Developing Soft and Parallel Programming Skills Using Project-Based Learning

Spring 2020

Gitter-geeks

Hyoungjun Lee

Michael Knight – Team Coordinator

Tek Acharya

Vedansi Parsana

Zoe Kosmicki

Planning and Scheduling

As Team Coordinator I was meet with challenges but overall with a great team my challenges weren't that difficult. We kept the same time as we had before to meet up to cover anything that needed to be covered and to record our video presentation. We used Slack a lot more this time around since we've gotten familiar with it. We came together and helped each other with any problem that we had with the assignment. It was challenging to assign tasks to everyone to be fair and not assign too much to any one person. Overall I'm proud of what we accomplished and I look forward with working more with my group.

Hyoungjun Lee

Parallel Programing Skills:

Identifying the components on the raspberry PI B+

HDMI port, USB port, Power port, SD card slot, Camera, display connecting slot, ethernet Controller, ethernet port, CPU/RAM

How many cores does the Raspberry Pi's B+ CPU have

it has quad-core, so 4 CPU

List three main differences between X86 (CISC) and ARM Raspberry PI (RISC).Justify you answer and use your own words (do not copy and past)

1. CISC has less register than RISC, also uses little endian when you store
2. CISC used for PC, or large capacity , because containing more feature instruction set, allow complex instructions to access into memory
3. RISC has simple instruction set, because main purpose is for small devices such as phone, tablets etc.

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

Difference between sequential and parallel computation is sequential can handle only one instruction when parallel computation can handle multiple, by using parallel computation we can multi-tasking, practical significance of sequential is easy writing, and parallel is multi tasking

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

Data parallelism – same computation or one operation into multiple data items

Task parallelism – allow one task to one core, another to one core, multiple function or thread than data parallelism

Explain the differences between processes and threads

when running the program, process does not share memory, thread is subset of process, threads share their common memory with the process, and same address

What is OpenMPand what is OpenMP pragmas?

OpenMP – compiler which can do multithreading, makes user's task simpler, also reducing an errors

OpenMP pragmas – compiler that directive to programs into parallel computation, generate thread code

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

Benefits of using multi-core application are data server, CAD/CAM , multimedia application ,data server

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

Multicore can running multiple process, allow to operate multi tasking, execute faster than single core, in single core it could be heat up by speeding up , muticore manage this problem, process in short time , output faster

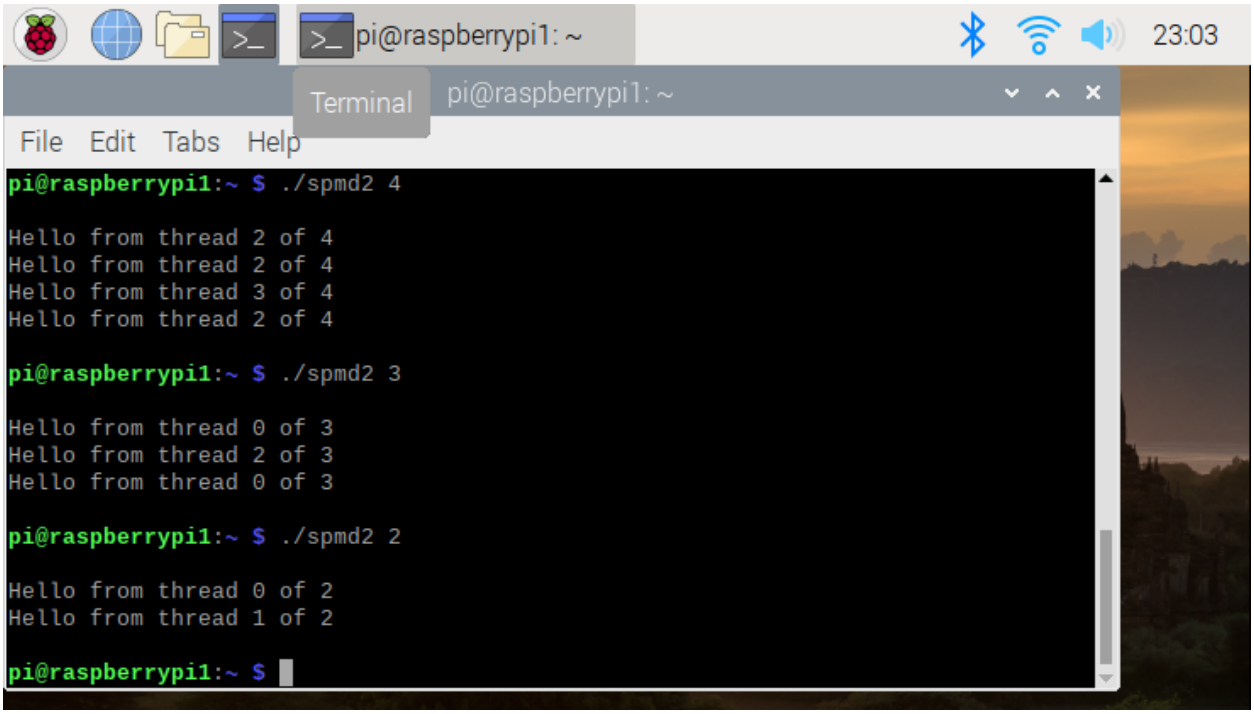
B) Parallel Programming Basics

as following guide from top to bottom, I wrote first code and ran it, it did not run what I expected because id declared in main method, but there's more than 1 id in code

As I write code into PI and check the result, I can double check all the cores share same memory, cannot have multiple variables in same id. Have to declare other

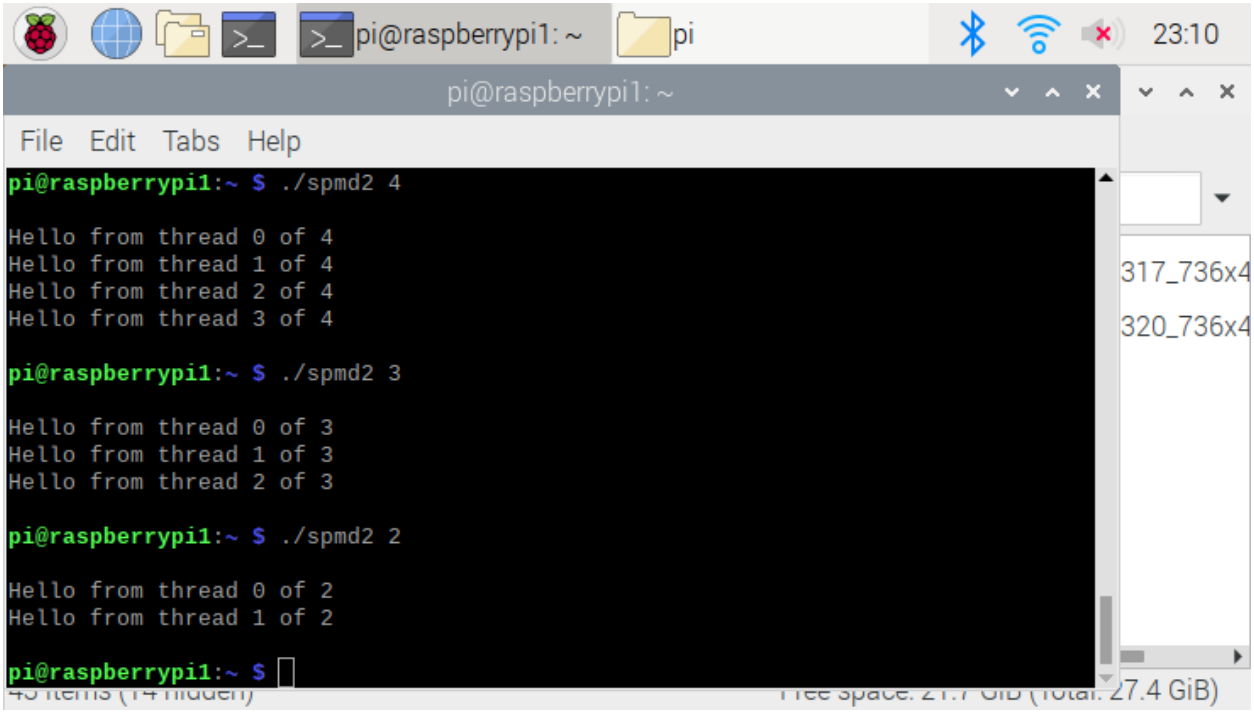
So first part did not run correctly,

As shown in below



Then change the id value inside, int id , I got correct answer what I expected

Second running result shown in below

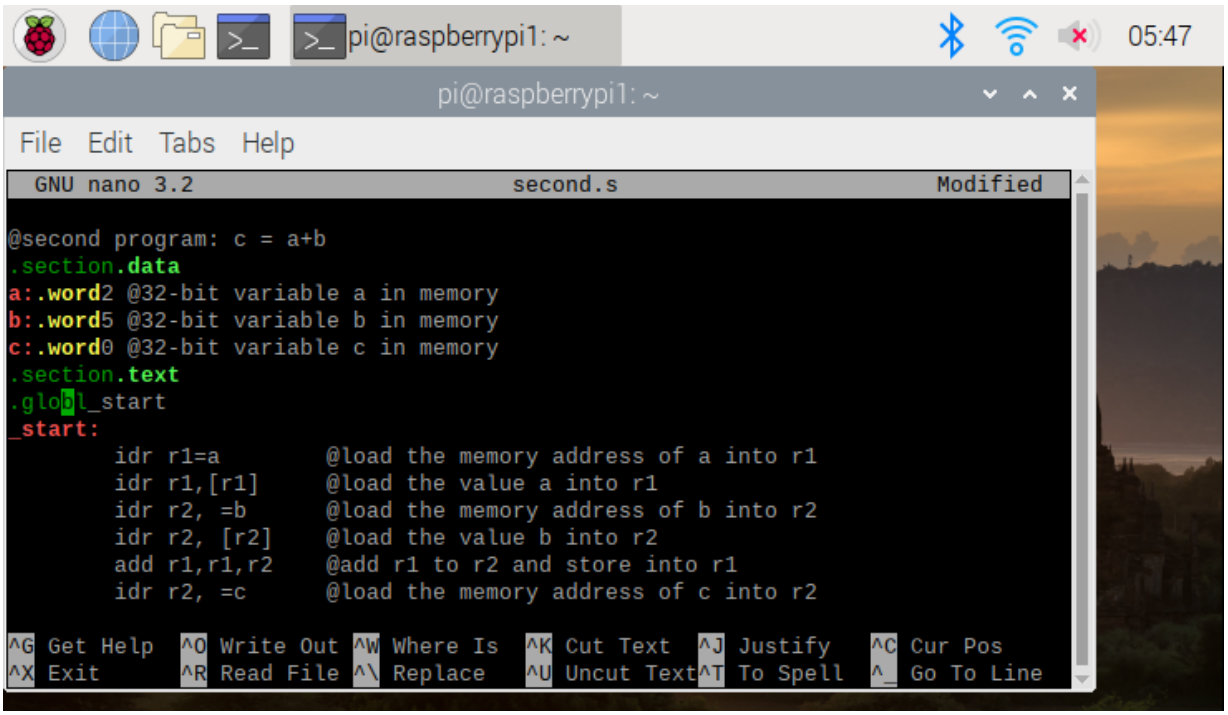


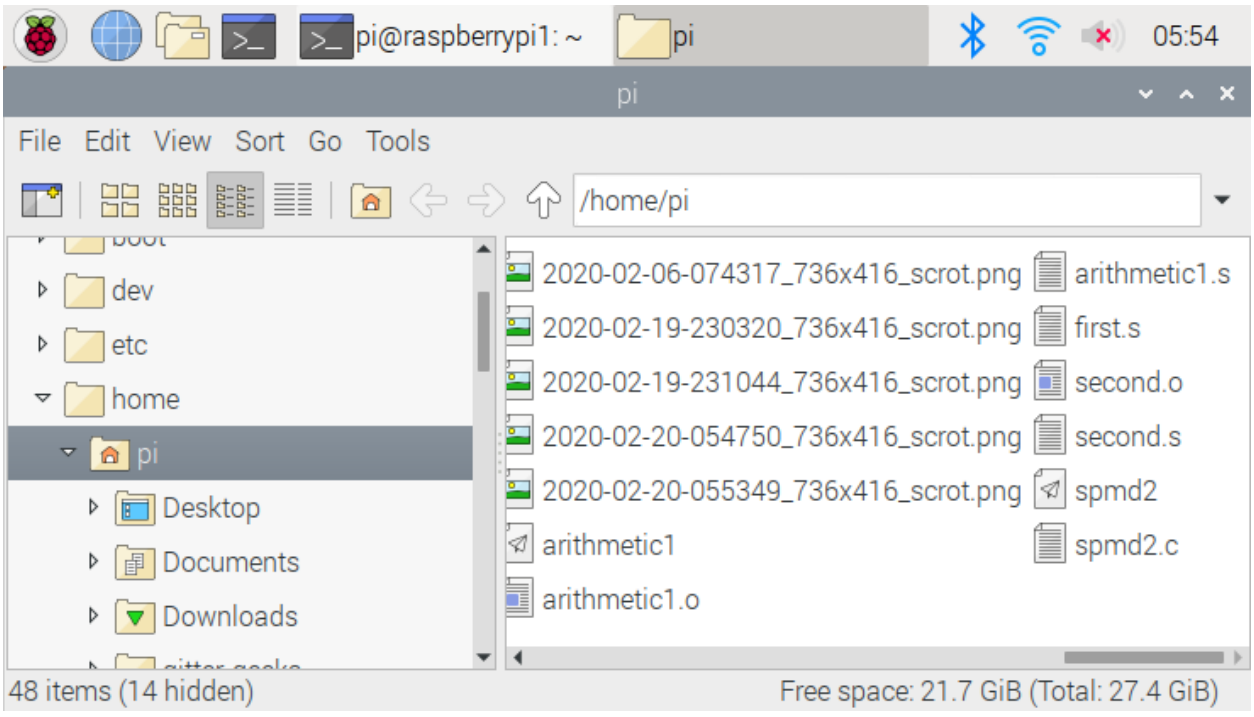
Now, working parallel

Task 4, ARM assembler in Raspberry PI

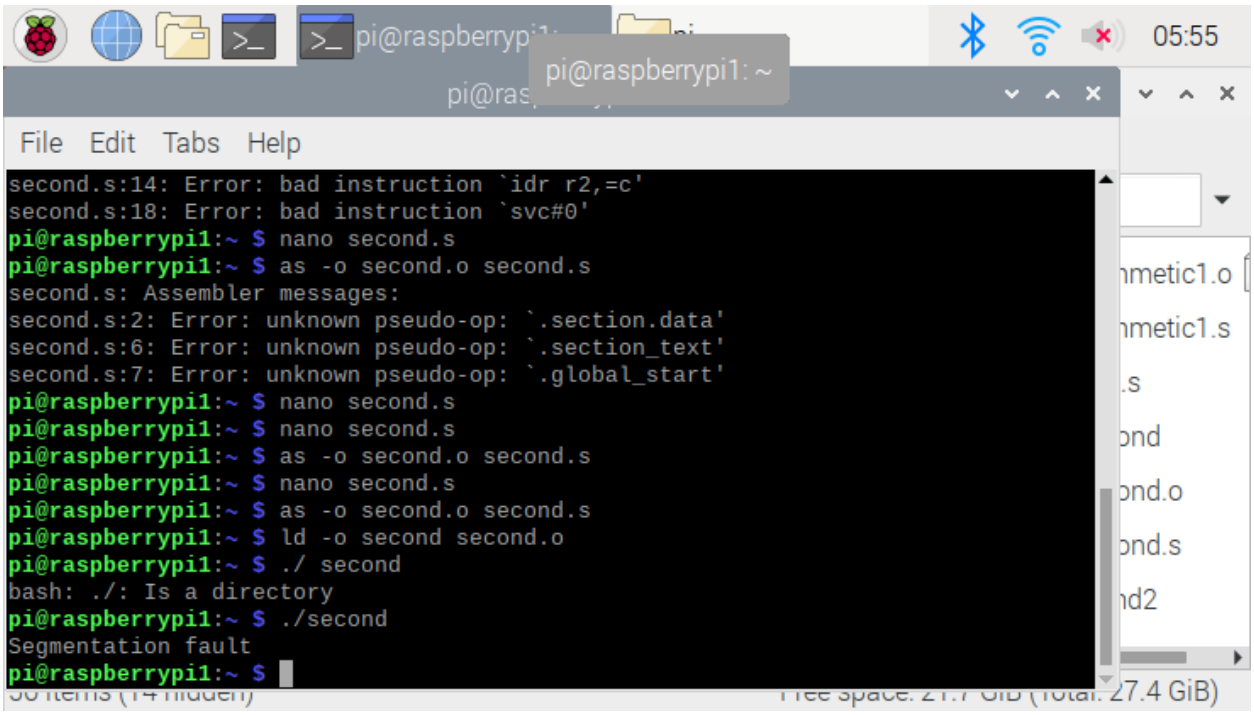
a) Part1, Second Program

Here is program to represent c = a + b, written code is below

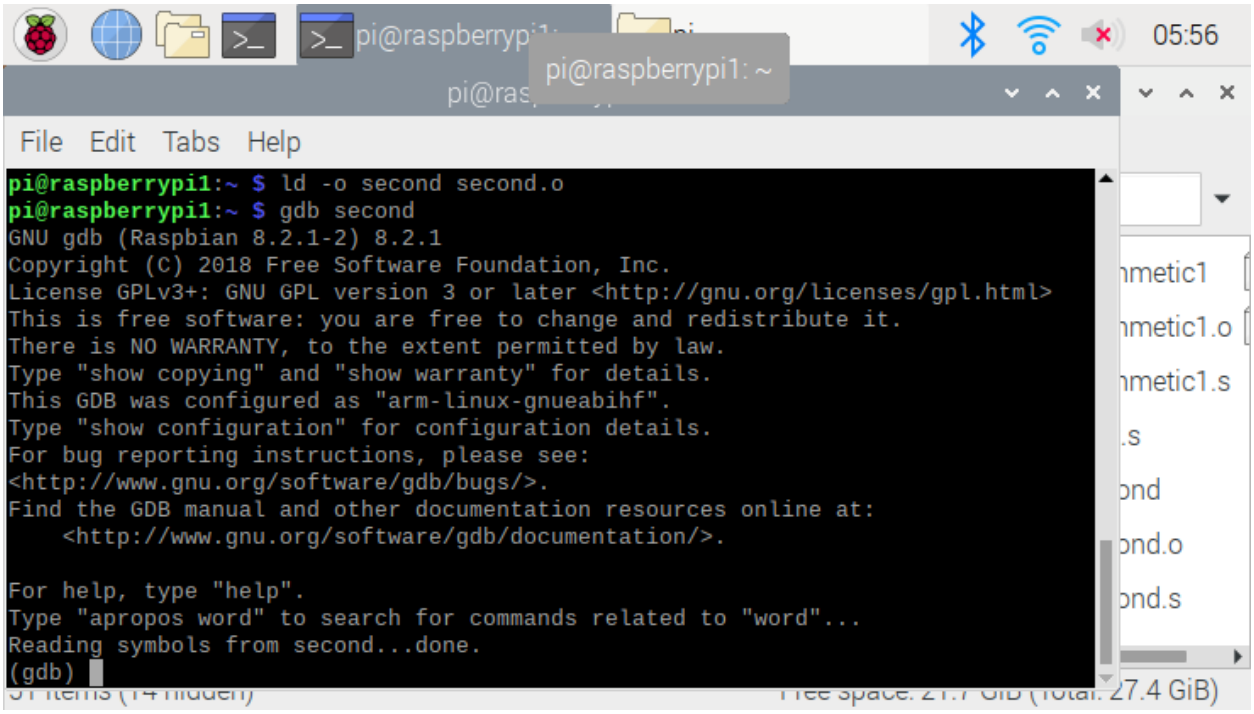


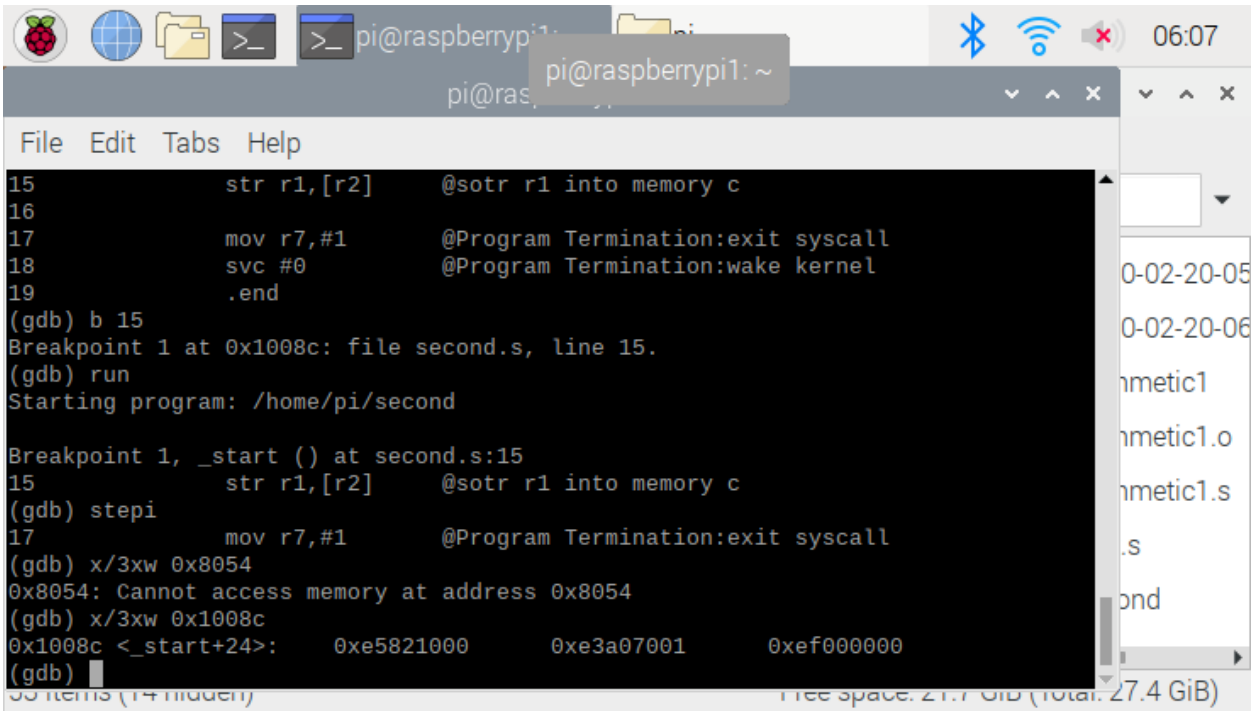


I assembled file first, and create second.o file, and link this file to executable, but there was no output we can find, all the output should be inside registry that we can't see it but we can see by debugging



After I see the all the list inside second.o by debugging, to stop debugging, we set break point at b 15, in order to step by step , I used “stepi” command to execute program, as shown in the below



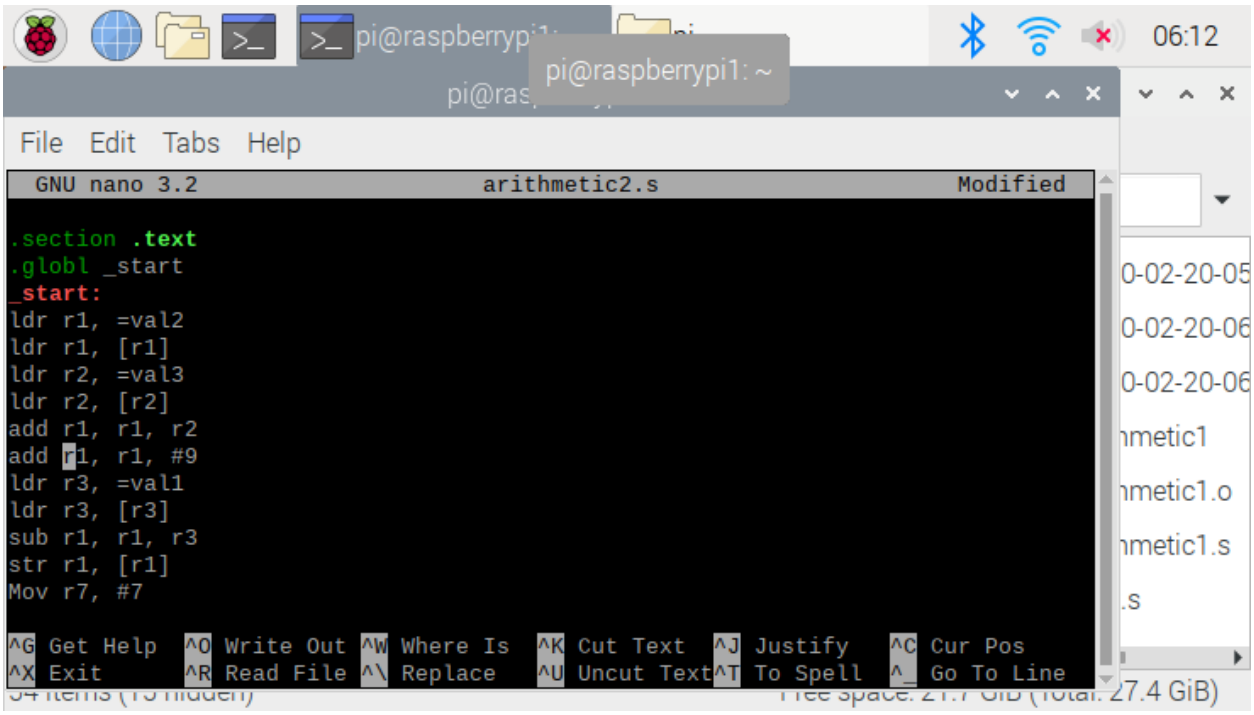


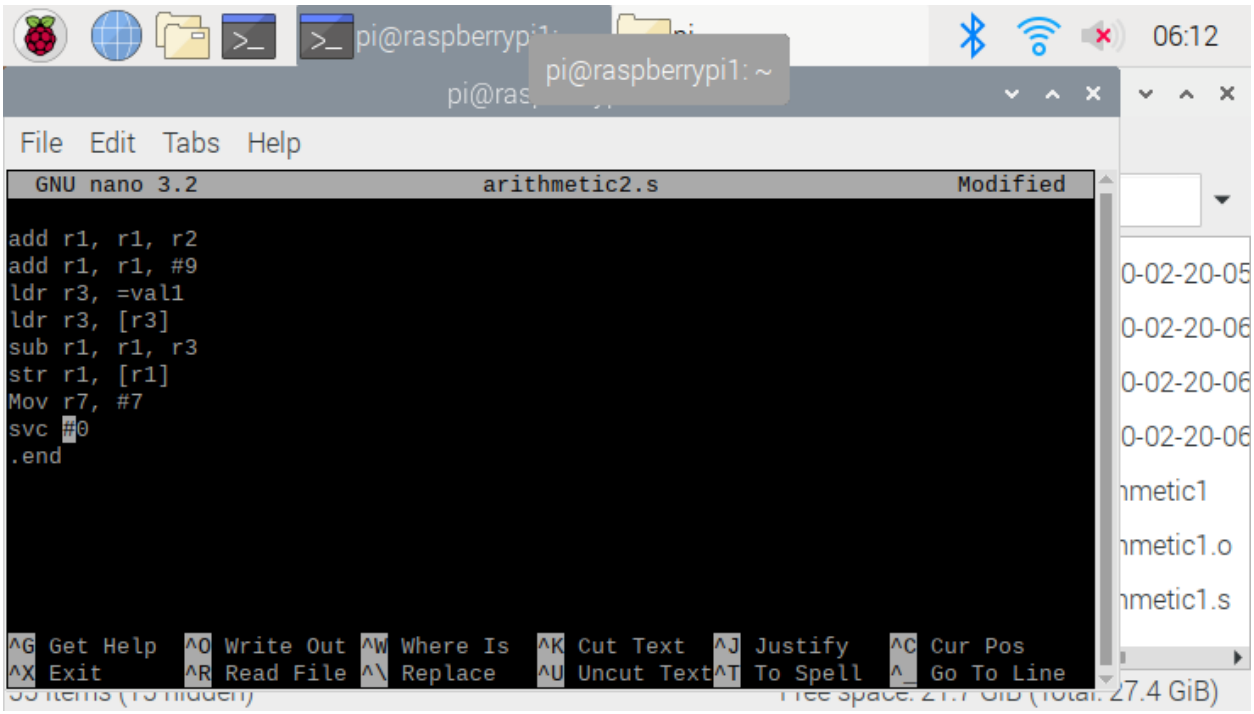
This command x/3xw 0x100990 was used to generate three words in hexadecimal: 0xe5821000, 0xe3a07001, 0xef000000.

Part2)

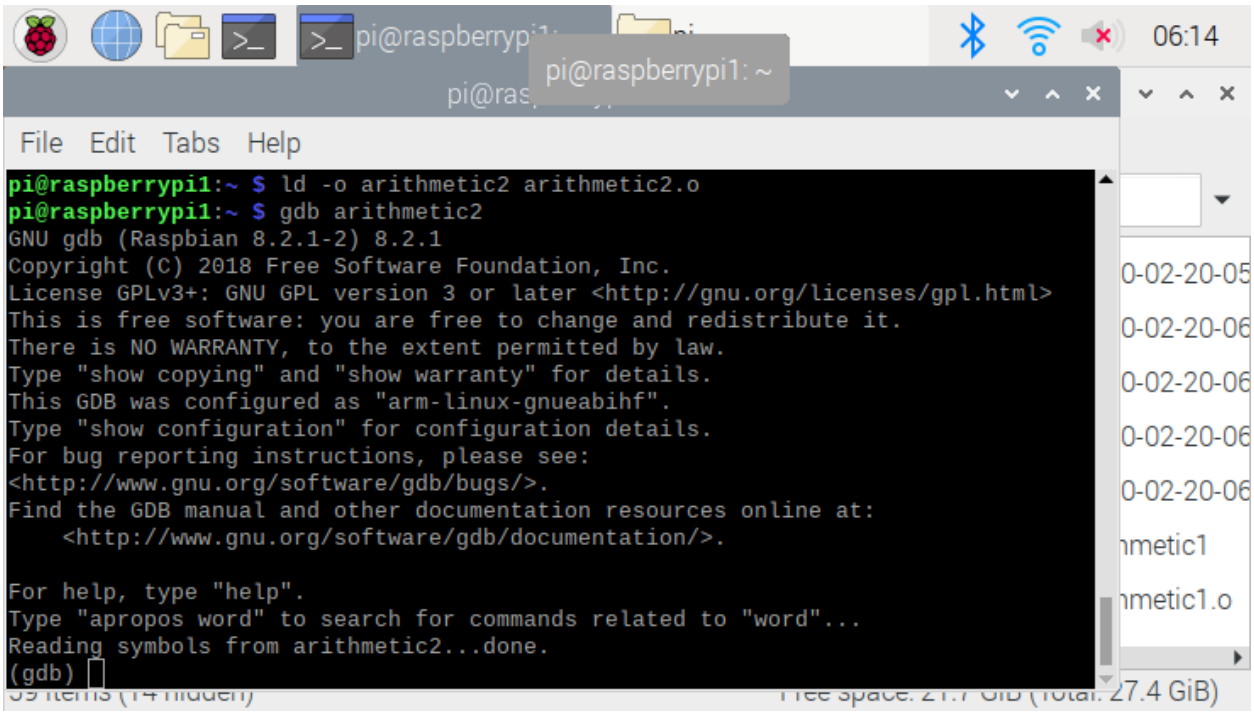
And here is program to represent Register = val2 +9 +val3 -val1

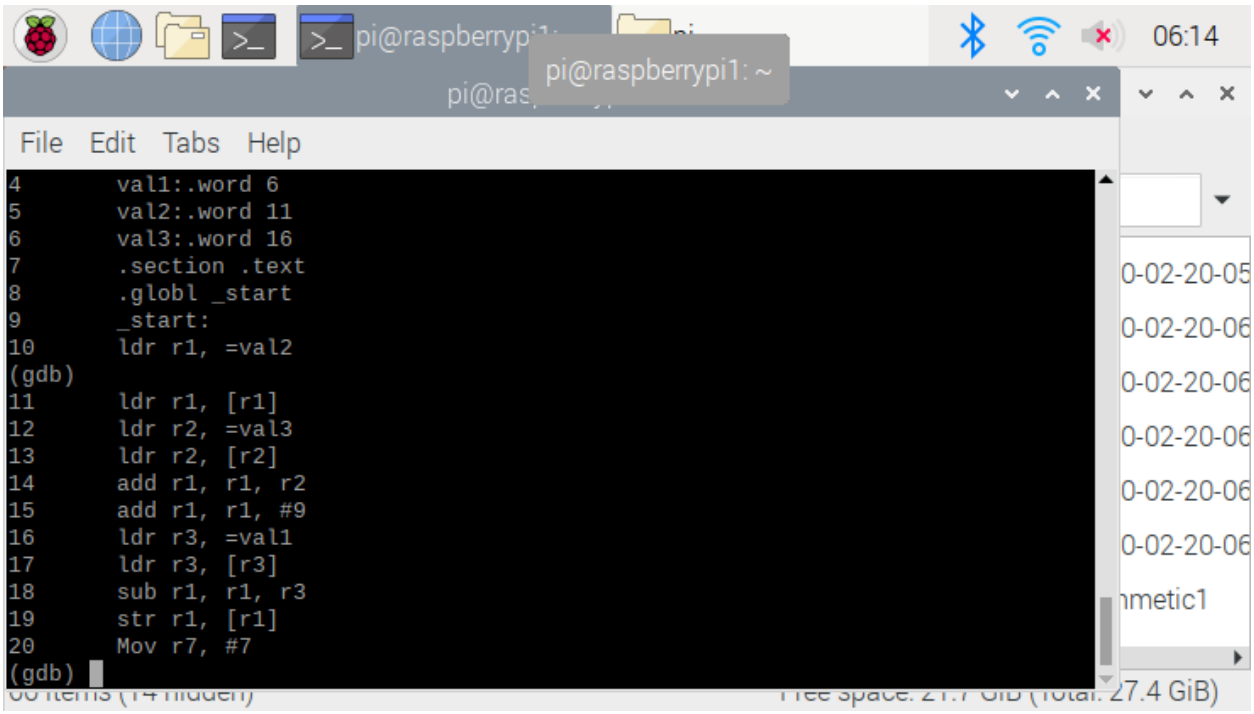
After write code inside arithmetic2.s, I assembled file and linked it in order to receive an executable file, but also, same as part 1, I could not see any output, because, all the things inside register, and I debug to see output (GDB), picture of codes are below



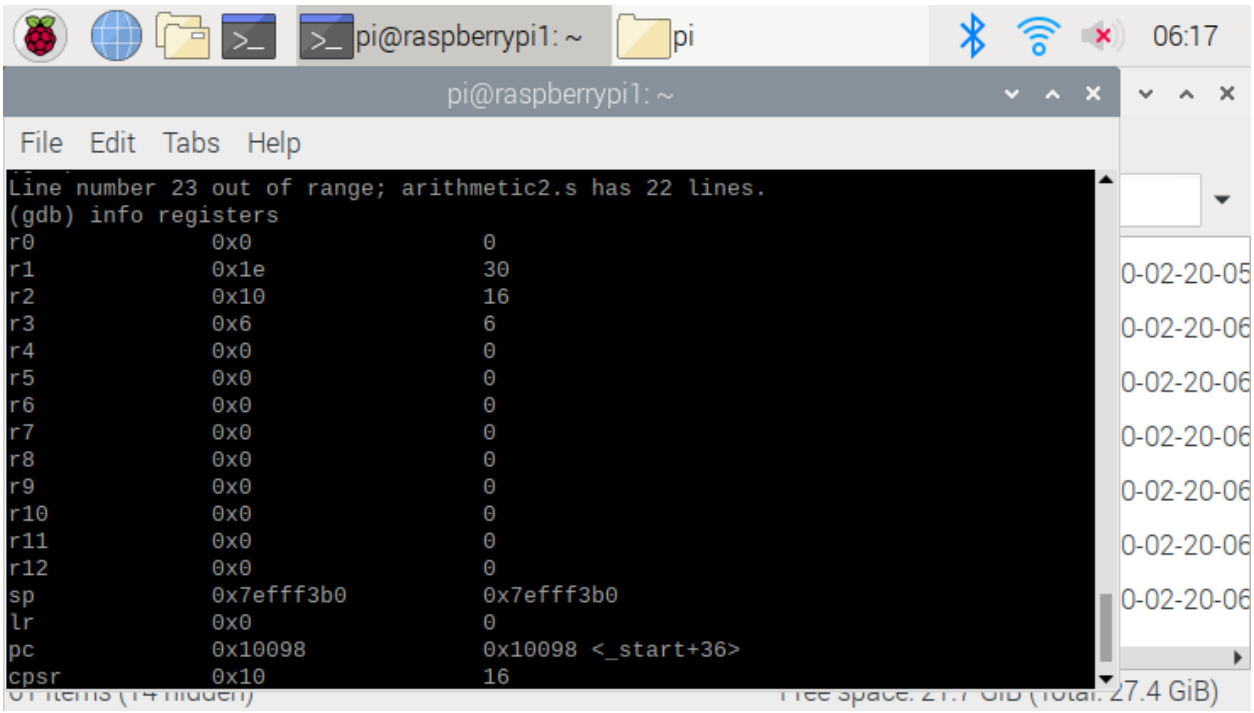


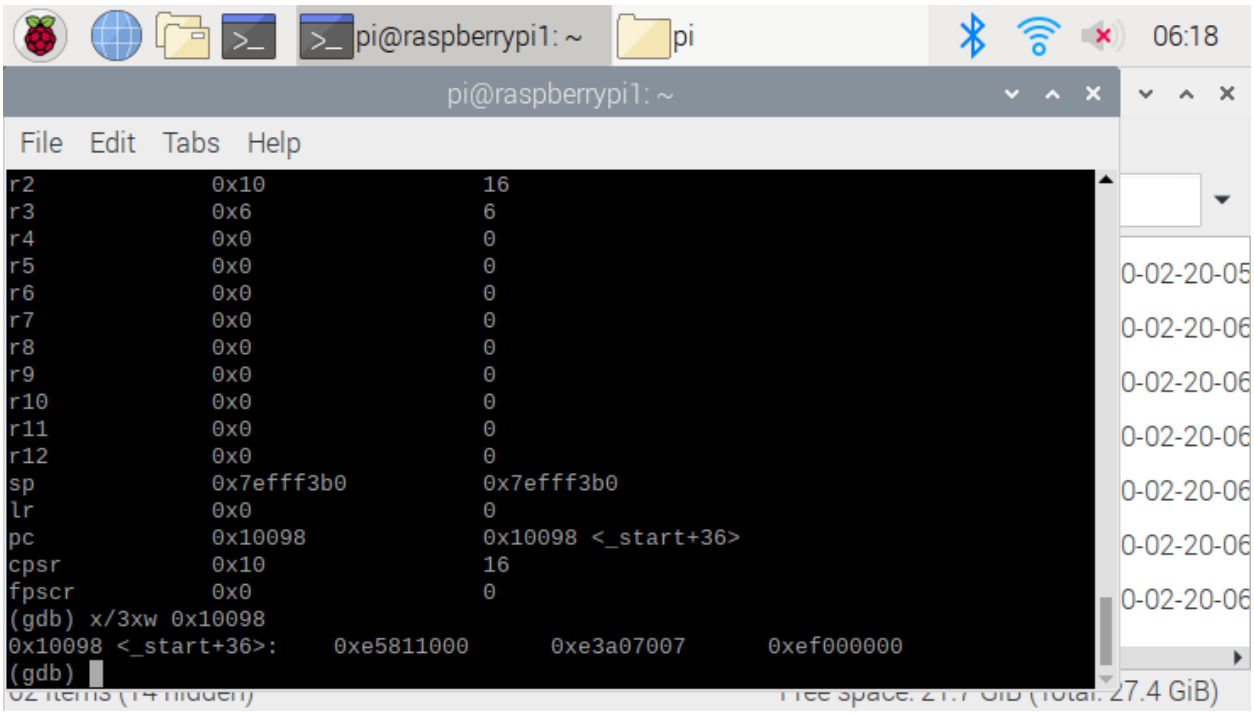
As I did in part 1, I set the break point to see how program work through, to stop debugging I set break point at “b 19” to make sure, I used “info register” to check value of register and compare to the result.
to execute step by step, I used “ stepi” command to help this out. Pictures are below





This command x/3xw 0x10098 was used to generate three words in hexadecimal: 0xe5811000, 0xe3a07001, 0xef000000 below picture





As the result, we can check that register 1 value is same as our equation, Register = val2, +9 +val3 – val1

Which is 30, register 1 through 3, have same value which I declare value inside .data section.

Michael Knight

Parallel Programing Skills:

Part: A

1. Identifying the components on the raspberry Pi B+.

Ethernet port, 4 USB ports, HDMI port, micro USB port, Ethernet controller, CPU and RAM, Display

2. How many cores does the Raspberry Pi's CPU have?

Quadcore - 4

3. List three main differences between X86(CISC) and ARM Raspberry PI(RISC). Justify your answer and use your own words.

ARM uses less power, due to the fact that their cores don't require heatsinks to help reduce their heat.

ARM is primarily used for programing of smaller devices where X86 is used for larger computer programing.

RISC is a more register based programming then CISC uses more memory.

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

Sequential computation is as the name suggests one task after another. Easier code.

Parallel computation helps reduce downtime while the CPU waits to receive information to carry on with the next task, it goes ahead and starts the next request so instead of waiting as long it starts the next step while it waits for the information.

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

Data Parallelism allows multiple data operations to be computated at once.

Task Parallelism refers to a running multiple tasks at once.

6. Explain the differences between processes and threads.

A process is when a program runs independently, and threads are multiple processes that share information.

7. What is OpenMP and what is OpenMP pragmas?

OpenMP is an interface which supports parallel computation, and pragmas is a complier that supports multithreaded code.

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

Multimedia applications, Web Servers, CAD/CAM, Compliers

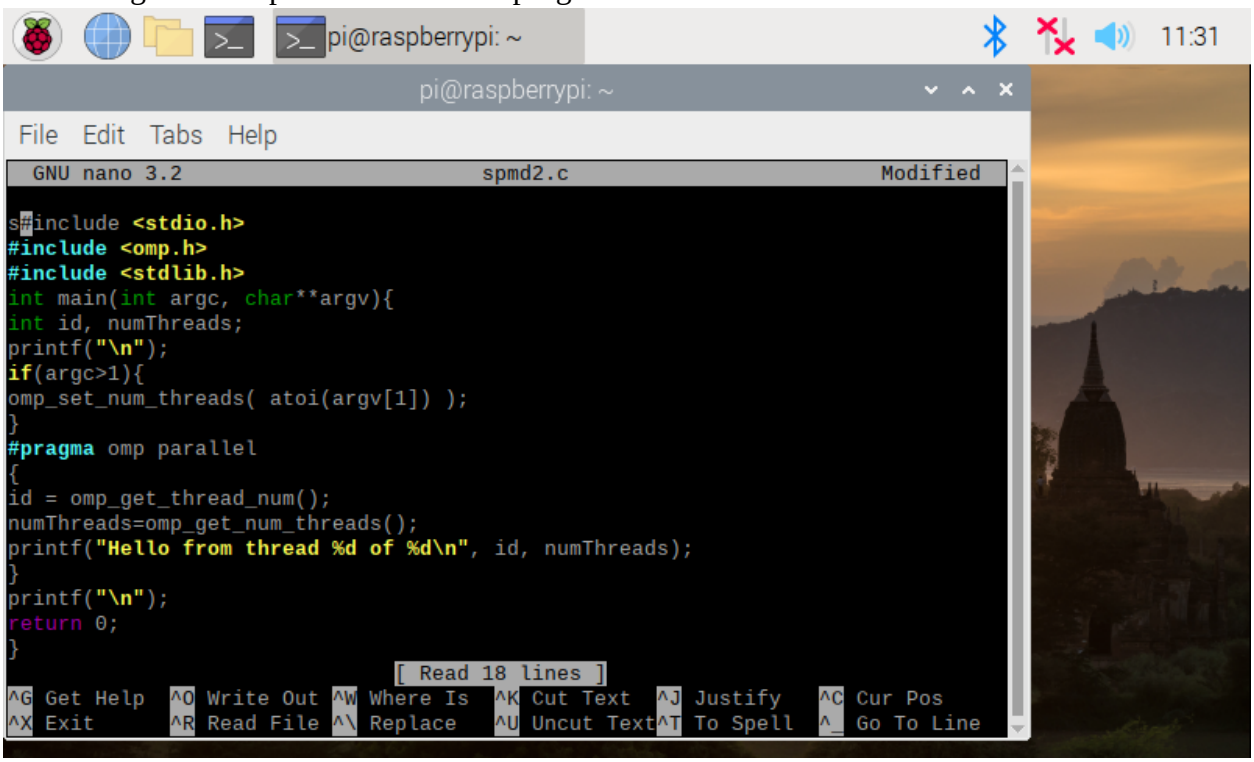
9. Why Multicore?(why not single core, list four)

Faster results while computing, allows for multitasking, more energy sufficient, more compact.

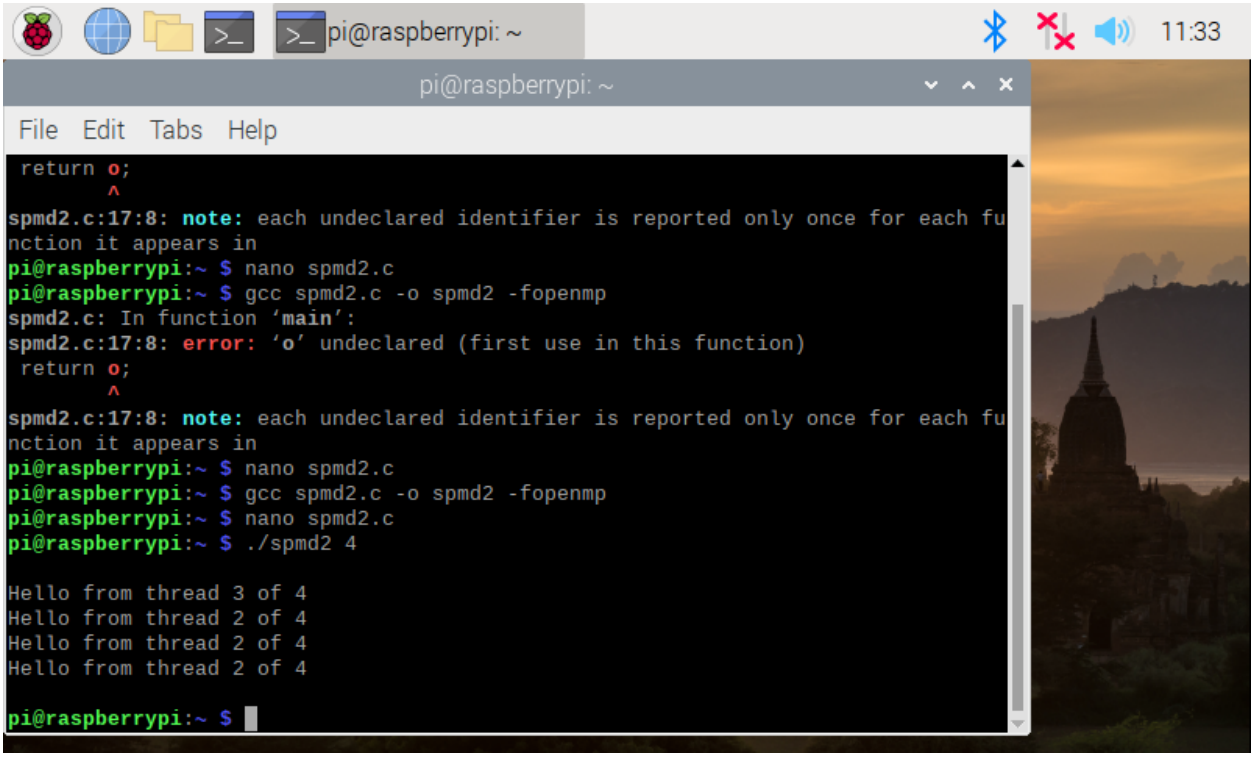
Part: B

Parallel Programing Basics

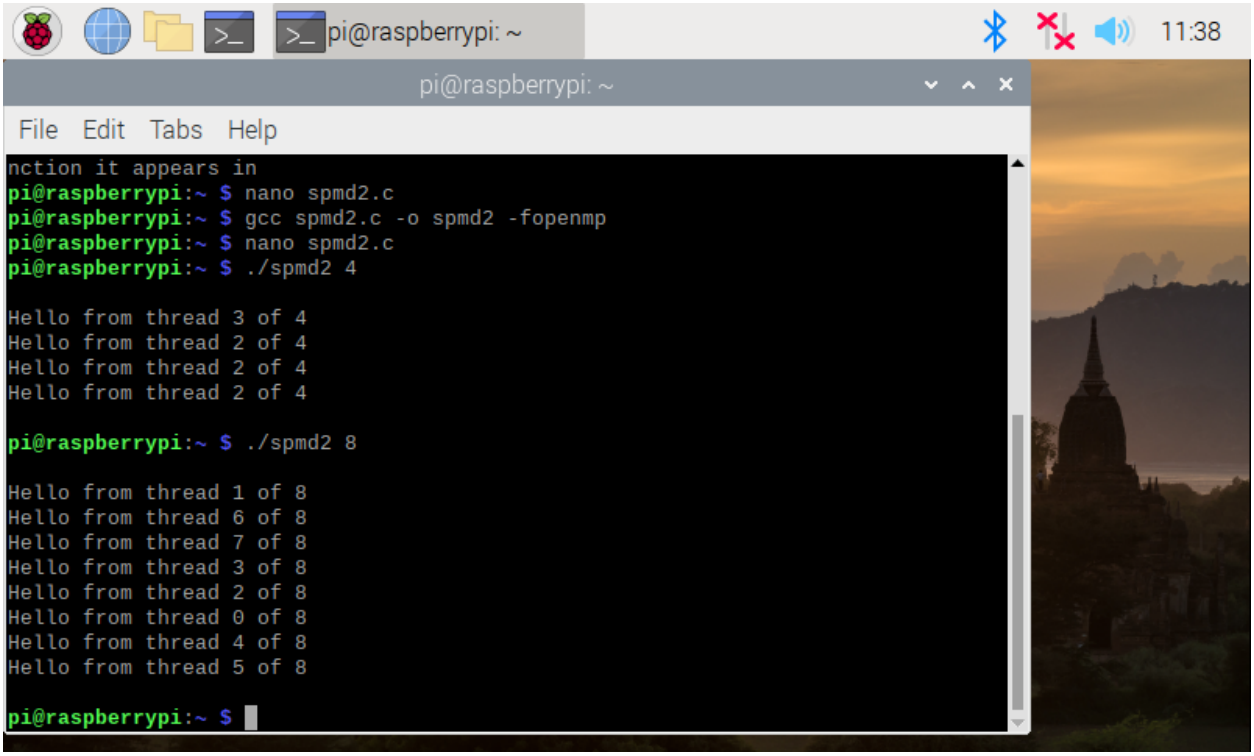
First Using the code provided I ran this program.



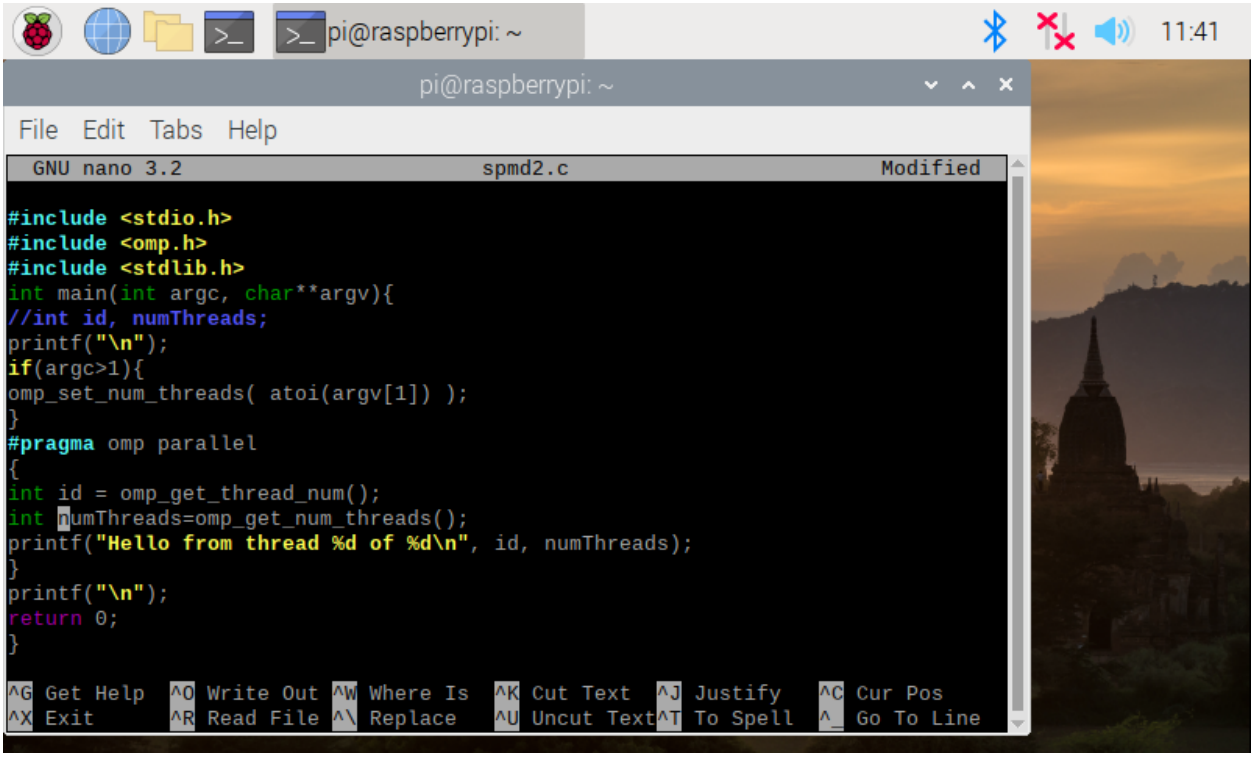
Which gave me this result.



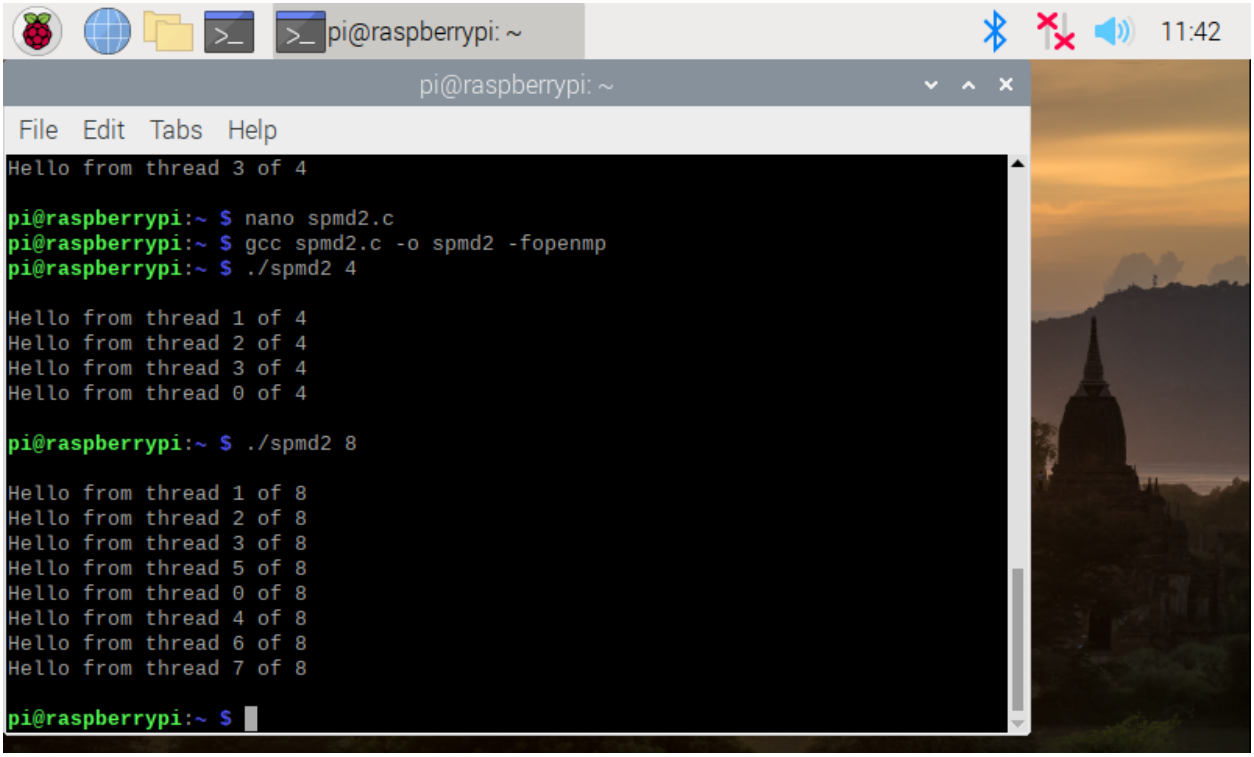
I would have thought each tread 1 threw 4 would be displayed but instead 3 2 2 2 was the result. I also ran tests.



I then made the corrections to the code



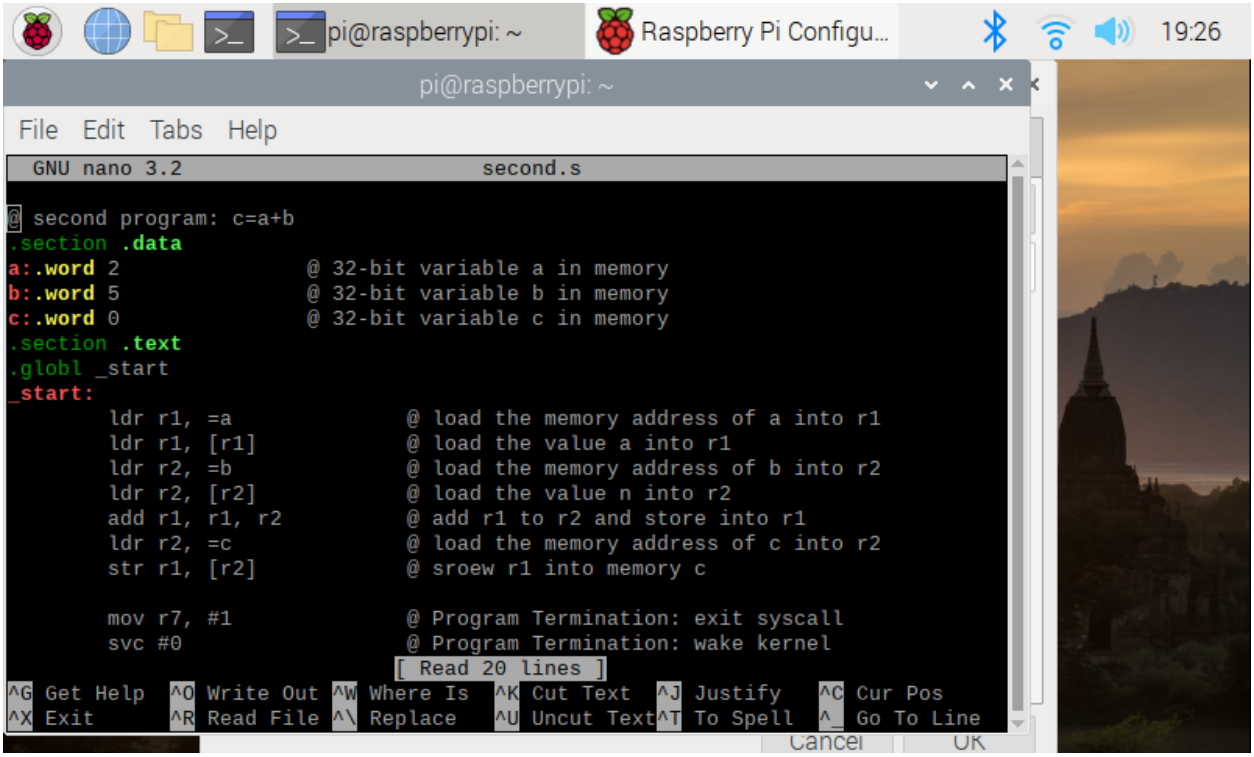
And ran the code.



Arm Assembly Programming A2:

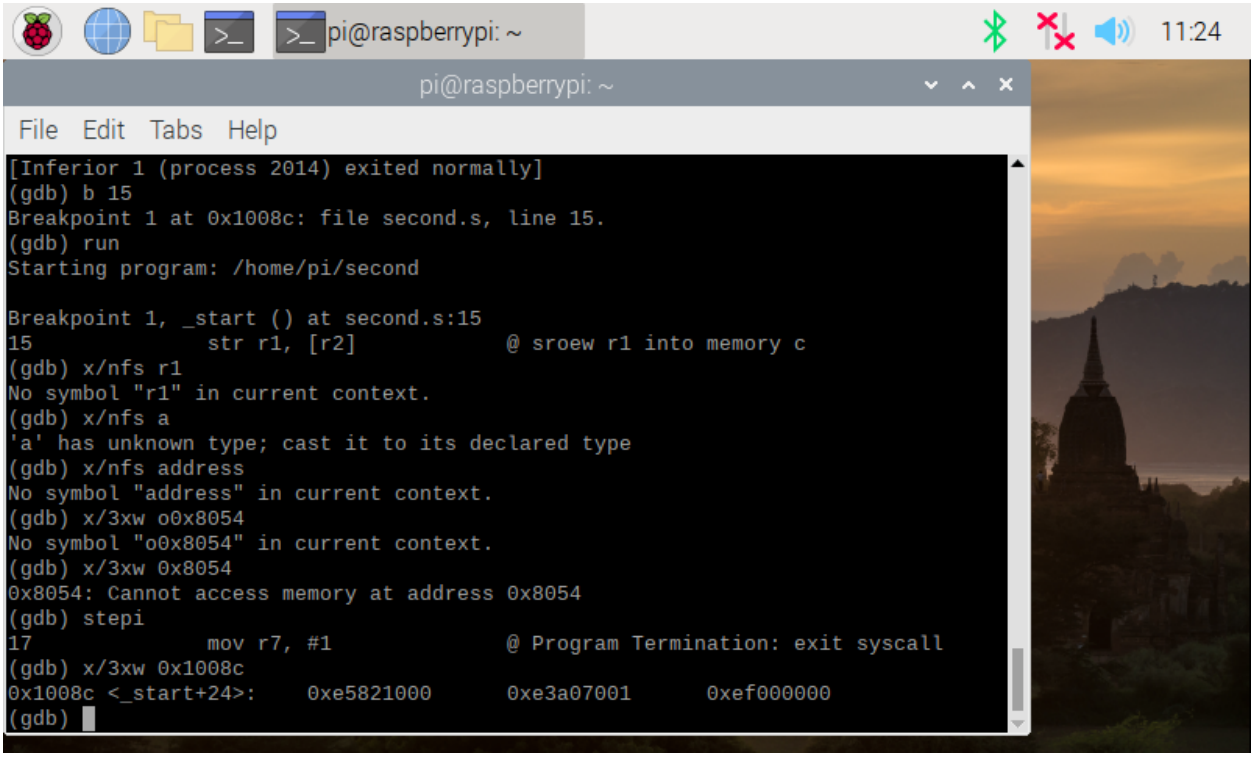
PartA:

First I coded the sample code as instructed:



After assembling, linking, and running I saw no output because there is no output to display, its only storing and playing with registers.

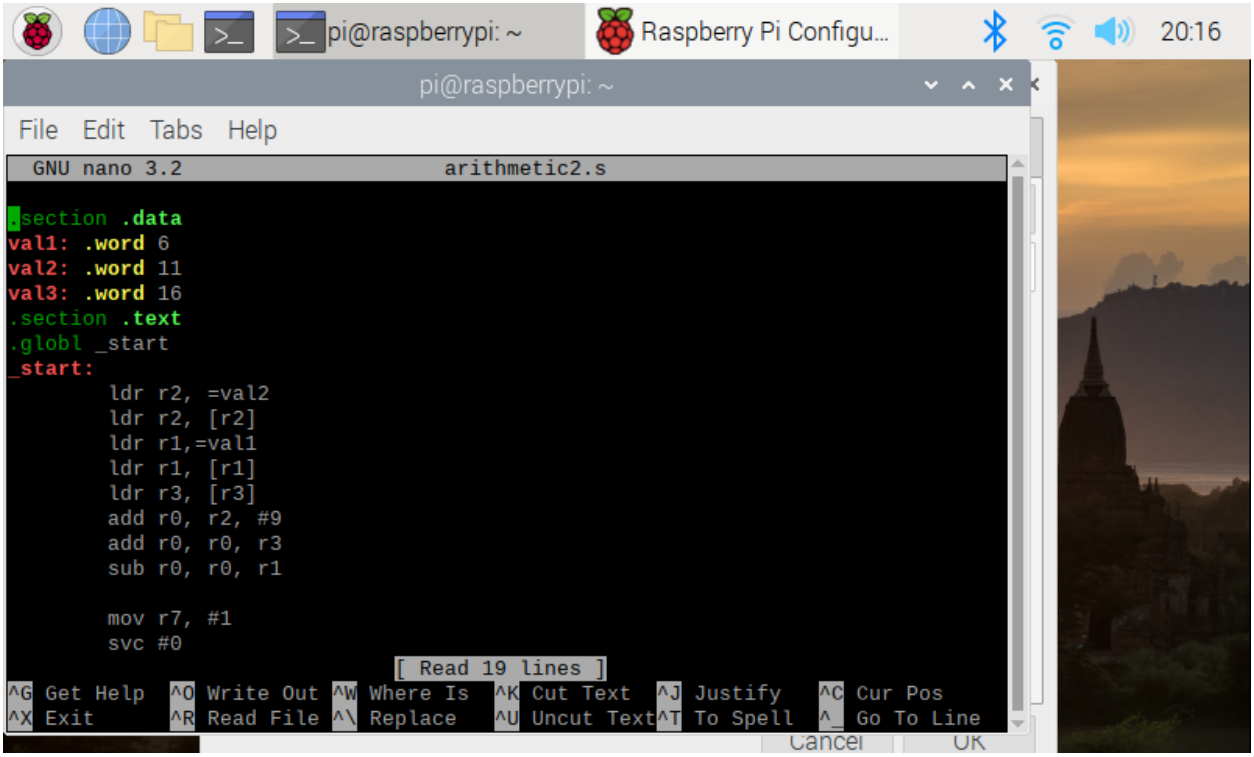
Then ran debug on the program with a break point at 15 aswell as checking the register



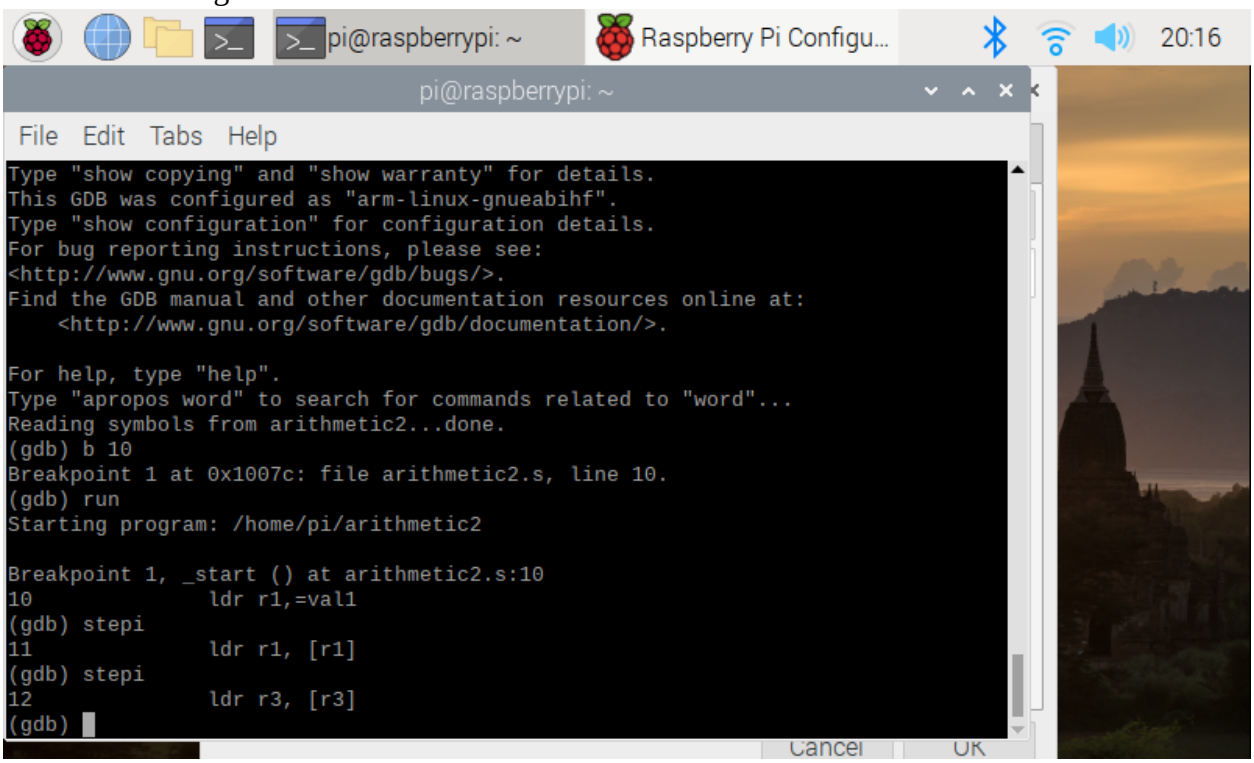
Gave me the results 0x25821000 0xe3a07001 0xef000000

Part2:

Coded what was instructed Arithmetic2:



Then I ran Debug on it as instructed:



Tek Acharya

Parallel Programming Skills:

- Identify the components on the Raspberry PI B+ (5)

CPU/RAM, Power socket, HDMI socket, ethernet controller, USB sockets, ethernet socket, camera connecting pin, display connecting pin, SD card slot, transistors, capacitors, diodes, serial buses, audio connector and many more.

- How many cores does the raspberry pi's B+ CPU have? (5)

The raspberry pi B+ version is a quad-core; so, four cores.

- List three main differences between x86(CISC) and ARM Raspberry PI(RISC). Justify your answer and use your own words (do not copy-paste) (8)

x86 processor has relatively fewer registers as compared to ARM. This is because x86 has a complex and ARM has reduced instruction sets.

x86 is used in PCs, large workstations and supercomputers whereas ARM is used for smaller purpose devices such as phones, routers, wearables, and tablets. Since ARM uses reduced instructions, it became easy to use them into smaller devices with fewer transistors and integrated circuitry.

ARM processor uses a register-register or load/store memory model but x86 uses the register-memory memory model. This is two separate ways of computing while in a clock cycle of instruction. ARM needs both operands in its register whereas for intel it will compute by loading one into its register and others can be done directly from the memory.

- What is the difference between sequential and parallel computation and identify the practical significance of each? (6)

The main difference between sequential and parallel computation is that in sequential only one instruction is handled by a processor at a time whereas in parallel computation multiple instructions are handled at a time using multiple cores/processors. Multi-tasking has been made easier due to this parallel computation technology. Editing a video while playing it is possible while using parallel computing but since only one instruction is processed at a given time both tasks cannot be possible in sequential computing.

- Identify the basic form of data and task parallelism in a computational problem. (5)

Data Parallelism:

One program → multiple data; single operation using multiple data while running a parallel computation is the base form of this problem-solving scheme. The data parallelism allows programmers to write scalable programs though the size of input and output.

Task/thread Parallelism:

The specification is given to multiple functions or threads rather than to data parallelism. In this system of parallelism, however, the computation is difficult to scale as it is hard to ensure that all works are done properly that contributed to the result.

- Explain the differences between process and threads. (6)

An executing instance of a thread is called a process.

The process is sure to execute but a thread may not always execute.

Thread is a subset of a process.

Usually, a process has only one thread of control – one set of machine instructions executing at a time.

All the threads running within a process share the same address space.

Usually, a process has only one thread of control – one set of machine instructions executing at a time.

- What is OpenMP and what are OpenMP pragmas? (3)

An OpenMP is a compiler with an implicit multithreading model reducing the error to a programmer. OpenMP pragmas in code mark through special directives that marks the programs into parallel computation models. Here each slave threads run independently completing its tasks and recombine later into its master while completing their tasks.

- What applications benefits from multi-core (List four)? (4)

Database server machines

Web servers

Scientific application

CAD/CAM

Multimedia applications

Compilers

- Why multi-core? (Why not single-core, list four?) (4)

Multi-core allows us to;

Process output faster

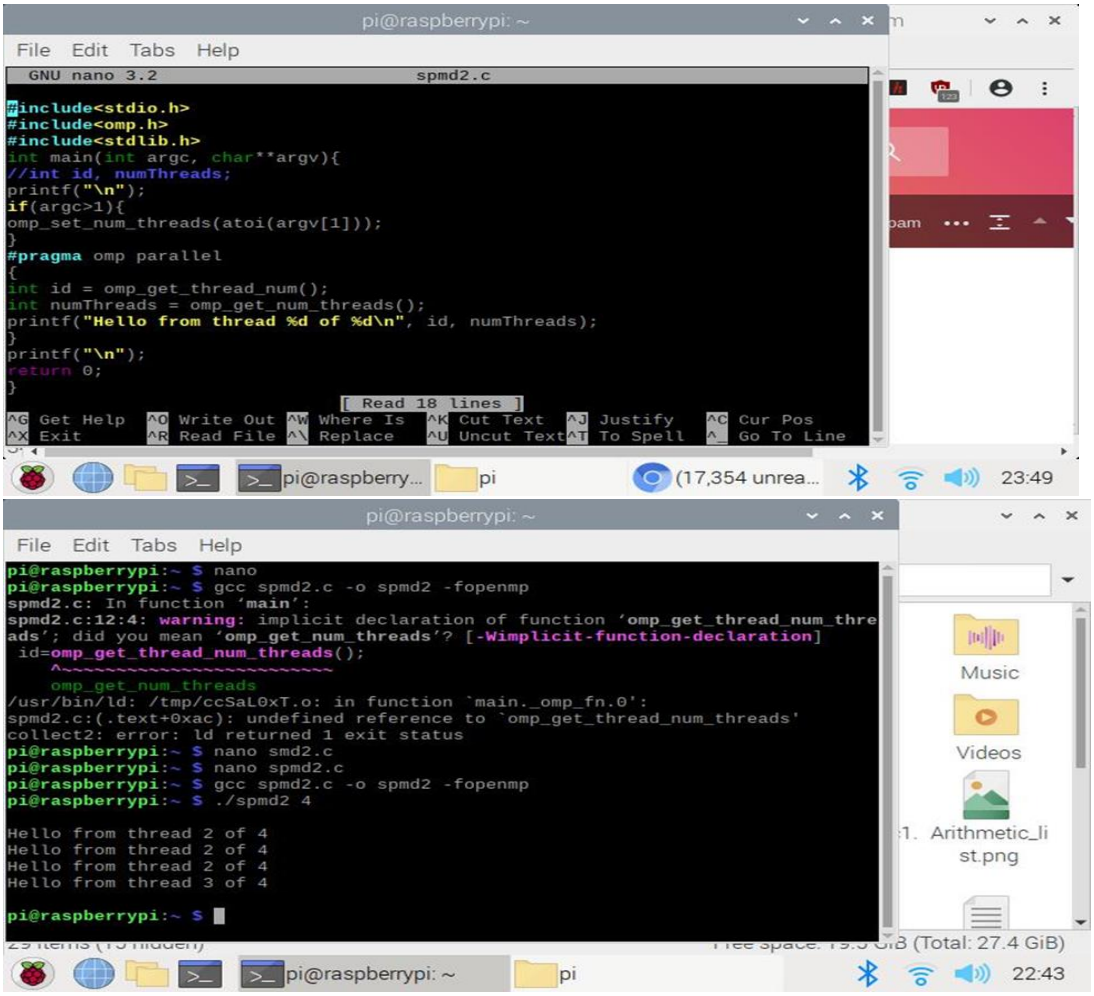
The need for multi-threaded tasks such as multi-tasking at a time in a computer

The computer does not get hot very often as all programs get executed faster

Low probability of signal degradation as it is processed in a short time.

The need for more parallelism in computing.

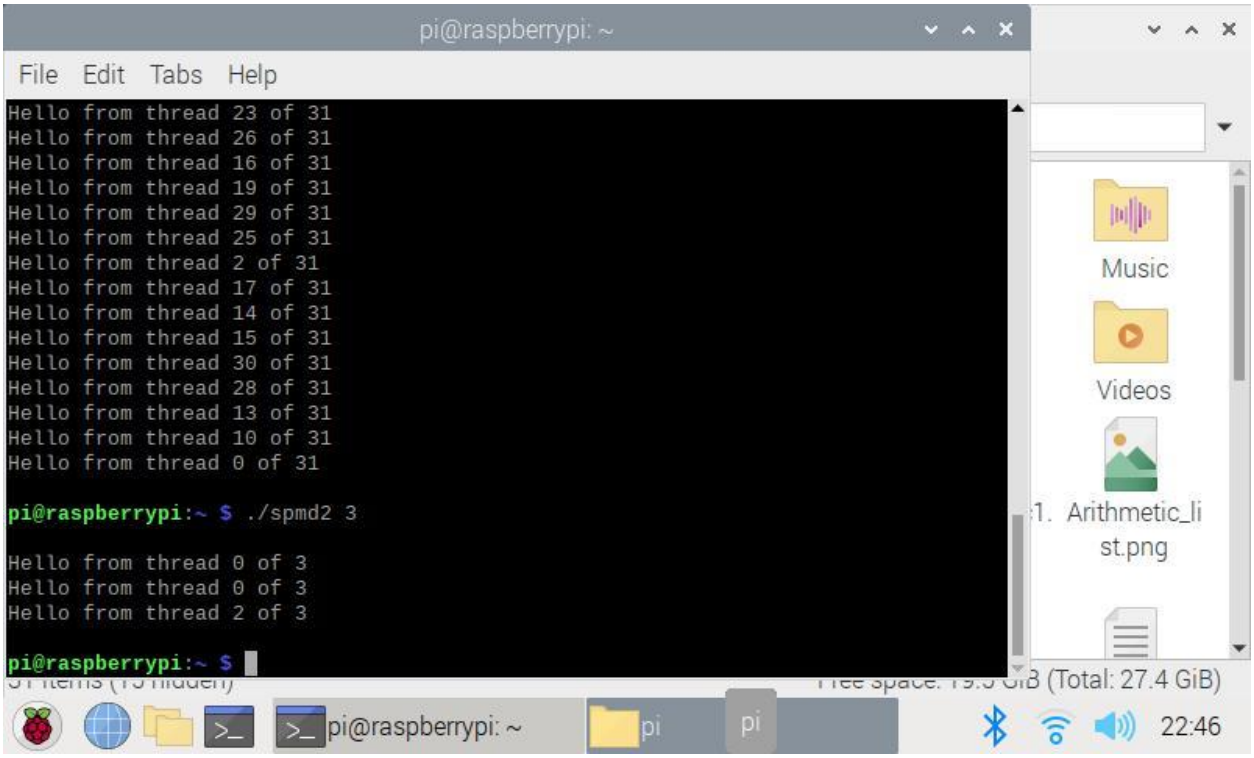
Parallel Programming Basics:



Compiled and ran the spmd2.c file in nano and tried pragma with 4 threads in parallel.

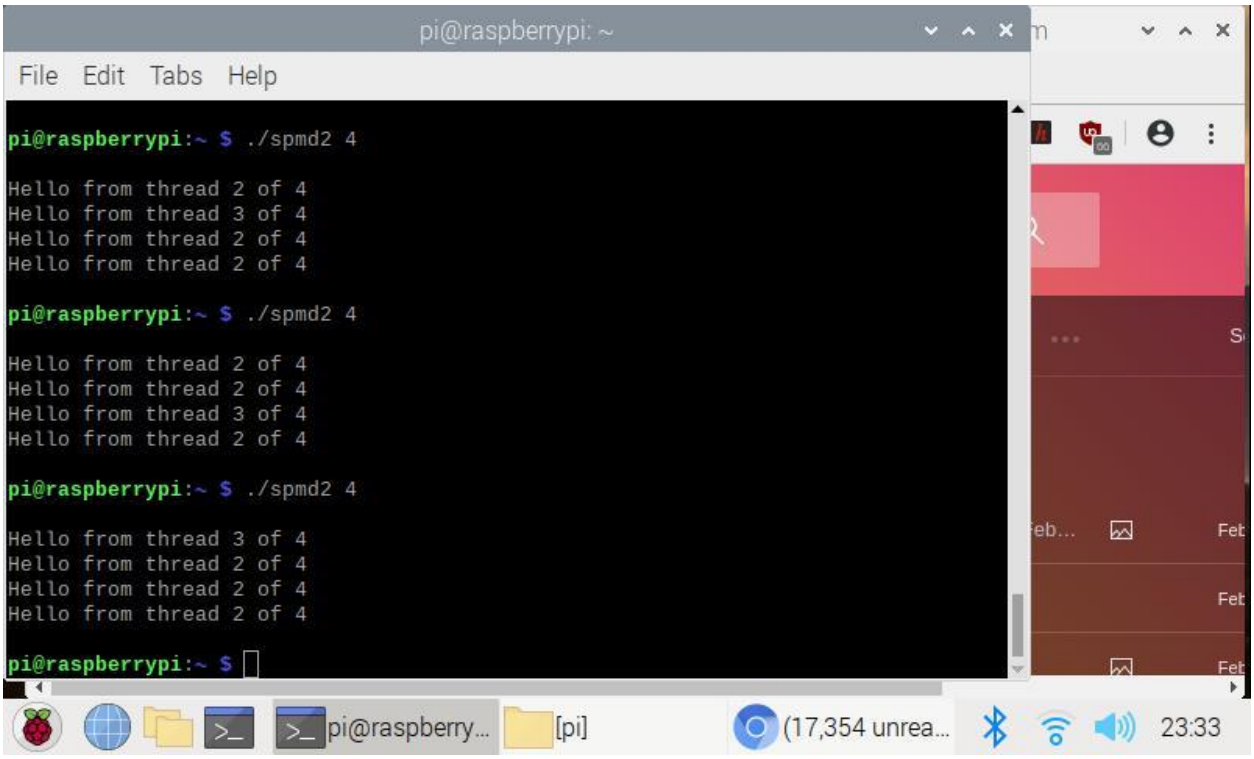
Observed that only three from id 2 and one from id 3 has been involved out of four cores.

This is because, the parallel programming hasn't been started that started with curly braces from where the pragma has started. Even though the code was there, it was incorrect-the variable declaration wasn't done.

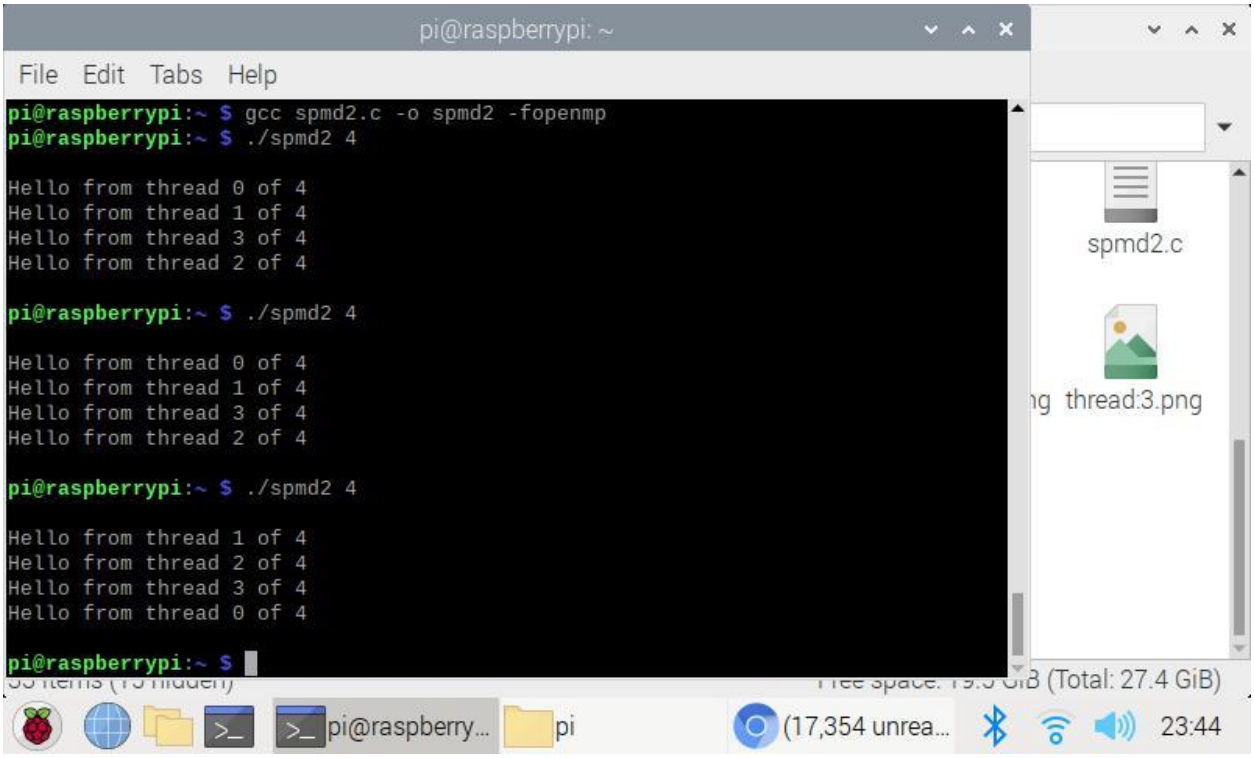


Tried running with different thread counts:

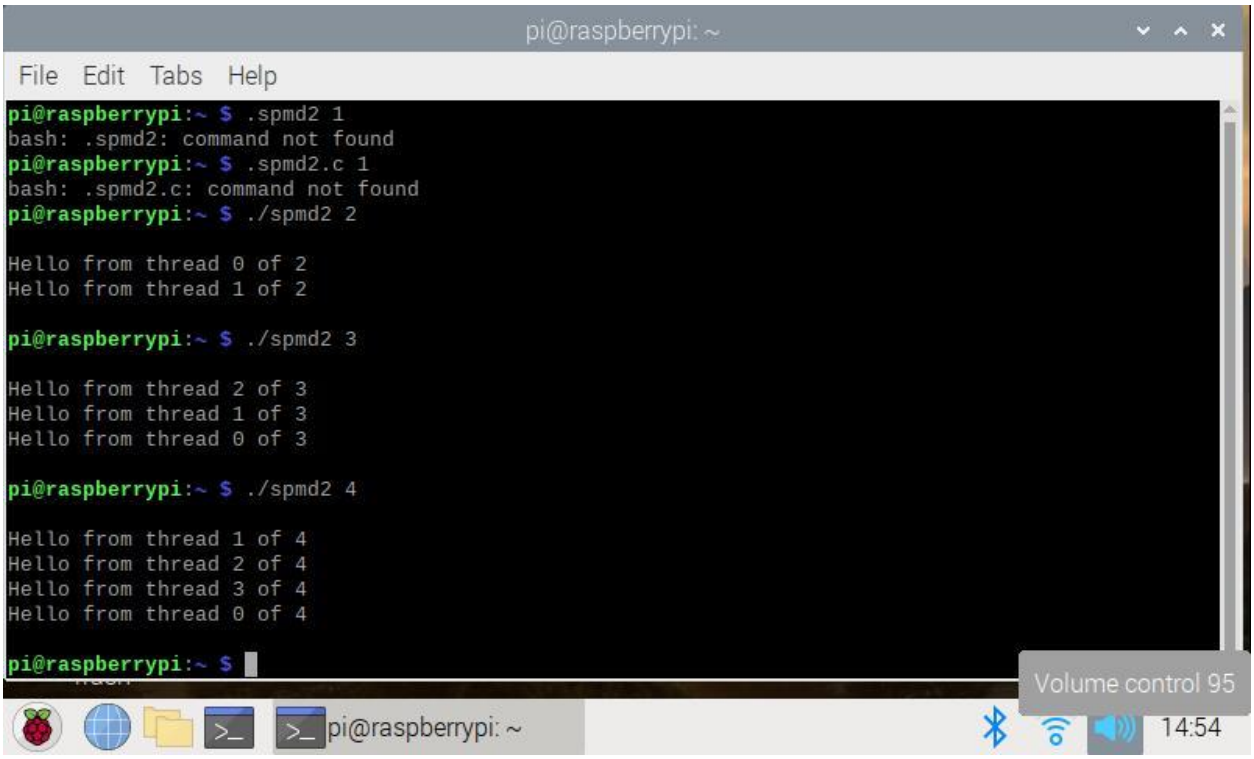
Noticed that not all cores have been used.



Tried with thread 4 multiple times. Noticed that only two cores have been used as before.



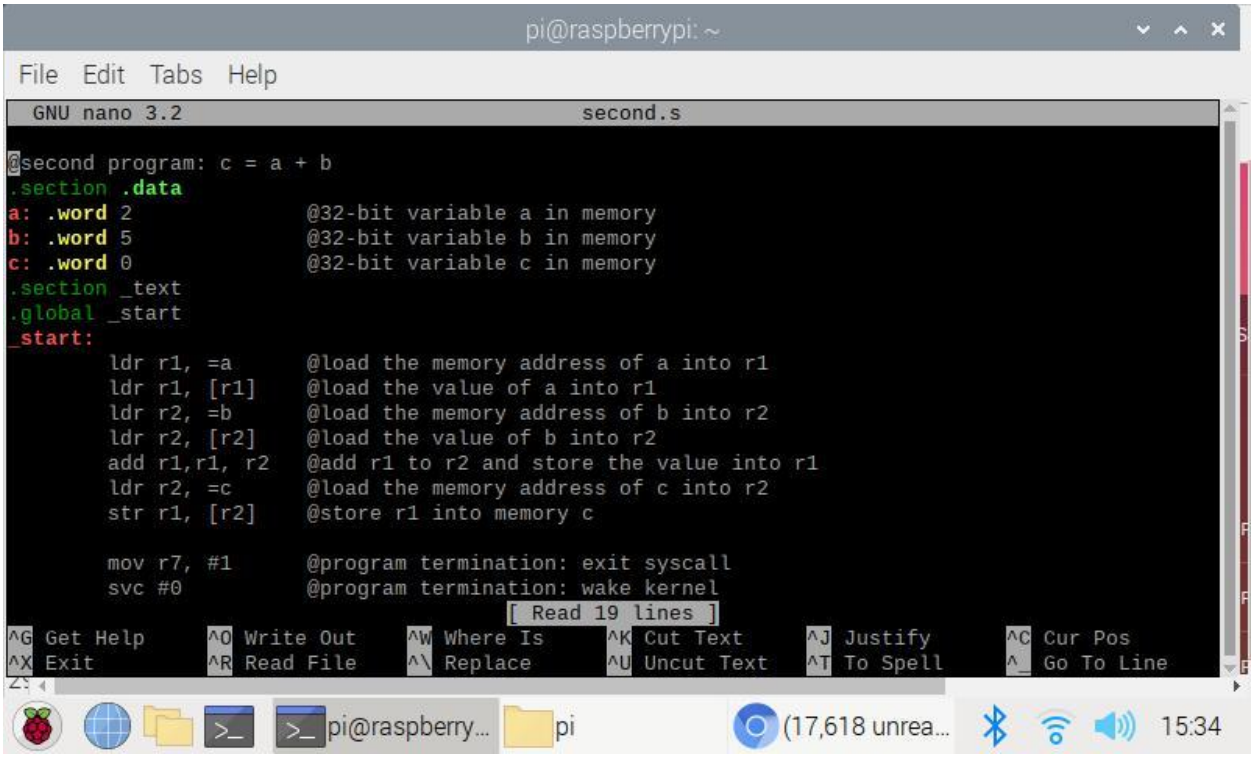
Once we fixed the code errors per instruction, we can see each thread coming from one core parallelly. This is because the pragma started the parallel computation as we declared the variable type within the curly braces of pragma.



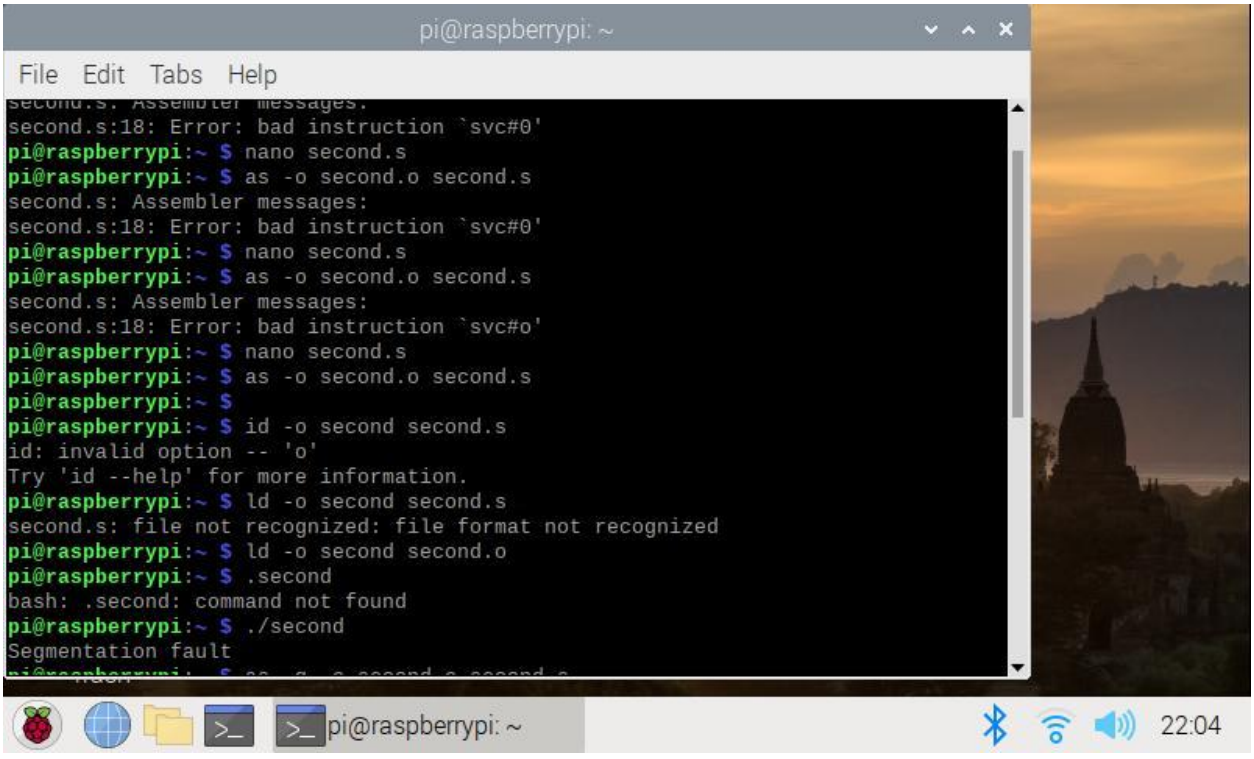
By now we can see the all cores working parallelly.

Section 3: ARM Assembly Programming

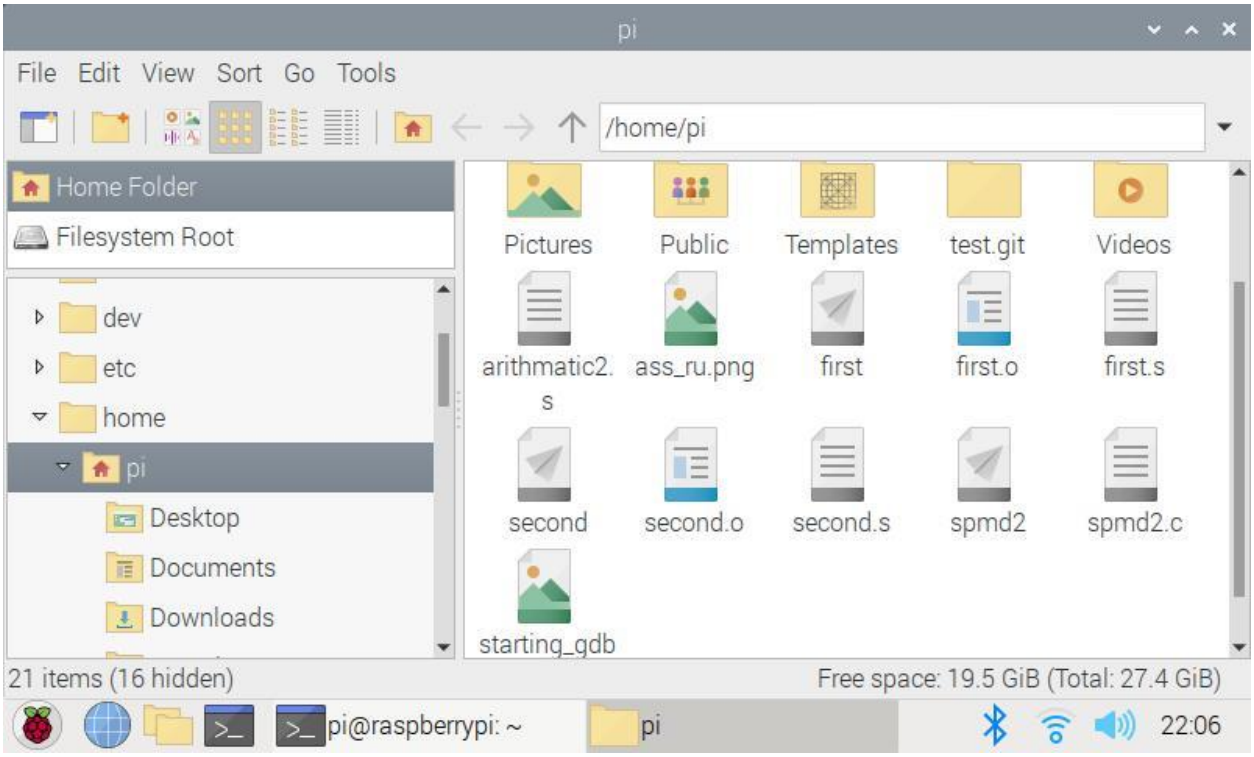
Part 1: Second Program



Second.s compiled using nano editor

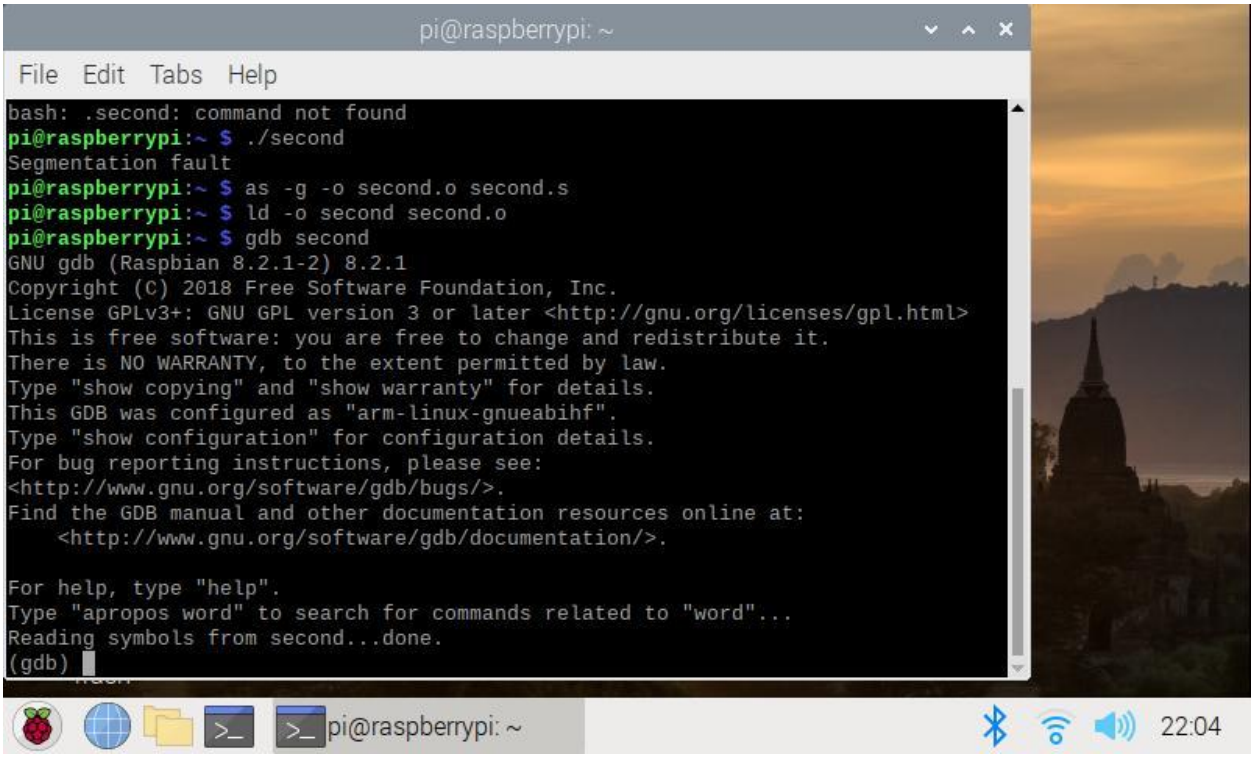


Assembled, linked, and ran the second.s file.

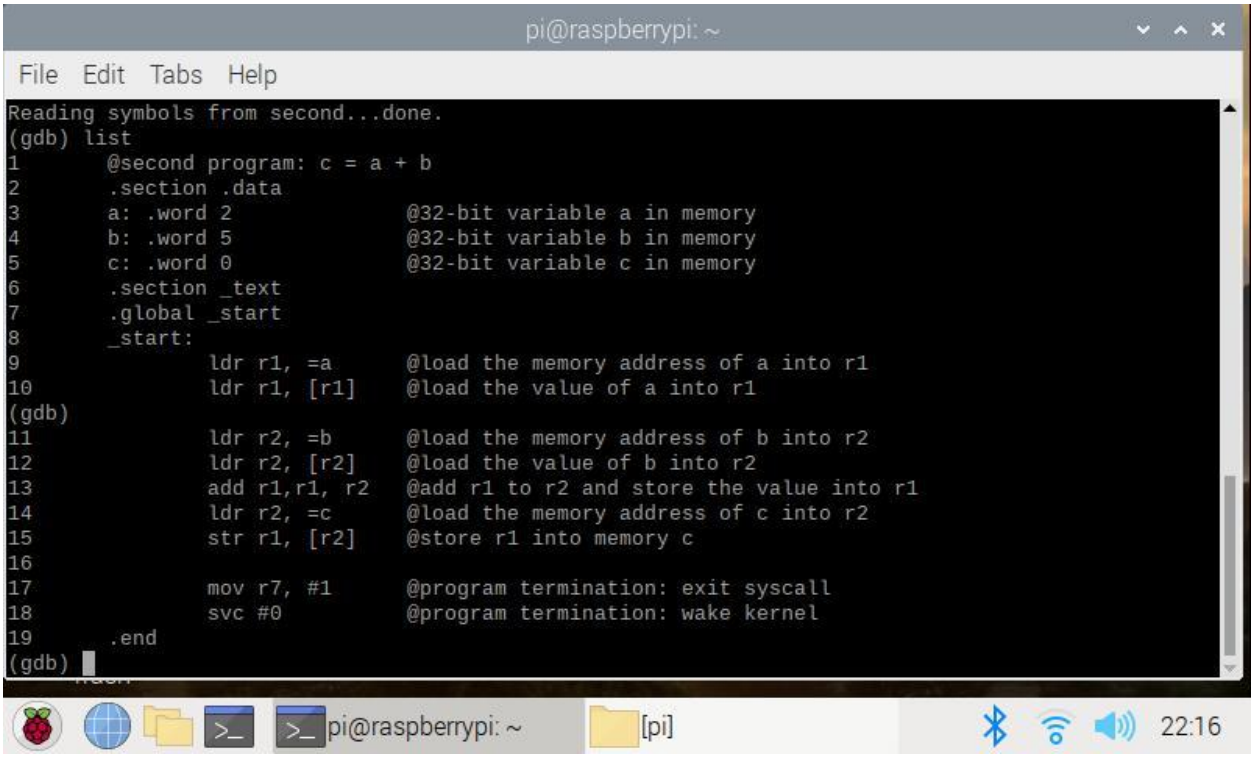


Created executable file

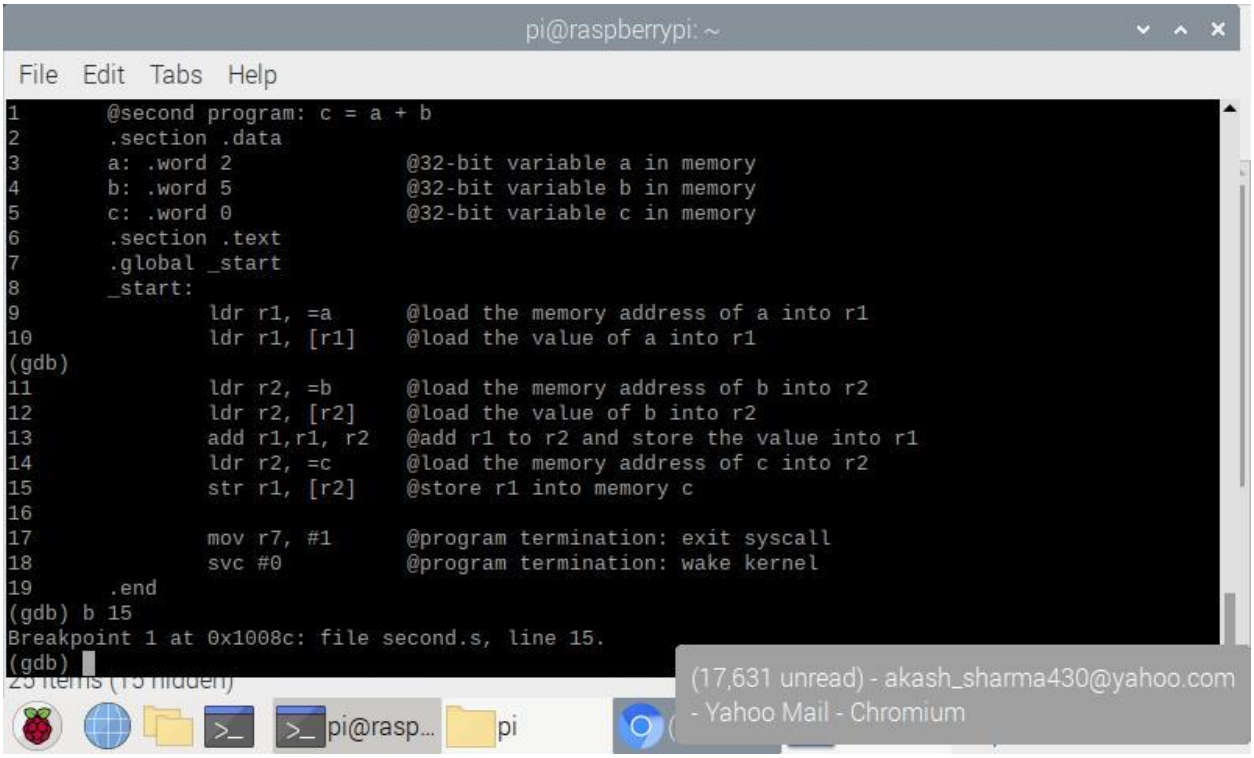
No output is seen as the output are not visible into the console but are inside the register of the microprocessor.



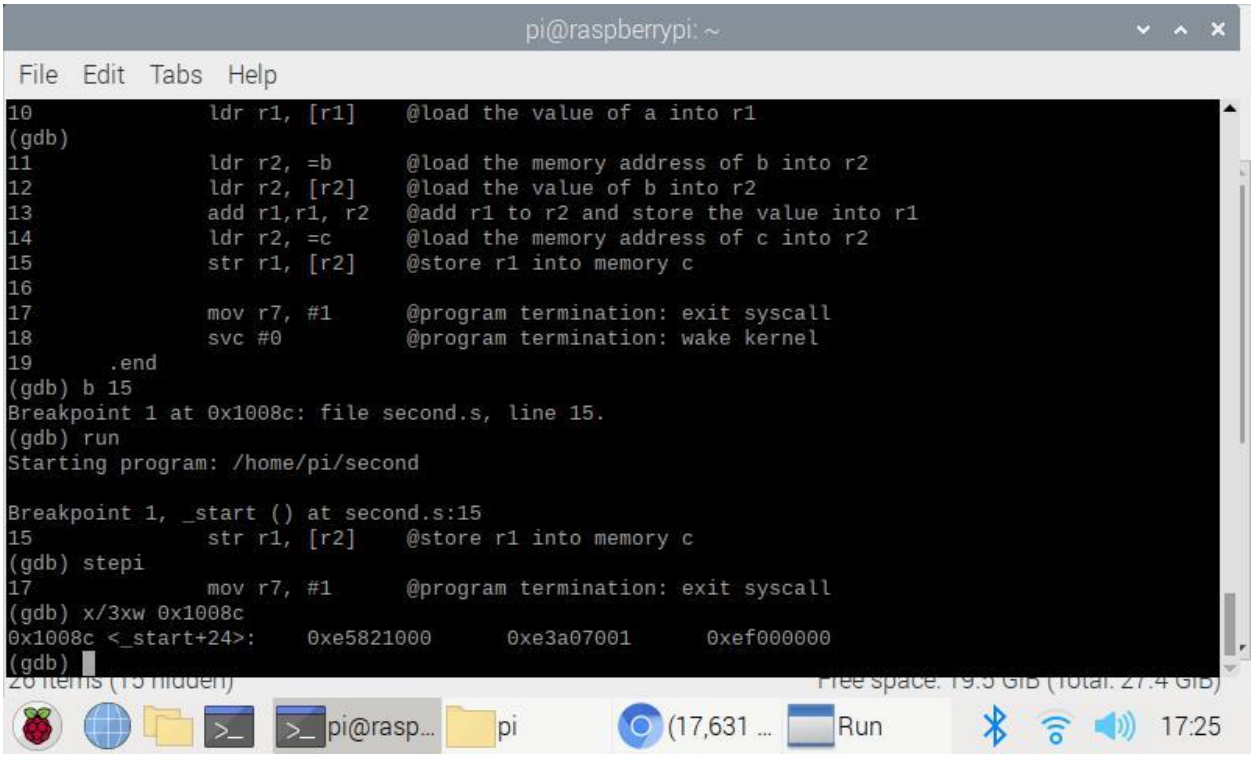
Started debugging using gdb



Using line number for accurate debugging.



Breakpoint set to line 15 using the command as shown above



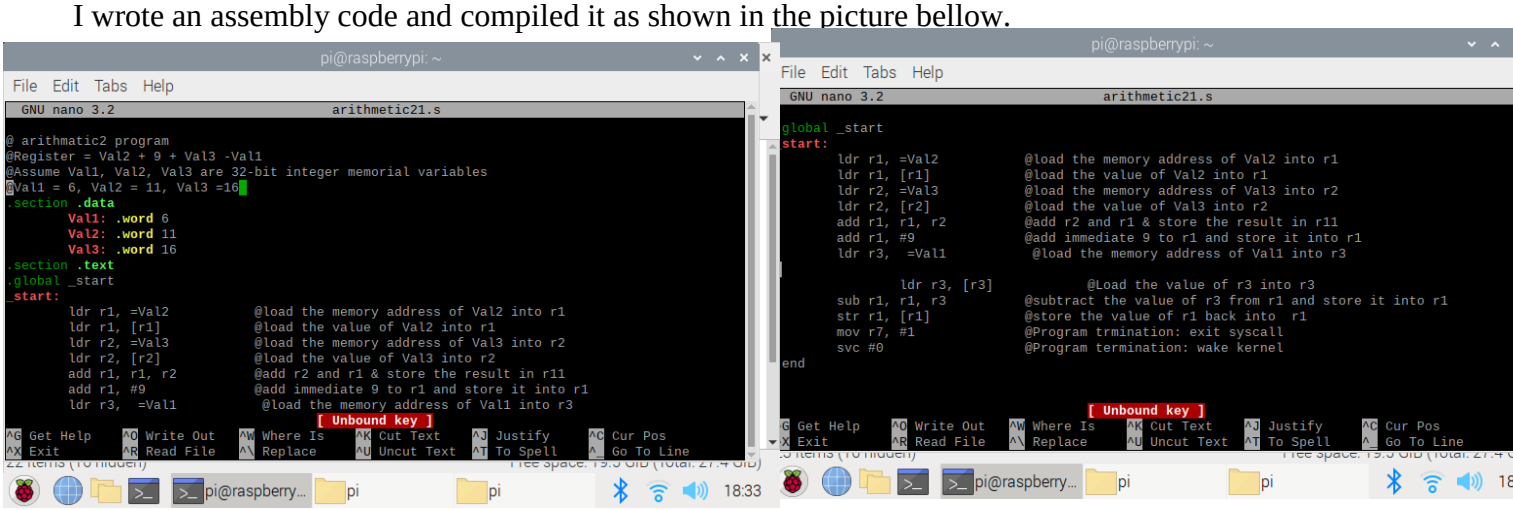
Using command x/3xw 0x1008c supplied, a combination of three words used to generate in hexadecimal. These words are seen in the above slide.

Part ll

The algorithm used in this project for assembly program is

Register = Val2 + 9 + Val3 – Val1 and the contents of the memory as given bellow in the data segment of the code

I wrote an assembly code and compiled it as shown in the picture bellow.



The executables have been created. Then I assembled, linked and run using the command line.

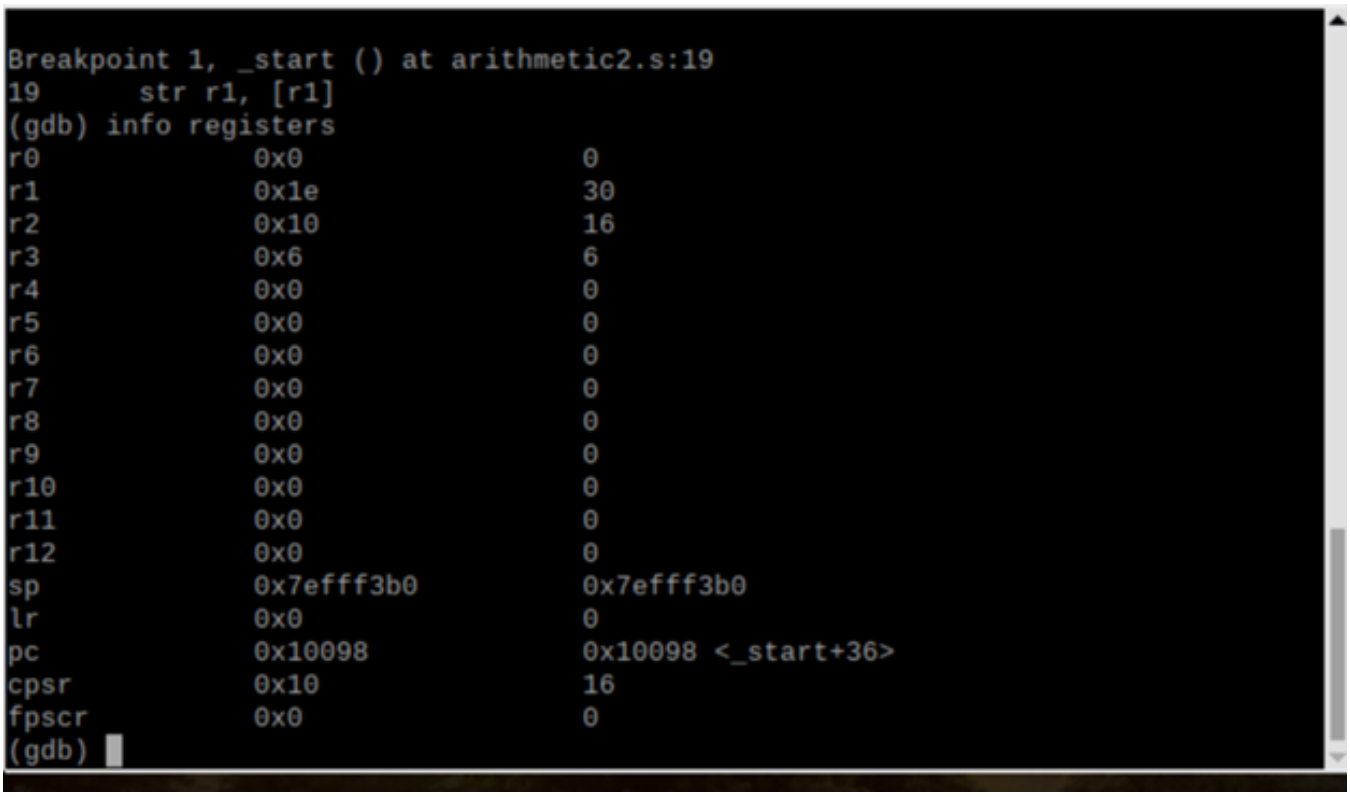
```
pi@raspberrypi:~ $ nano arithmetic2.s
pi@raspberrypi:~ $ as -o arithmetic2.o arithmetic2.s
pi@raspberrypi:~ $ ld -o arithmetic2  arithmetic2.o
pi@raspberrypi:~ $ ./ arithmetic2
bash: ./: Is a directory
pi@raspberrypi:~ $ as -g -o arithmetic2.o arithmetic2.s
pi@raspberrypi:~ $ ld -o arithmetic2  arithmetic2.o
pi@raspberrypi:~ $ gdb arithmetic2
GNU gdb (Raspbian 8.2.1-2) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabihf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
```

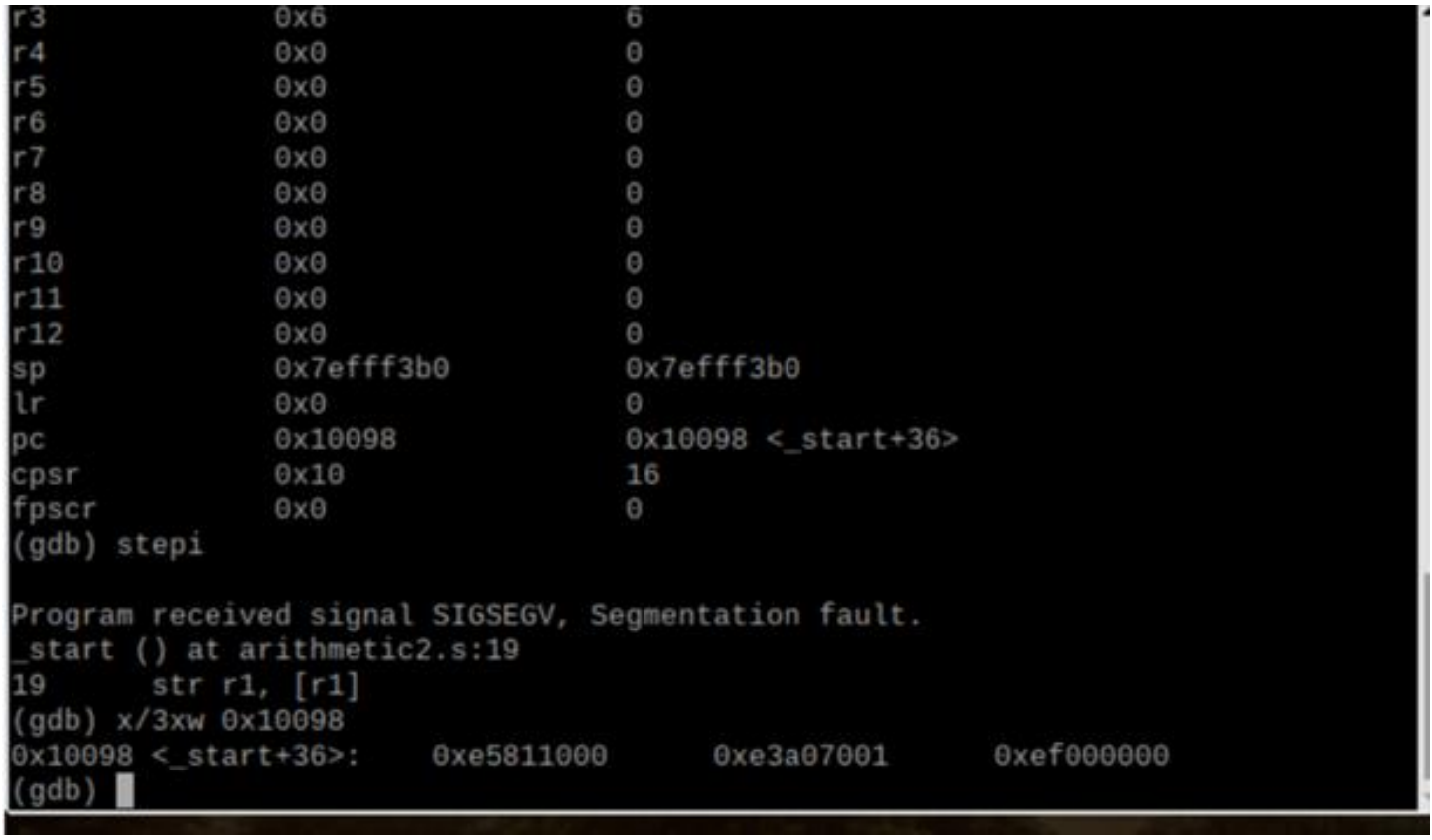
I then used gdb debugger to see the inside of the registers and memory

```
(gdb) list
1       @ arithmetic program : val2 + 9 + val3 - val1
2
3       .section .data
4       val1:.word 6
5       val2:.word 11
6       val3:.word 16
7       .section .text
8       .globl  _start
9       _start:
10      ldr r1, =val2
(gdb)
11      ldr r1, [r1]
12      ldr r2, =val3
13      ldr r2, [r2]
14      add r1, r1, r2
15      add r1, r1, #9
16      ldr r3, =val1
17      ldr r3, [r3]
18      Sub r1, r1, r3
19      str r1, [r1]
20      Mov r7, #1
(gdb)
```

Used list command for better debugging and studied inside the registers



Then I set the breakpoint at line 19 as shown above and used *info registe*r to view into the registers. I also used *stepi* command to get into the next step while studying the memory and registers value changing stepwise.



The command x/3xw 0x10098 as shown above has been used to view three words in hexadecimal from the code.

Those three words are shown in the above screenshot at the last line.

I then manually checked the arithmetic into my calculator and found that 30 corresponds to the final value of the register. Also each values of the assigned variable corresponded to the memory of the CPU.

Vedansi Parsana

Parallel Programming Skills

A) Foundation

The components on the raspberry PI B+ are:

Two USB ports, 2 power ports, a camera, HDMI port, ethernet Controller, ethernet port, CPU/RAM, and a display port.

Raspberry PI's B+ CPU have:

The Raspberry PI's B+ CPU has 4 cores.

The three main differences between INTEL x86 (CISC) and ARM Raspberry PI (RISC) are:

1) The CISC has a larger and more feature rich instruction set, which allows several complex instructions to access memory with more operations.
2) CISC uses little-endian format for storage.
3) CISC has more addressing modes but less registers than RISC.

1) RISC has a simplified instruction set (Which helps to execute the program quickly.), with more general-purpose registers than CISC.

2) RISC's instructions only operate on registers and uses to Load and Store for memory access.

3) RISC is stored with BI-endian (it can switch between small and big endian).

The difference between sequential and parallel computation and its practical significance of each are:

As we know software is written for serial computation. The problem is categorized into discrete series of instructions and executed on a single processor and it allows only one instruction to execute at a time. The practical significance of sequential computing is the ease of writing instructions.

Parallel computing is the instantaneous use of multiple compute resources to solve a computational problem. The problem is categorized into series of instructions that can be executed parallelly in different processors. The practical significance of parallel computing is the speed at which instructions are executed.

The basic form of data and task parallelism in computational problems are:

Data parallelism refers to a broad category of parallelism, in which same computation (tasks) is applied to multiple data items in different cores.

Task Parallelism is used where parallelism is organized around the functions to be performed rather than performing on the data around it. In simple words, it allows to compute one task on one core and another task on another core.

The differences between processes and threads are:

Processes means when the program runs, they do not share memory with each other, and they operate one per core. Whereas threads are easy processes that allow executables to be decomposed into independent parts and they share a common memory with the process to which they belong.

OpenMP and OpenMP pragmas means:

OpenMP is an application programming interface which uses an implied multi-platform model in which the library handles thread creation and management and makes the programmer's task simpler.

OpenMP pragmas is the compilers directive that enables the compiler to generate threaded code.

Applications that benefit from multi-core are:

Applications that benefit from multicore usage include are not limited to database servers, scientific applications (CAD/CAM), multimedia application, and web servers.

Multicore is used because:

Multi cores can operate running multiple processes at the time while single core can only operate one process at the time, which reduces the time to complete the task and increases the output of the system. Speeding up the clock frequencies in single core is difficult due to heat problems, maximum speed of light, and costs. Lastly, many new applications are multithreaded to utilize multicores.

B) Parallel Programming Basics

→This is a C program to get familiar with special kind of editions in OpenMP, which allow to run the parts of a given program on multiple threads of a multicore machine, which is shown in the following program.

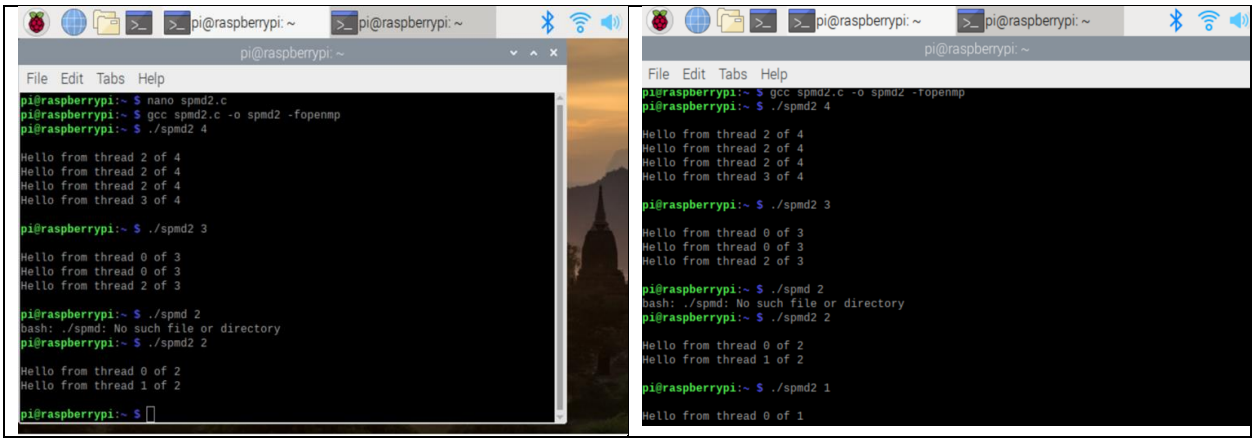
→When I first ran this example did not run as expected it showed that the first iteration of the code was that int id was declared in main method. The id appeared more than once in the output or when program was executed.

→As all the cores share the same memory, it cannot have multiple variables with the same id.

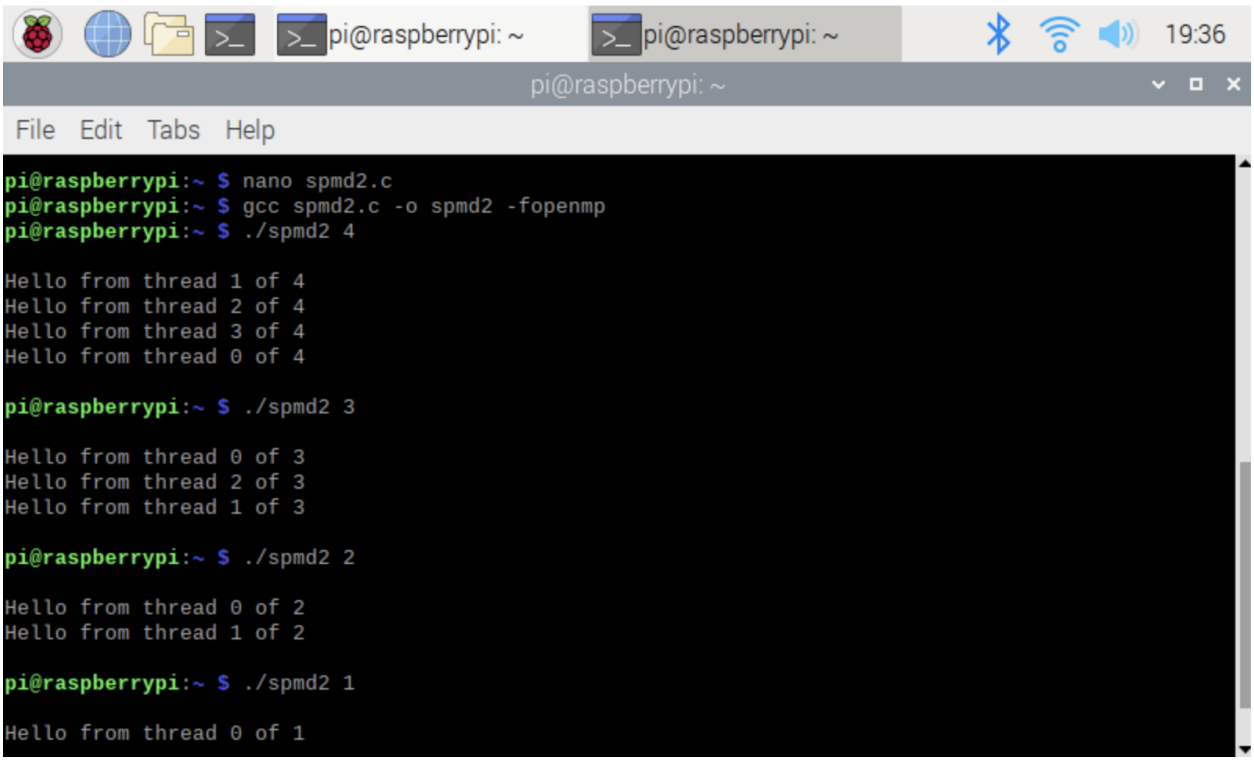
→ I then referred to the given instructions to solve the program. After it was fixed the difficult lines with full variable declarations were done and it executed successfully as expected.

Parallel Programming Skills

The program when ran for the first time.



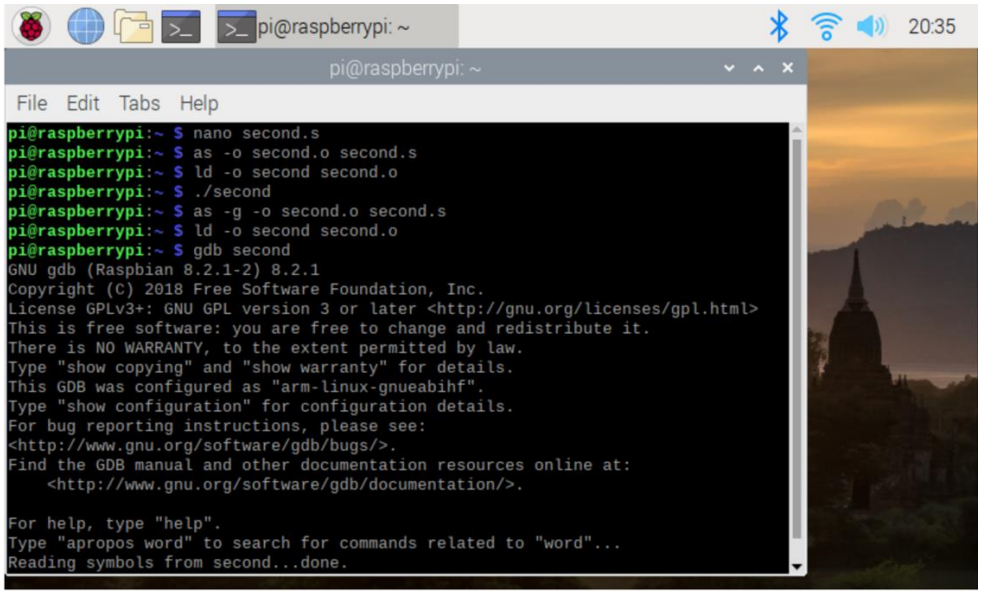
After fixing the problem:

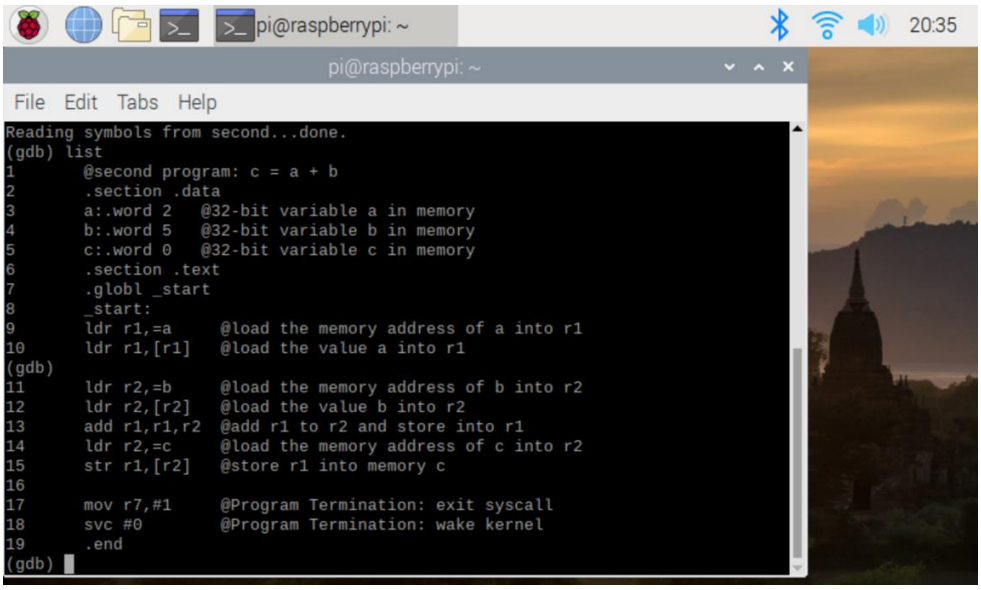


Section 3: ARM Assembly Programming

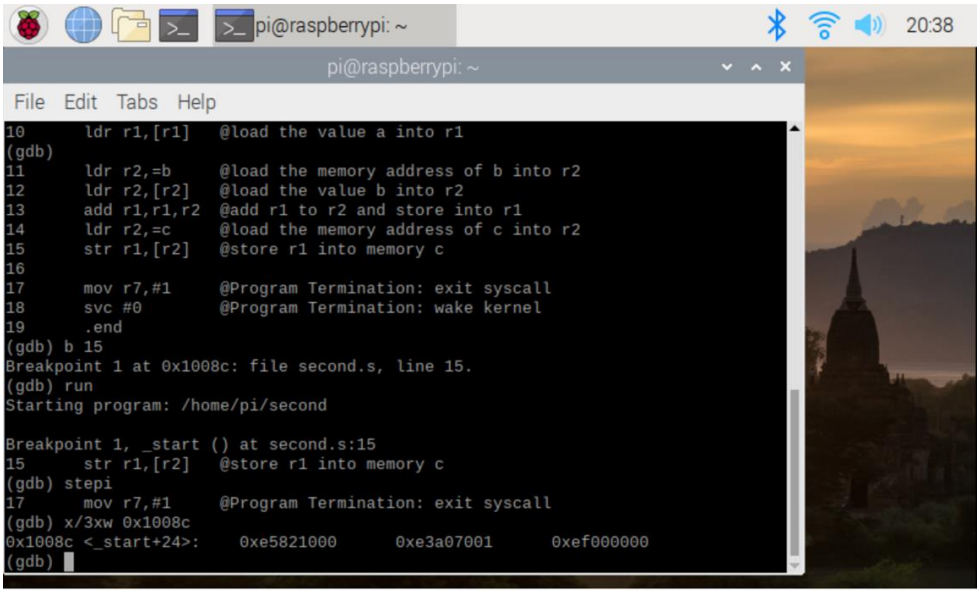
Part 1: Second Program

First, I composed the file,
then I assembled and linked it in order the receive an executable file. However, there was no output.
The entire program takes place in the registry, so we did not expect to receive any output.
Next, I practiced utilizing the GDB (GNU DeBugger),





Then I set a breakpoint to see how the program completing took place. To stop debugging I get a breakpoint at “b 15”as shown in the below image. With the help of run I executed the program. To execute the program step by step I used “stepi” command as shown in the image below.

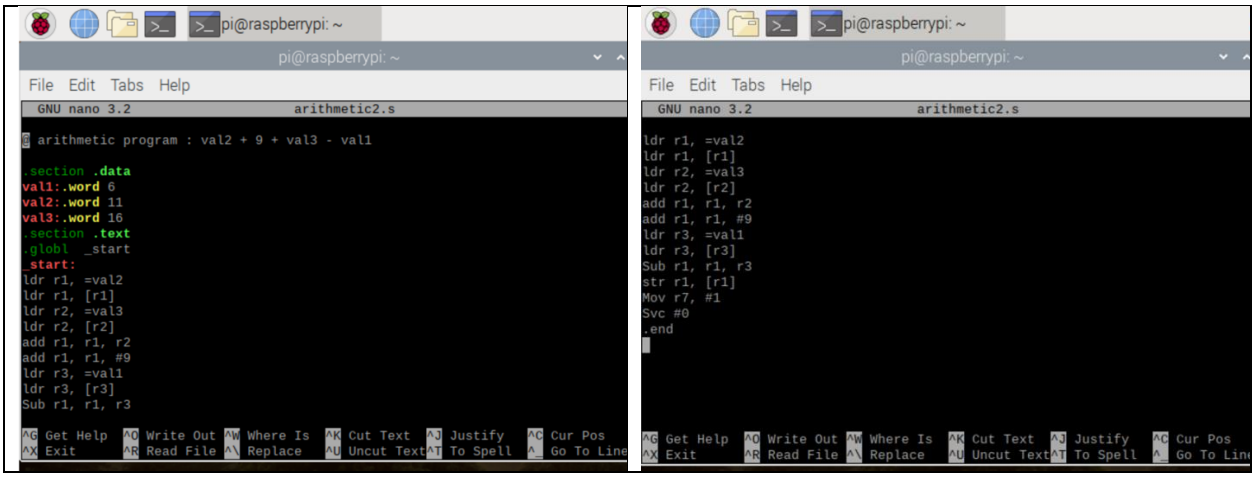


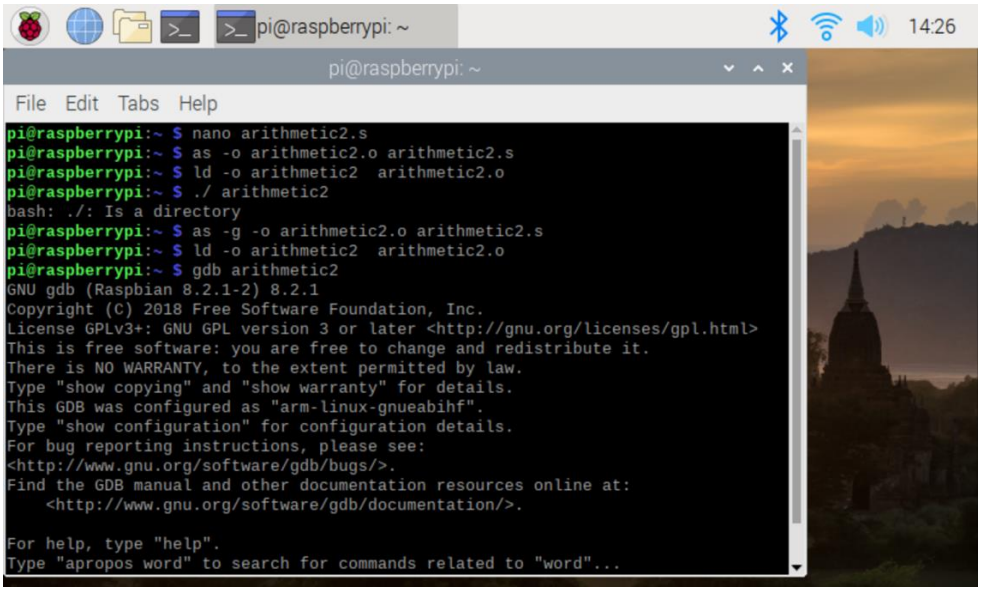
This command x/3xw 0x100990 was used to generate three words in hexadecimal: 0xe5821000, 0xe3a07001, 0xef000000.

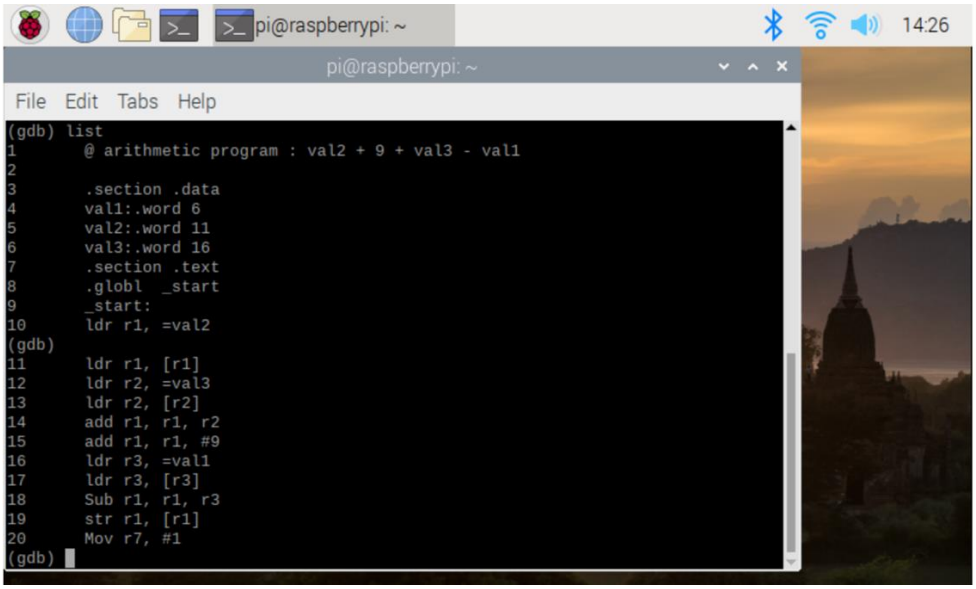
Part 2: Calculate the Expression

I had to write the program for the following: Register = val2 + 9 + val3 – val1, where val1=6, val2=11, and val3=16.
First, I composed the file,

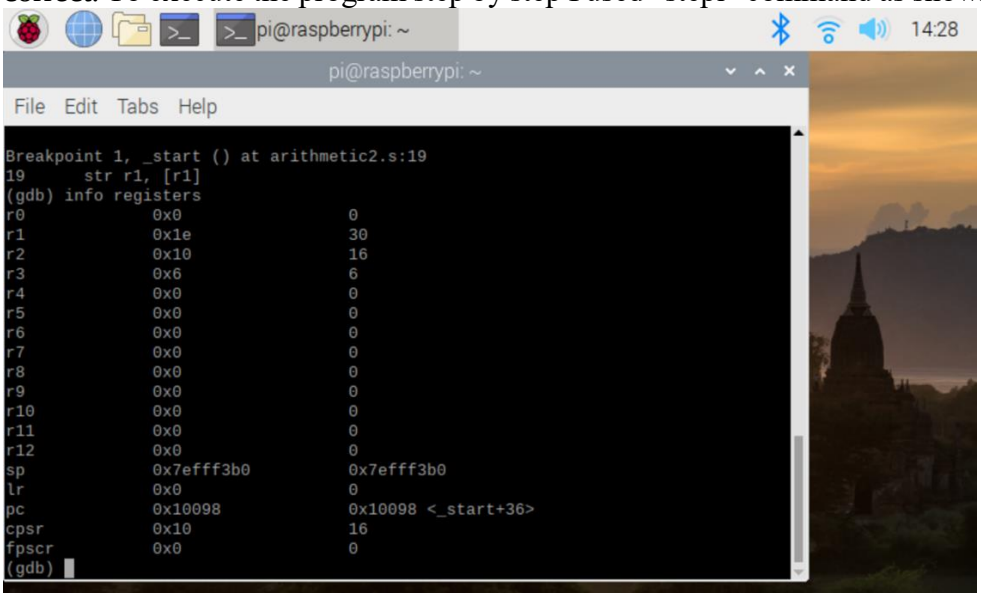
then I assembled and linked it in order the receive an executable file. However, there was no output. The entire program takes place in the registry, so I did not expect to receive any output.
Next, I practiced utilizing the GDB (GNU DeBugger) as I did in the above example.



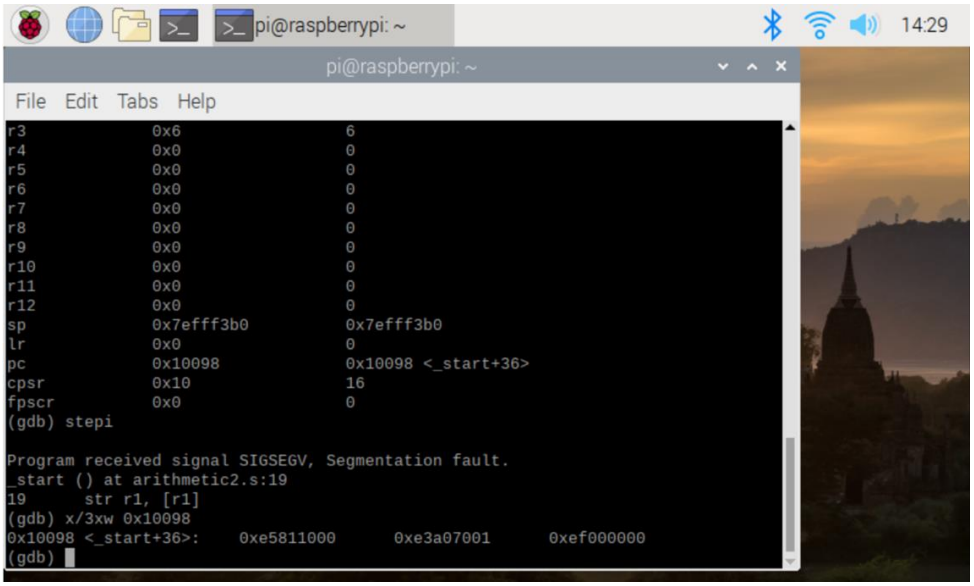




As I did in the earlier example, I thent set a breakpoint to see how the program completing took place. To stop debugging I get a breakpoint at “b 19”as shown in the below image. With the help of run I executed the program. I also used the command “info registers” to make sure my code is correct. To execute the program step by step I used “stepi” command as shown in the image below.



This command x/3xw 0x10098 was used to generate three words in hexadecimal: 0xe5811000, 0xe3a07001, 0xef000000.



→To make sure the code and the answer for the equation is correct I compared the values in the register and all of them were correct.

→Register 1 has the assigned value because the answer for the equation: val2 + 9 + val3 – val1 is 30. Also, Registers 1 to 3 has the values 30, 16, and 6 which were the values assigned to them.

Zoe Kosmicki

Parallel Programming Skills

1) The raspberryPI B+ has a quadcore CPU that includes RAM, a microSD chip reader, and USB ports.

2) It has four cores.

3) ARM has a simplified ISA, meaning it can sometimes be faster than x86 processors. ARM also has different modes for it's instructions, while x86 processors only have one. X86 processors also use little endian, while current ARM processor scan switch between little and big endian.

4) Parallel computing is different from sequential computing because it can execute many different tasks at once, whereas sequential can only do one task at a time. As such, parallel computing can be faster, but there needs to be no dependencies between the various threads that are running, or else sequential computing must be done.

5) The basic form of data parallelism and task parallelism is that they're both types of parallel computing, but data parallelism focuses on finishing one task on lots of subsets

of data, while task parallelism is oriented around doing different tasks on various sets of data.

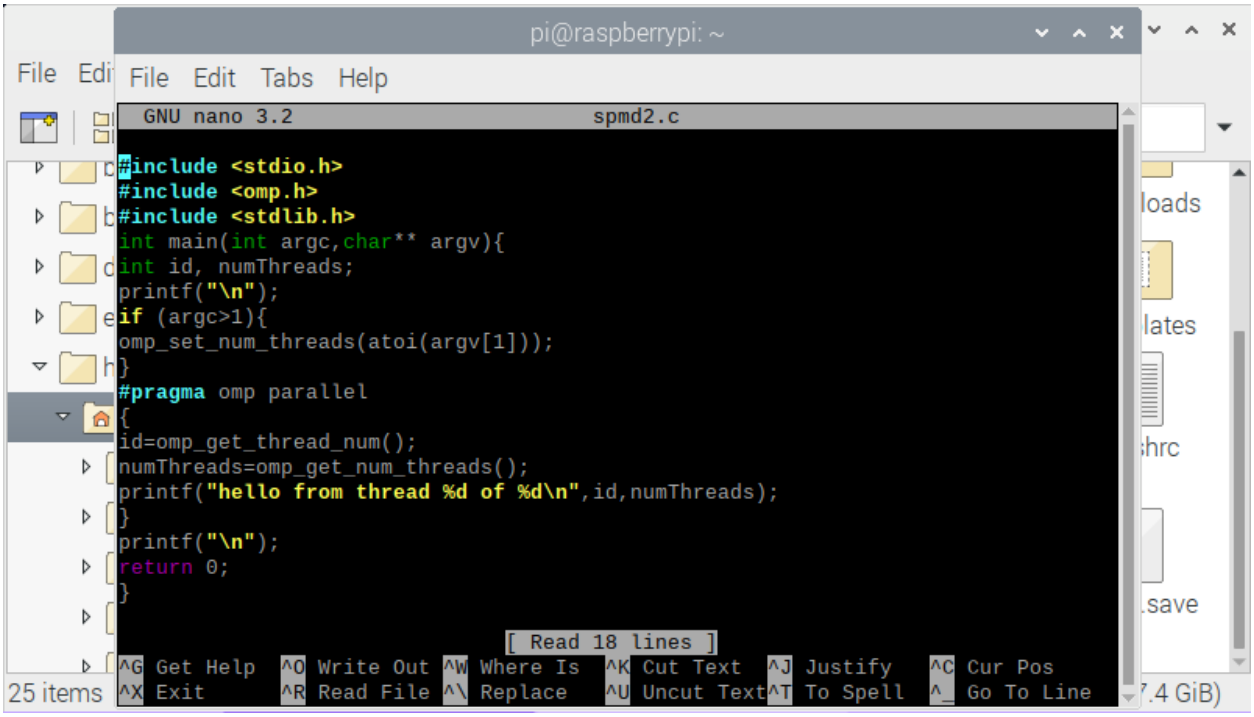
6) Threads are a type of process that can be broken down into smaller steps, while processes are the entire program to be executed. A process can be made of many threads, but a thread cannot contain processes.

7) OpenMP is an interface introduced in the 90's that supports parallel processing and computing. OpenMP pragmas are compiler directives that allow for multithreaded code to be run and compiled.

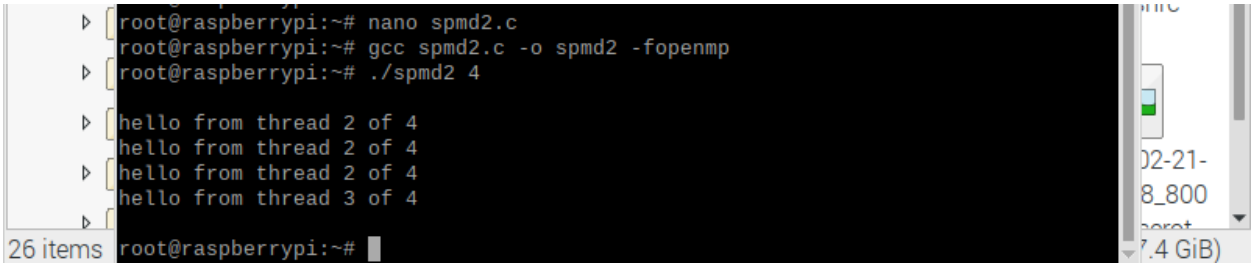
8) Web servers, compilers, multimedia applications, web browsers.

9) Multicore is generally faster than single, lets the user do multiple activities at once, multicore CPUs can be more energy efficient than single core, and it takes up less space on a circuit board since the cores are packaged into one CPU.

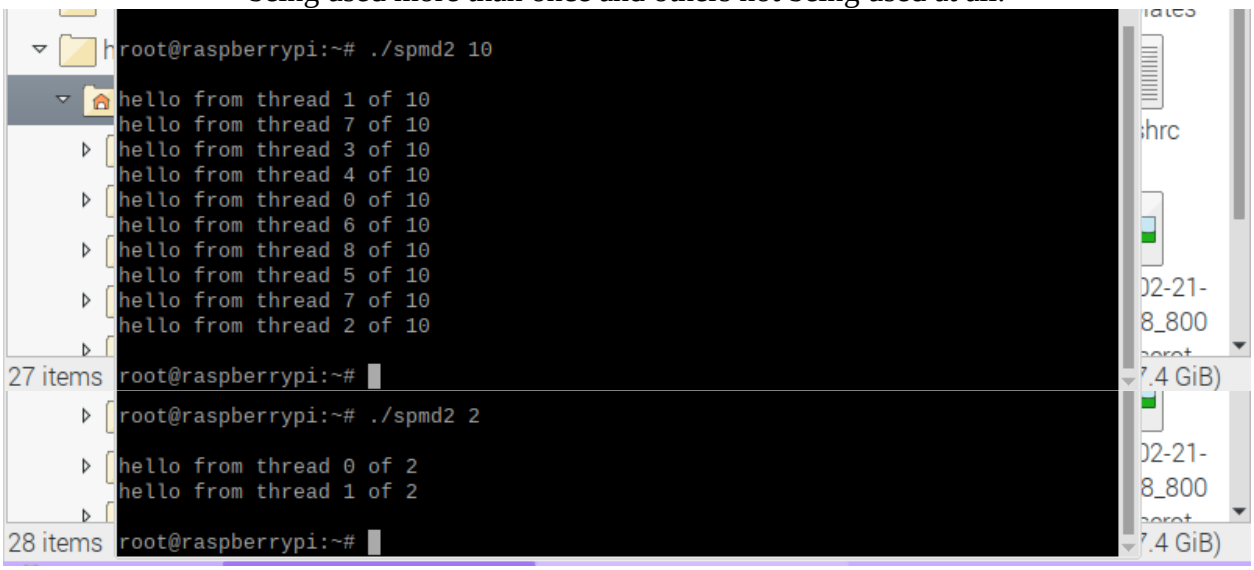
Parallel Programming Task



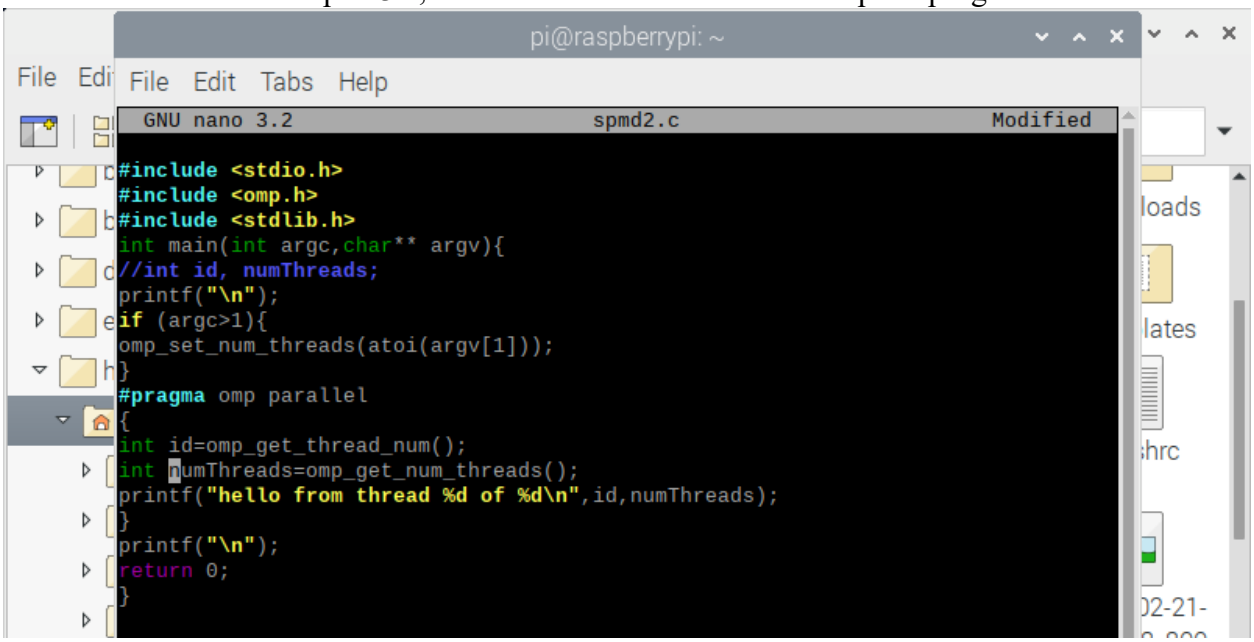
I began by typing out the program as listed in the instructions, checking over it to be sure I didn't mistype anything.



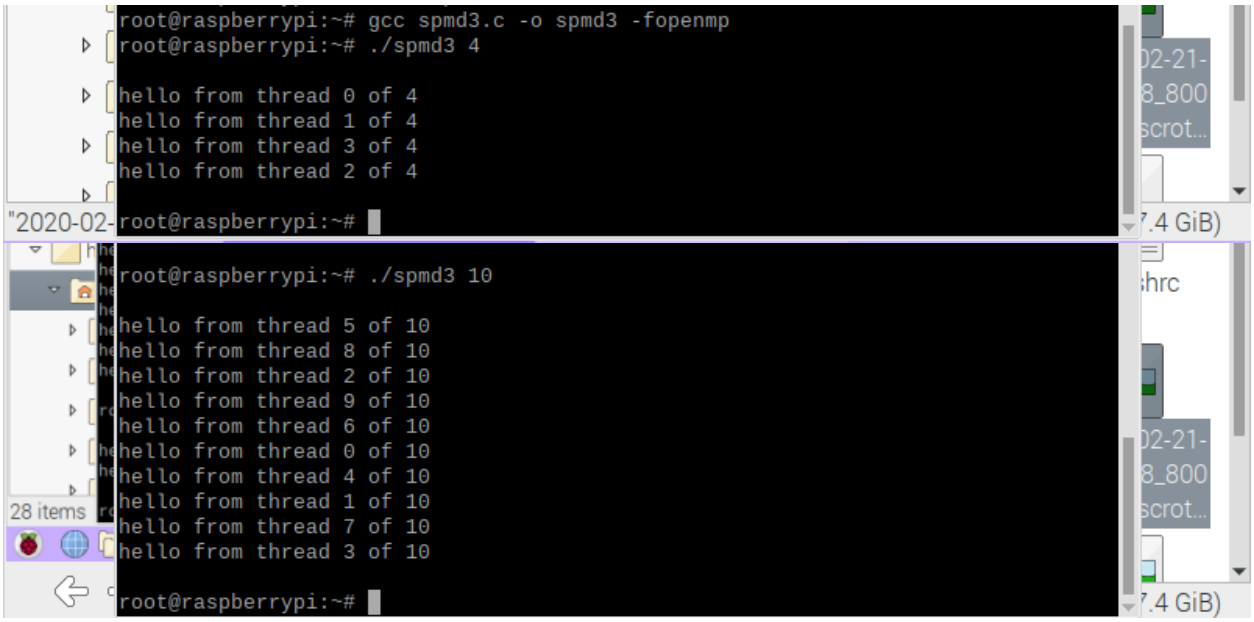
Next, I ran the program using the dot slash command and tested the output with 4 threads. I tested again with 10 and 2 threads. I noticed that not all threads were being used, with some being used more than once and others not being used at all.



Next, I altered the code to the specifications in the instructions. I decided to rename this altered version spmd3.c, so I wouldn't confuse it with the prior program.



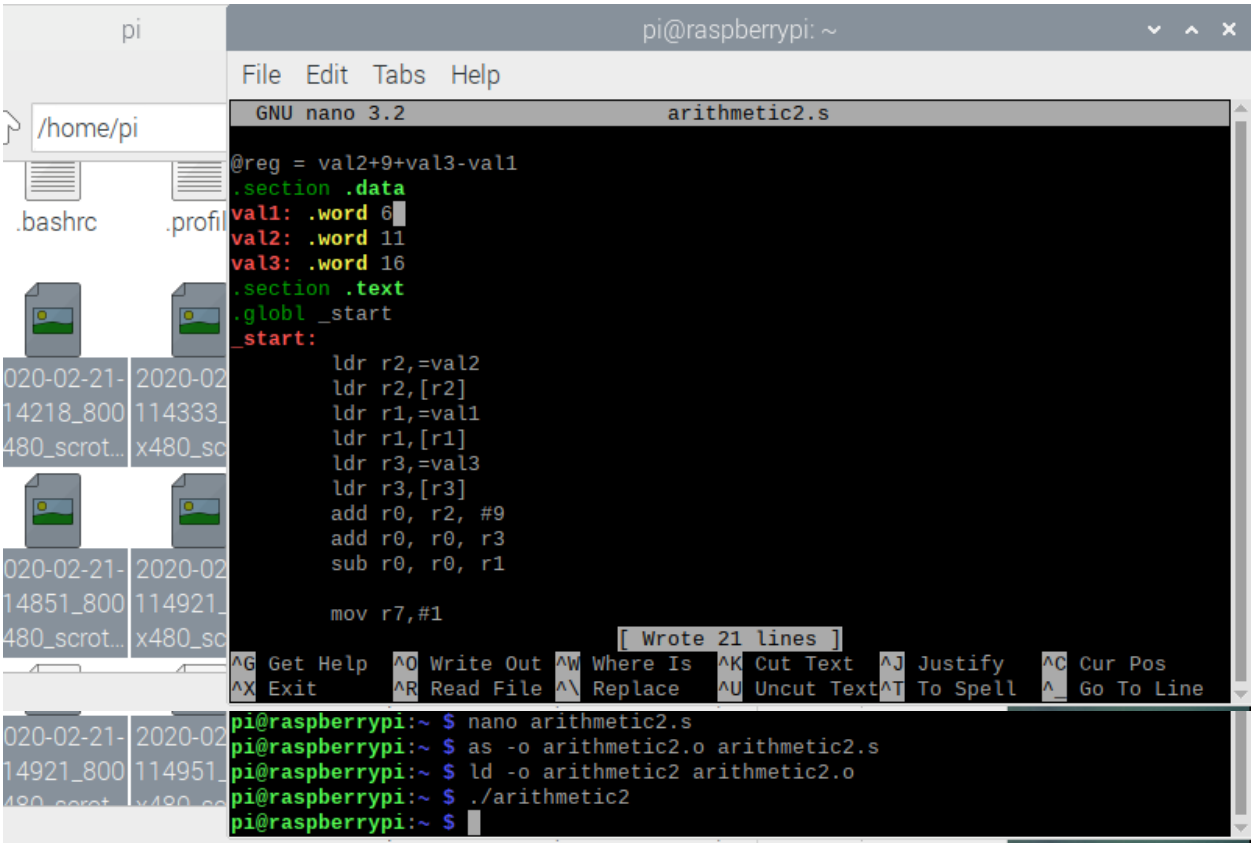
I linked the file and ran it, and then it began using each thread once and only once. I again tested it with the inputs 4, 10, and 2, just to be sure my outputs were consistent across all initial inputs.



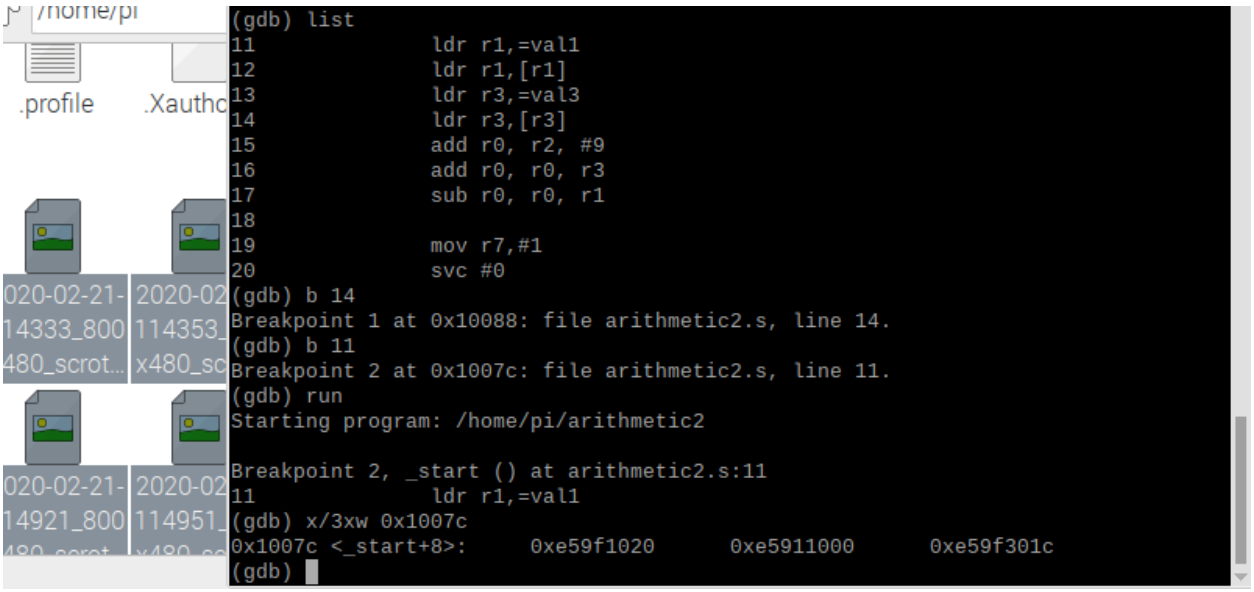
They were all consistent, confirming the changes fixed the initial problem.

ARM Assembly Programming Task

Using second.s as a template, I wrote up a version of arithmetic2 that loaded the memory address and then values of var1, var2, and var3 into r1, r2, and r3 respectively. Then I used r0 to store all the arithmetic and finished values as they were being done.

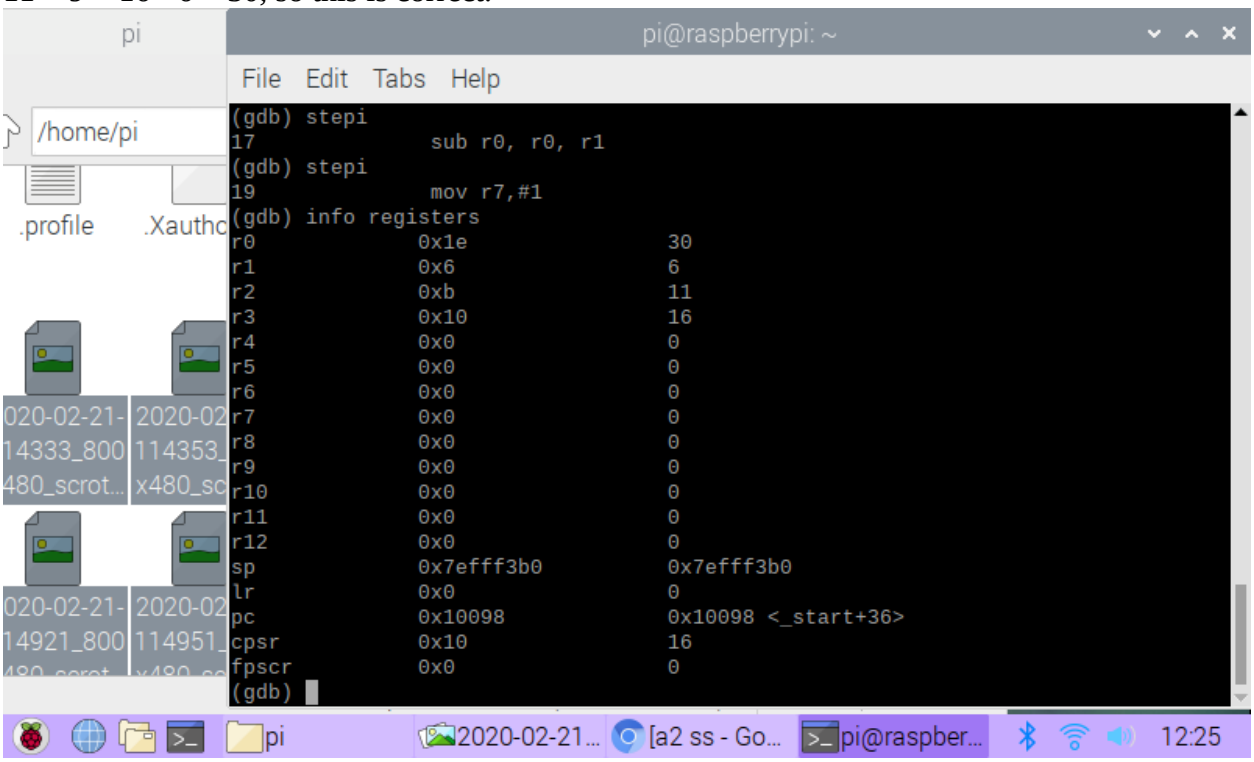


After linking arithmetic2, I ran it with the dot slash instruction, and nothing happened because no output is specified. Next, I ran arithmetic2 with the debugger, adding breakpoints at lines 11 and 14.



I checked the memory for three hexadecimal words at the address specified at line 11, and above is what I got. After checking the memory, I used the stepi command to walk through lines 12 to 19, and checked the registers to make sure the program executed correctly.

11 + 9 + 16 - 6 = 30, so this is correct.



Appendix

LINKS

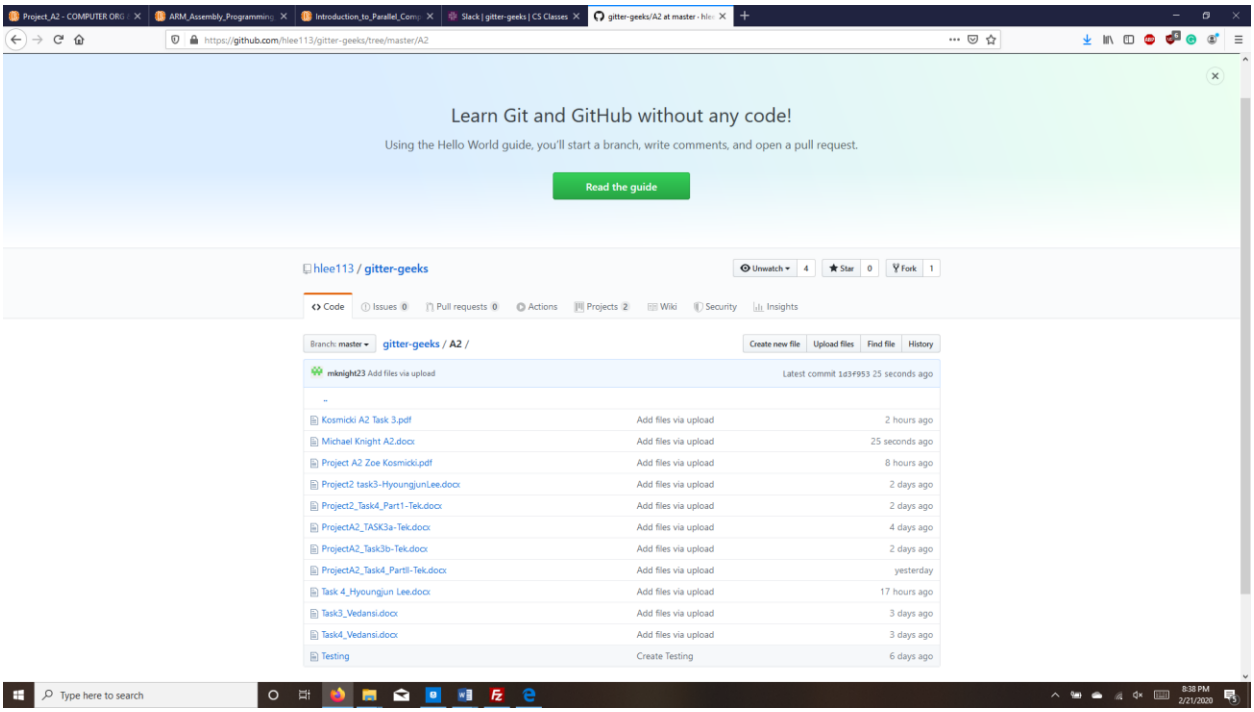
GitHub: https://github.com/hlee113/gitter-geeks/tree/master/A2

YouTube: https://www.youtube.com/watch?v=w85mGA-7ugs&feature=youtu.be

Slack: https://app.slack.com/client/TN46XP41L/GSUL94RJ5

SCREENSHOTS:

GitHub:



Planning and Scheduling Timetable:

Names	Email	Task	Dependencies	Duration	Due Date	Notes
Hyoungjun Lee	hlee113@student.gsu.edu	Programming with Raseberry pi and Updating Github with new tasks	None	3 hours	2/19/20	100%
Michael Knight	mknight23@student.gsu.edu	Programming with Raseberry pi and Assigning tasks and Writing the final report and turning it in	Needs Everyone's submissions to combine into a report for submitting.	5 hours	2/21/20	I am very proud of my group and what we accomplished. Everything ran smoothly because everyone was ontime or early with their submission.
Tek Acharya	tacharya2@student.gsu.edu	Programming with Raseberry pi and uploading video to youtube	None	3 hours	2/19/20	100%
Vedansi Parsana	vparsana1@student.gsu.edu	Programming with Raseberry pi	None	3 hours	2/19/20	100%
Zoe Kosmicki (coordinator)	zkosmicki1@student.gsu.edu	Programming with Raseberry pi and Video Recording/Editing	None	4 hours	2/20/20	100%