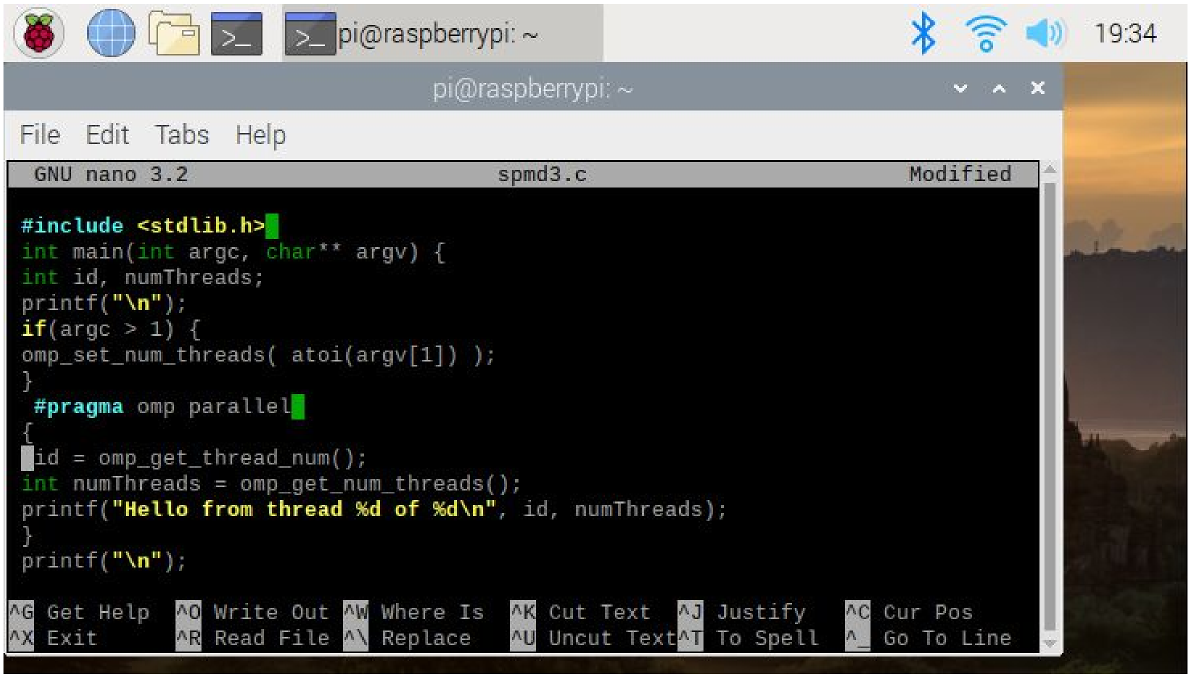
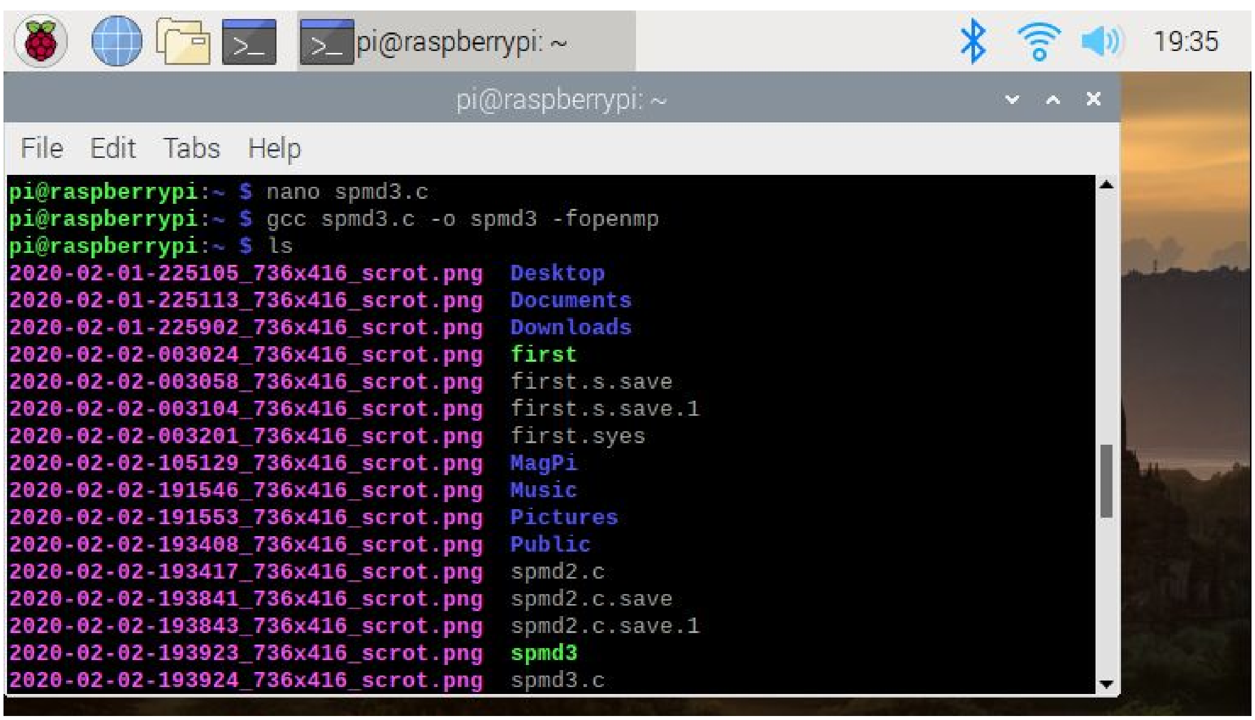
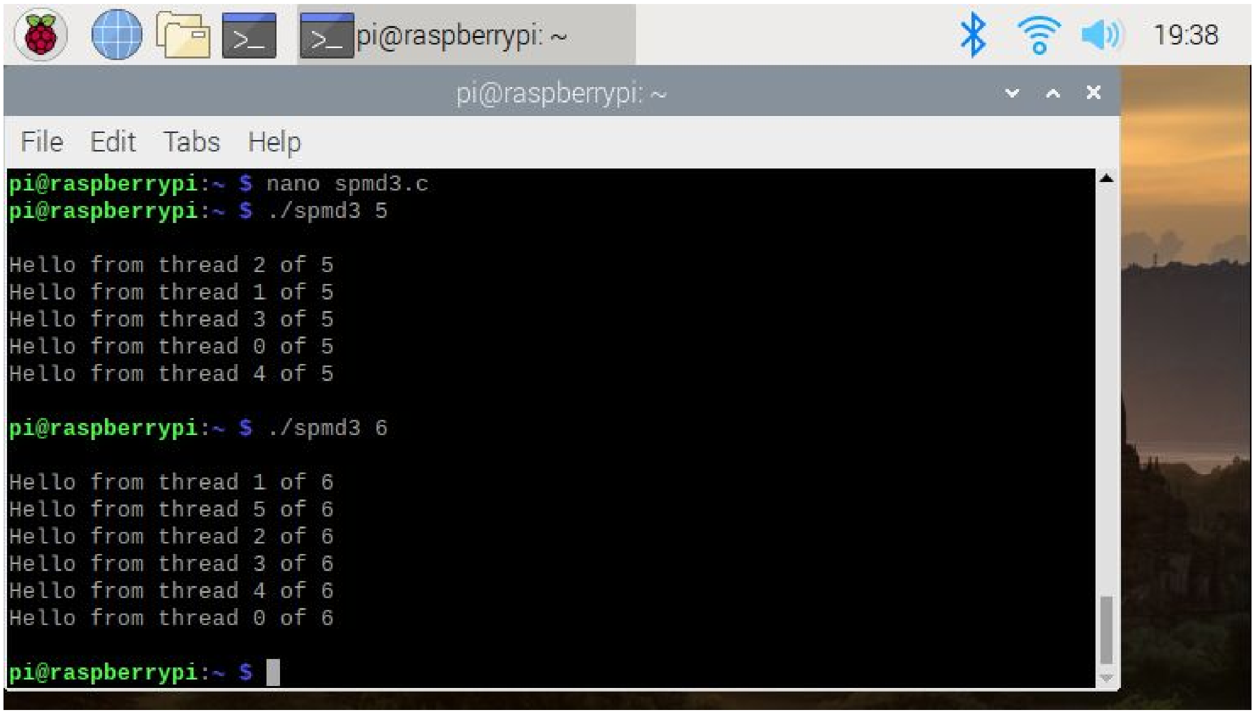
PARALLEL PROGRAMMING



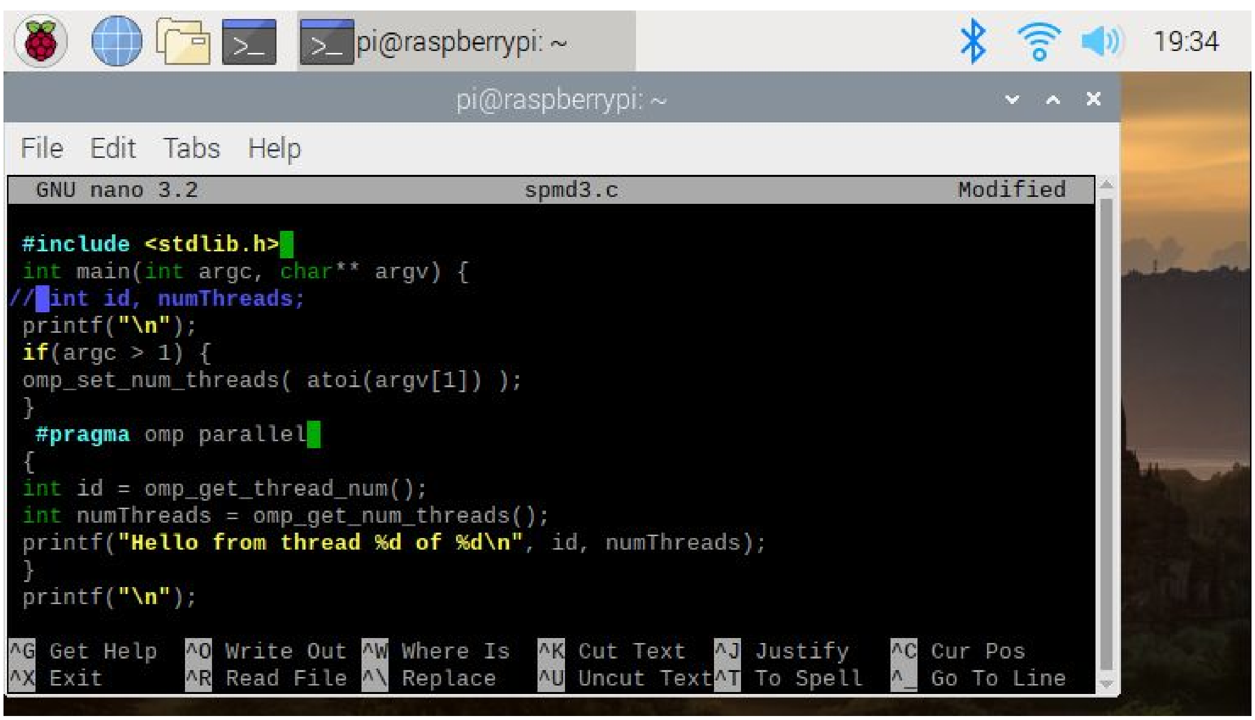
This is the original source code for spmd3.c. I saved my file differently compared to the guidelines, that is why I have spmd3 instead of spmd2.c.



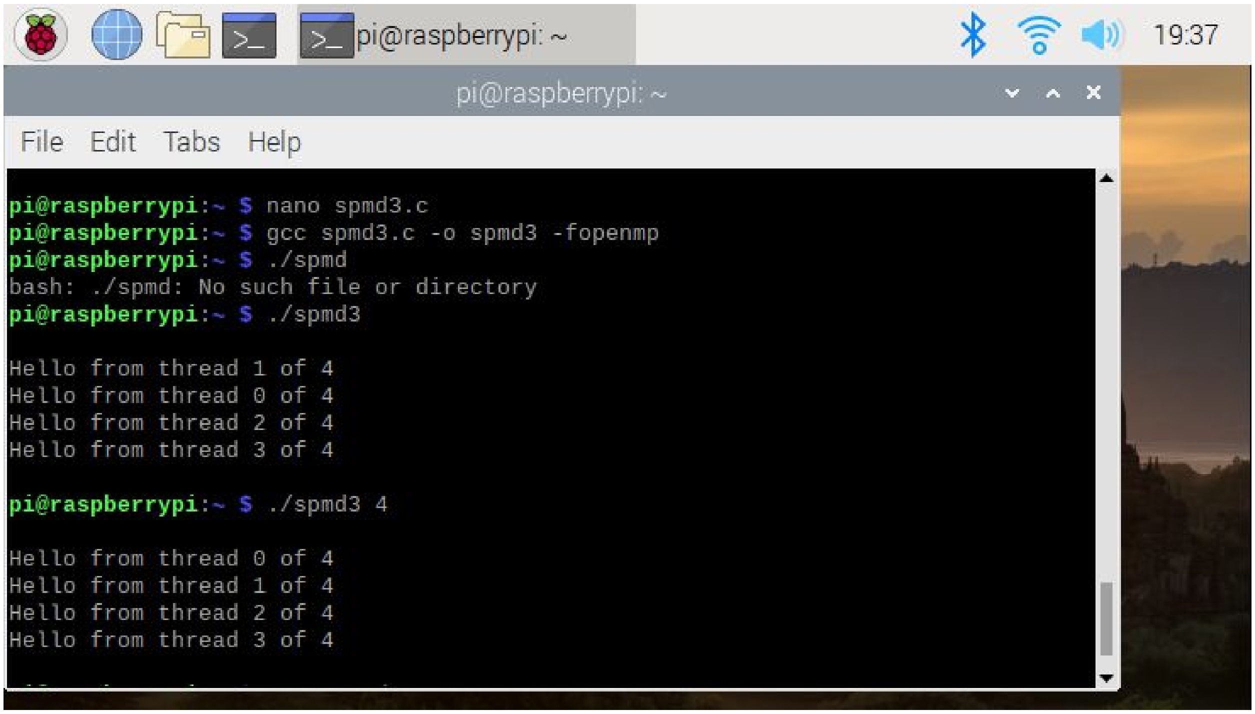
In this screenshot, I used ls to check the current directories and make sure it is saved.



The original source code gives an output of thread 2, thread 3, thread 2, thread 2. This is reasonable because the source code is designed to assign the id variables which is why the values of the thread ids share.

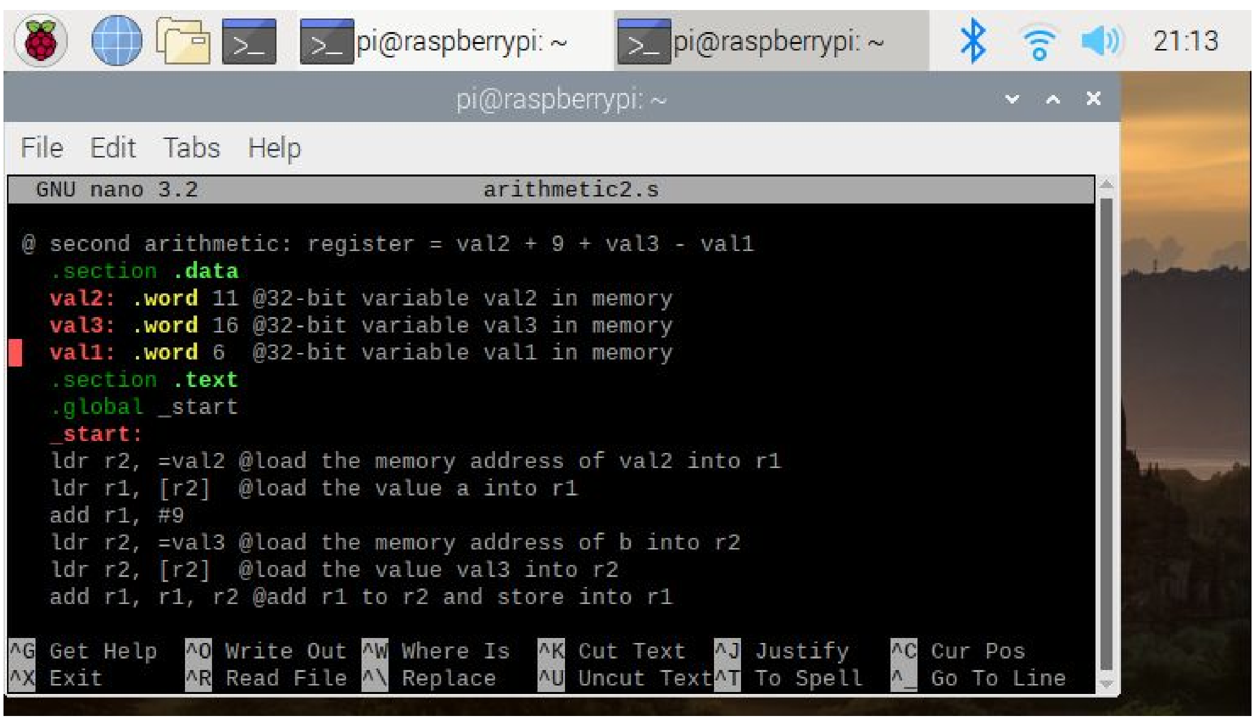


In order to make the threads align, the revised source code should comment the declaration of the id and numThreads (line 5). We individually declare the id and numThreads on lines 12 and 13.

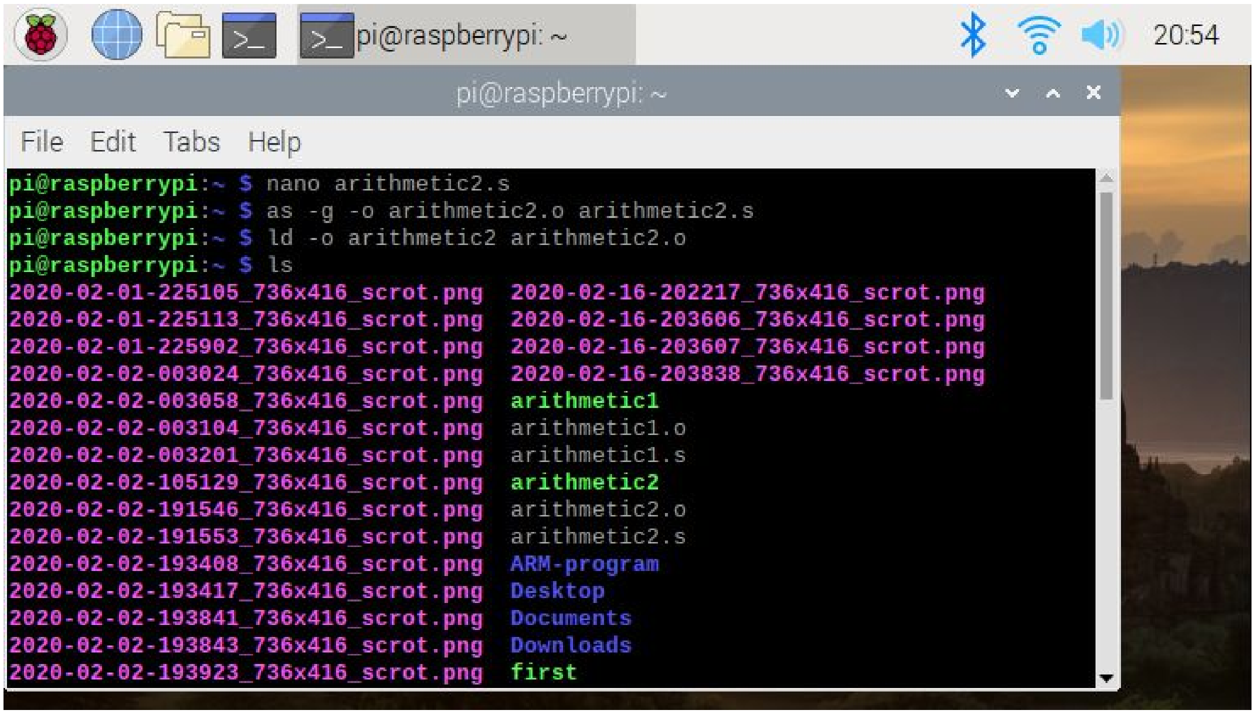


After editing the source code, we can see that the threads are aligned which was the goal where all 4 threads are shown and each have different ids and are aligned.

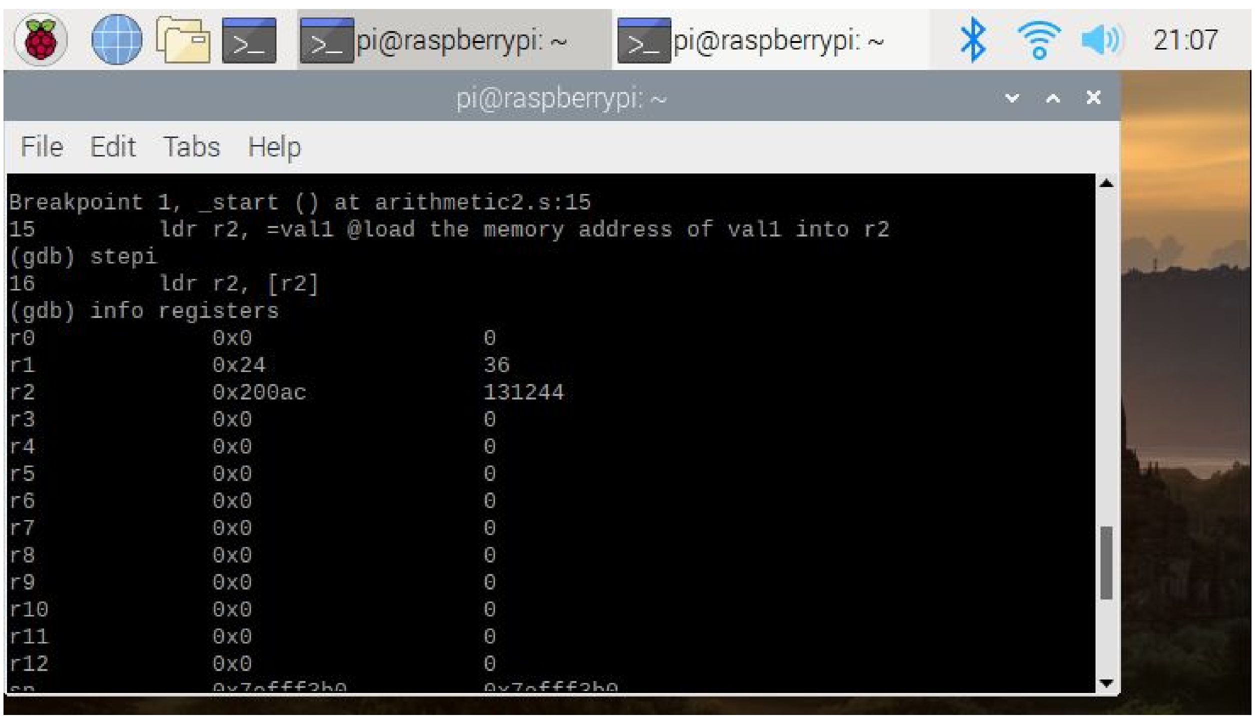
ARM PROGRAMMING



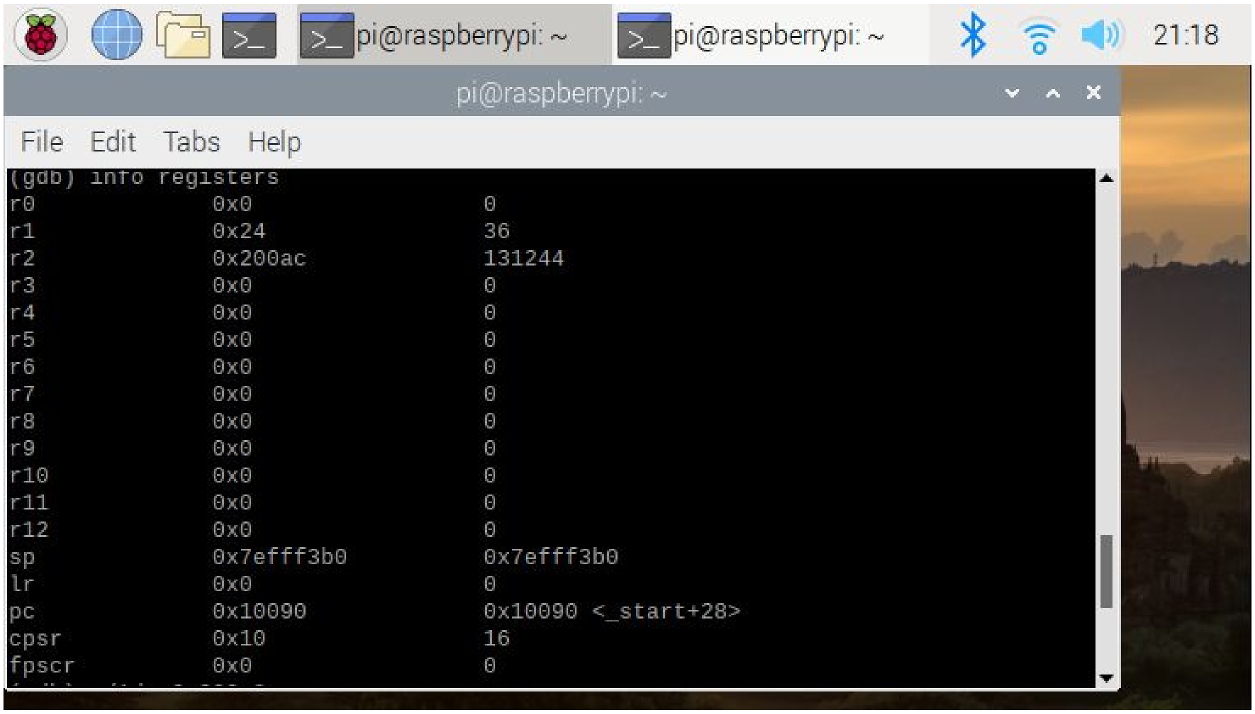
This is the source code for arithmetic2.s. In this code it can be seen that 11 is stored in val2, 16 in val3 and 6 in val1. The goal of this code is to print the result of 11+9 +16 - 6.



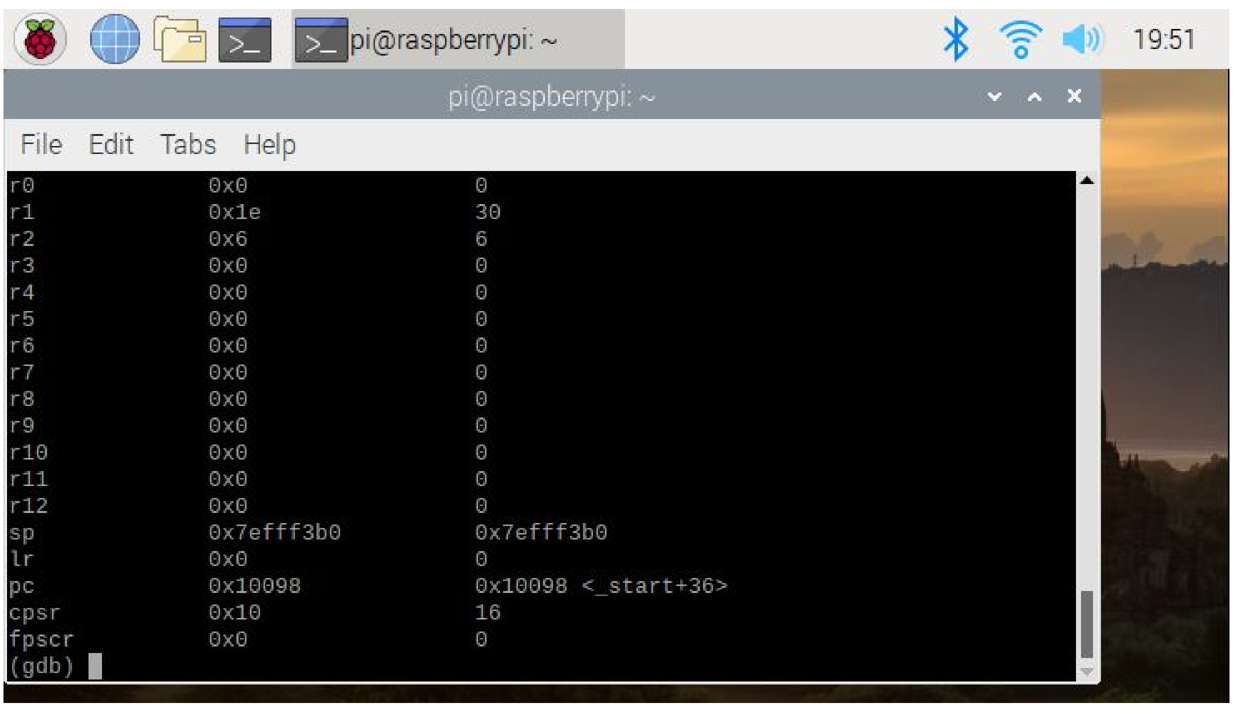
Command ls was used to check the current directories and to make sure the current file was saved in the places intended for it to be saved.



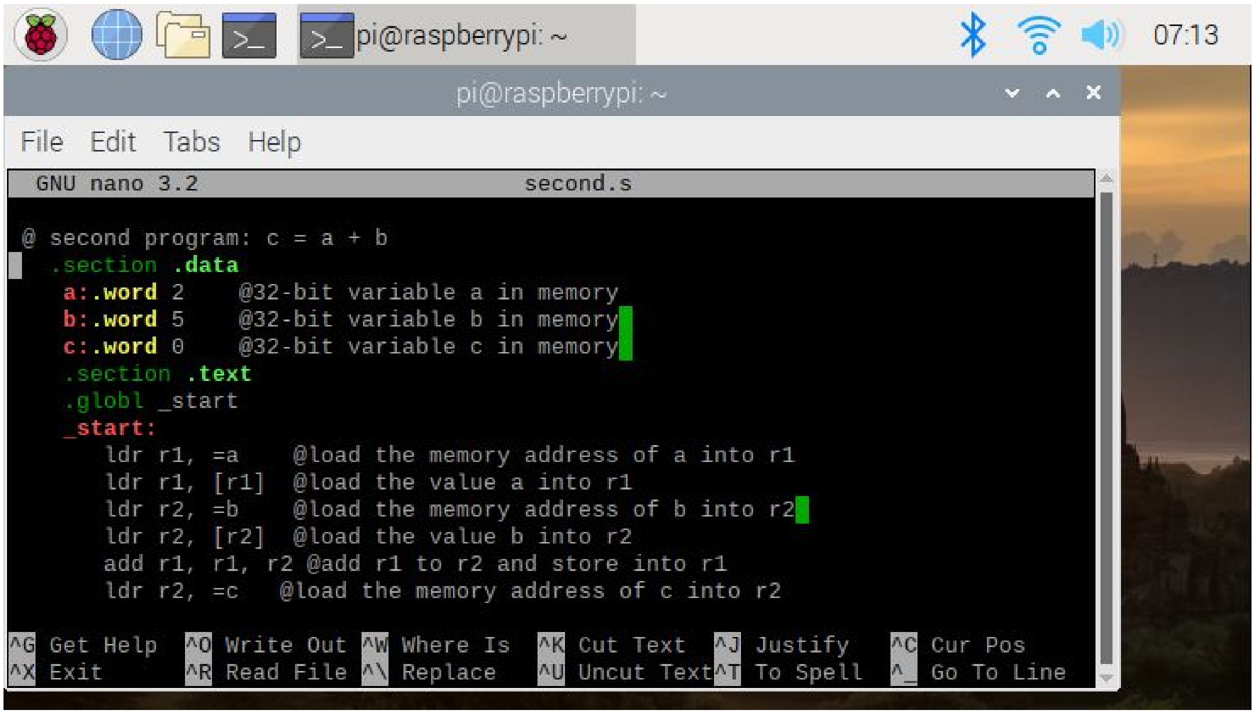
The breakpoint is set to line 15 and (gdb) info registers pulls up all the registers and the stored values in them. And based on this you can see that register r2 is the location where it is saved.



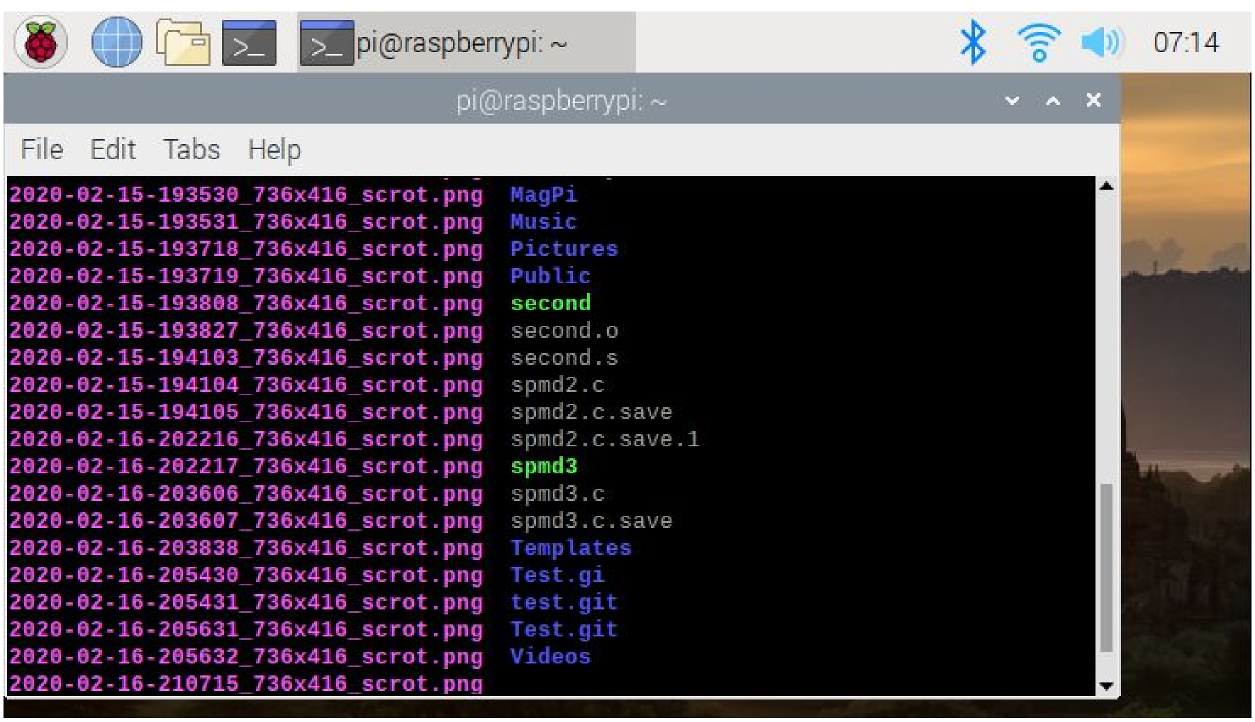
6 comes from val1 and adding 16 to 20 gives 36.



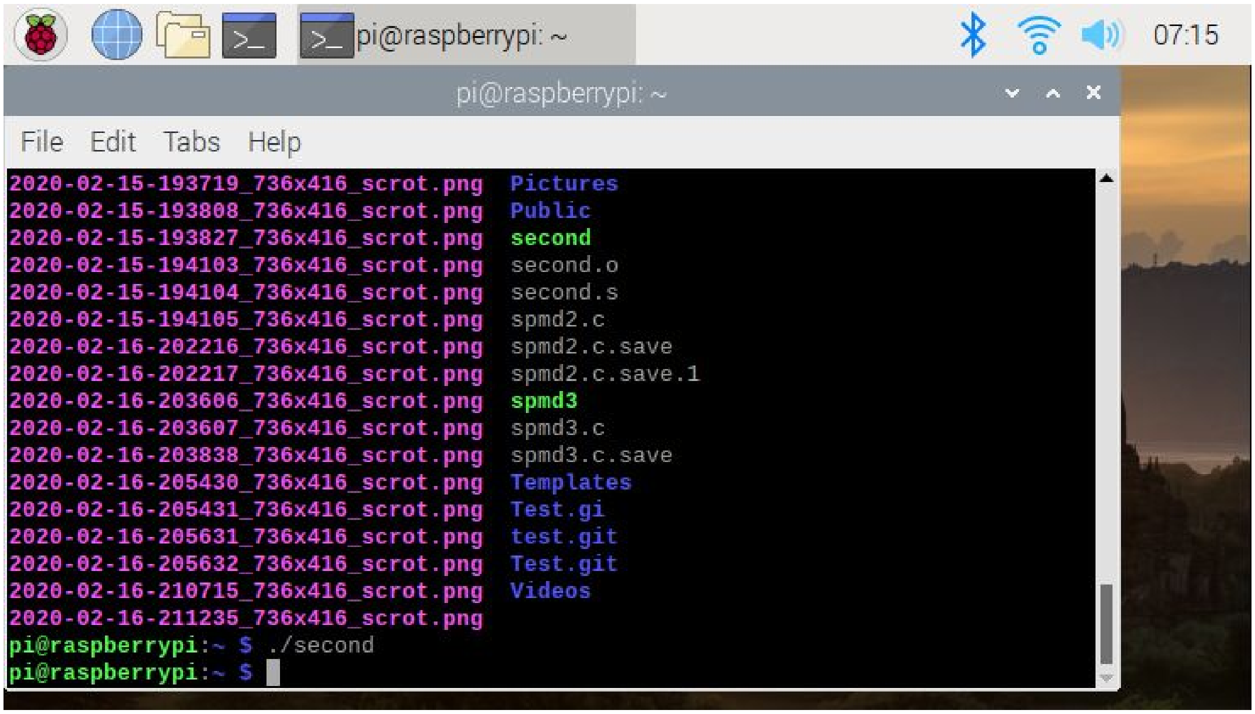
30 is the final result after subtracting 6 from 36 and it is stored in r1. Since the output shows what is being expected, it is a valid output. And 0x1e shows the hexadecimal format of 30.



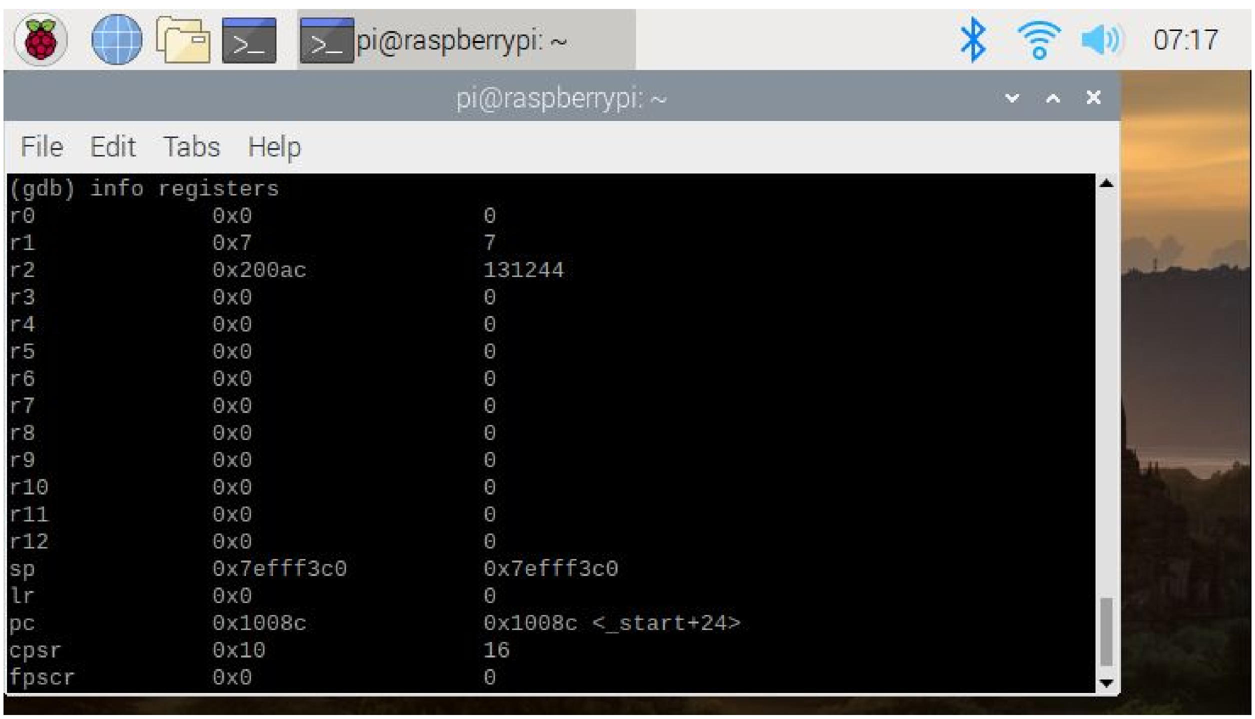
This is the source code for second.s. It traces the code that there is 2 word value in a memory address, 5 word value in the b memory address and 0 word value in the c memory address. After declaring this, the address of a will be loaded into register r1, therefore 2 will be stored in r1. The address of b will be loaded into register r2. Since there is an add function, 2 and 5 will be added and stored into r1. C is being loaded in r2 where 7 (2+5) is stored.



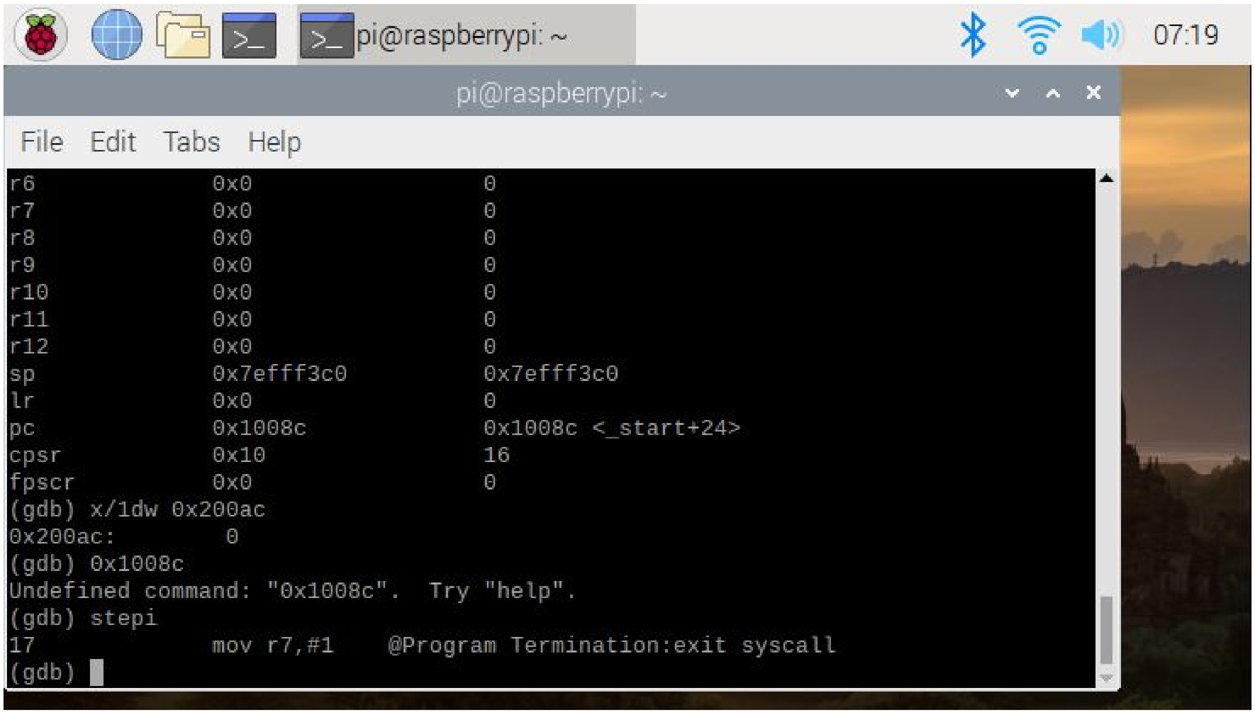
Command ls was used to check the current directories.



There was no output after entering ./second because the code in second.s only manipulates the numbers in the registers and does not print or return any values. Therefore, there is no output.



(gdb) info registers lets us know what is stored in each register.



x/ 1 was used to look at the memory in only one entry. dw indicated the decimal format in a word size entry. 0x200ac that is stored at r2 indicates that the memory location has been stored at this point and its 0. Stepi is used to run the last line of the code.

<u>Parallel Programming Questions</u>

1. **Identifying the components on the raspberry PI B+.**
   The components include board computer, 1 gigabyte RAM and multi-core CPI
2. **How many cores does the Raspberry Pi's B+ CPU have?**
   The Raspberry Pi's B+ CPU has four cores.
3. **List three main differences between X86 (CISC) and ARM Raspberry PI (RISC). Justify your answer and use your own words (do not copy and paste).**
   ARM are RISC processors where the instructions are only in the form of registers unlike CISC, RISC has a more generalized form of registers. CISC has more instructions to accomplish a task than RISC, therefore the code is more complex in CISC systems. Latest ARM processors are using bi-endian instead of little endian and be eligible for changing endianness according to the code.
4. **What is the difference between sequential and parallel computation and identify the practical significance of each?**
   Parallel programming takes a problem and breaks it more and more so that it can be executed all at once by multiple processors. Sequential programming breaks a problem step by step and executes on one processor. The significance of this programming is that

when there is a problem that cannot be broken down into smaller independent instructions. The significance of parallel programming is that it uses the resources efficiently therefore it saves time.

5. **Identify the basic form of data and task parallelism in computational problems.**
   Data parallelism is when the same operation is conducted on all the data presented and this can only be done when the instructions are independent. When there is more data to be covered, data parallelism is the best way because everything is done parallely so it consumes less time. Task parallelism is what the computer performs on the data and not What the output of the data is. This is used only when the tasks are independent from each other.
6. **Explain the differences between processes and threads.**
   The Process is when a program is being executed. Processes don't share memory spaces, instead they run separately in the memory spaces. Whereas threads share memory spaces. Thread is within the process that can be called for execution. Processes provide resources for the code to be executed. Processes deals only with the environment the code is written but a thread interacts with the hardware.
7. **What is OpenMP and what is OpenMP pragmas?**
   OpenMP is an API that helps shared memory in multi platform programming on different operating systems. OpenMP pragmas is when a set of compilers controls how the program is executed. Pragmas are created in a way that if the compiler does not support them then the program will still execute a valid output without parallelism involved.
8. **What applications benefit from multi-core (list four)?**
   The top applications that benefit from multi-core are Web servers, compilers, database servers and Multimedia applications.

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

- Since computer architecture follows the concept of parallel programming, it needs many threads instead of one.
- A single core requires expensive materials to be used and it needs high maintenance since they have intense pipelined circuits.
- The single threading has a limit to how fast it can be done. So anything that exceeds the point should use multiple threads for execution.
- Most upcoming applications are in the use of multithreading.