and r4(9) and store in r2
    sub r3, r3, r1 @subtract r1(val1) from r3(val3) and store in val3
    add r1, r2, r3 @add r2(val2+9) and r3(val3-val1) and store in r1
    mov r7, #1 @Program Termination: exit syscall
    svc #0 @Program Termination: wake kernel
.end

```

Figure 6.2 – Task Four, Raspberry Pi Screenshot B

I moved 9 into r4, did the math, and calculated the result in register r1. Checking the registers, the desired value of 30 was found in the correct register and the operations worked as intended. (see Figure 6.3)

```

File Edit Tabs Help
pi@raspberrypi: ~
2019-02-17-200249... 2019-02-17-200816... 2019-02-17-195927... arithmetic2.s
pi@raspberrypi: ~

9      ldr r1, =val1 @load the memory address of val1 into r1
10     ldr r1, [r1]  @load the value of val1 into r1
(gdb) b 17
Breakpoint 1 at 0x10094: file arithmetic2.s, line 17.
(gdb) run
Starting program: /home/pi/arithmetic2

Breakpoint 1, _start () at arithmetic2.s:17
17     sub r3, r3, r1 @subtract r1(val1) from r3(val3) and store in val3
(gdb) stepi
18     add r1, r2, r3 @add r2(val2+9) and r3(val3-val1) and store in r1
(gdb) stepi
20     mov r7, #1    @Program Termination: exit syscall
(gdb) stepi
21     svc #0        @Program Termination: wake kernel
(gdb) info registers
r0      0x00 0
r1      0x1e 30
r2      0x14 20
r3      0xa 10
r4      0x9 9
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      0x7efff050 0x7efff050
lr      0x0 0
pc      0x100a0 0x100a0 <_start+44>
cpsr    0x10 16
(gdb) p&val1
$1 = (<data variable, no debug info> *) 0x200b0
(gdb) x/3xw 0x200b0
0x200b0: 0x00000006 0x0000000b 0x00000010
(gdb)

File Edit Search Options Help
arithmetic2.s
B SecondProjectProgram Register = val2 + 9 + val3 - val1
.section .data
val1: .word 6 @32-bit variable val1 in memory
val2: .word 11 @32-bit variable val2 in memory
val3: .word 16 @32-bit variable val3 in memory
.section .text
.globl _start
_start:
    ldr r1, =val1 @load the memory address of val1 into r1
    ldr r1, [r1] @load the value of val1 into r1
    ldr r2, =val2 @load the memory address of val2 into r2
    ldr r2, [r2] @load the value of val2 into r2
    ldr r3, =val3 @load the memory address of val3 into r3
    ldr r3, [r3] @load the value of val3 into r3
    mov r4, #9 @load the value of 9 into r4
    add r2, r2, r4 @add r2(val2) and r4(9) and store in r2
    sub r3, r3, r1 @subtract r1(val1) from r3(val3) and store in val3
    add r1, r2, r3 @add r2(val2+9) and r3(val3-val1) and store in r1
    mov r7, #1 @Program Termination: exit syscall
    svc #0 @Program Termination: wake kernel
.end
  
```

Figure 6.3 – Task Four, Raspberry Pi Screenshot C

VII. Appendix

A. Link to Slack, GitHub, YouTube Channel, YouTube Video

Slack:

<https://csc-3210.slack.com>

GitHub:

<https://github.com/jupitorink>

GitHub Assignment One Repository:

<https://github.com/jupitorink/Assignment2>

YouTube Channel:

<https://www.youtube.com/channel/UChTgBqHzG9G-Mvv-q7uK96g>

YouTube Video:

<https://youtu.be/-NfXb5pMrBo>

B. Task One Screenshot

Task Assignment Table:

<https://github.com/jupitorink/Assignment2/blob/master/Task%201.png>

C. Task Two Screenshot

GitHub Projects Page:

<https://github.com/jupitorink/Assignment2/blob/master/task2.png>

D. Task Three Parallel Programming Report, Screenshots

Parallel Programming Foundation

<https://github.com/jupitorink/Assignment2/blob/master/Task%203%20Foundation%20Questions.docx>

Parallel Programming Report, Screenshots

<https://github.com/jupitorink/Assignment2/blob/master/Task%203%20Write-Up.docx>

E. Task Four ARM Assembly Programming Report, Screenshots

ARM Assembly Programming Report

https://github.com/jupitorink/Assignment2/blob/master/Assignment2_Task4_AlecBuchinski.docx

Raspberry Pi Screenshot A

<https://github.com/jupitorink/Assignment2/blob/master/Task4-program1-Variables-register.png>

Raspberry Pi Screenshot B

<https://github.com/jupitorink/Assignment2/blob/master/Task4-program2-registers.png>

Raspberry Pi Screenshot C

<https://github.com/jupitorink/Assignment2/blob/master/Task4-program2-variables.png>