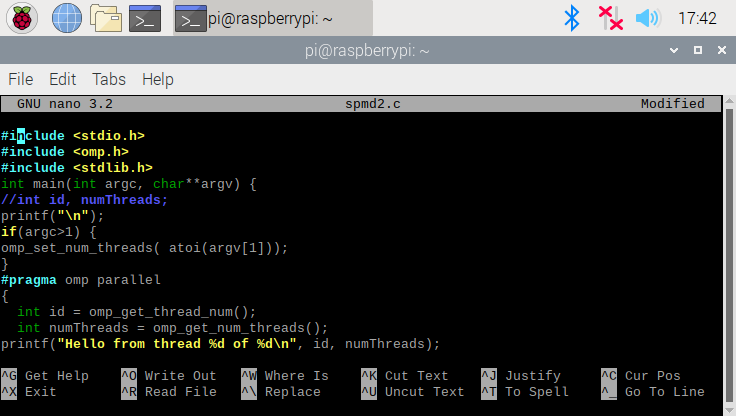
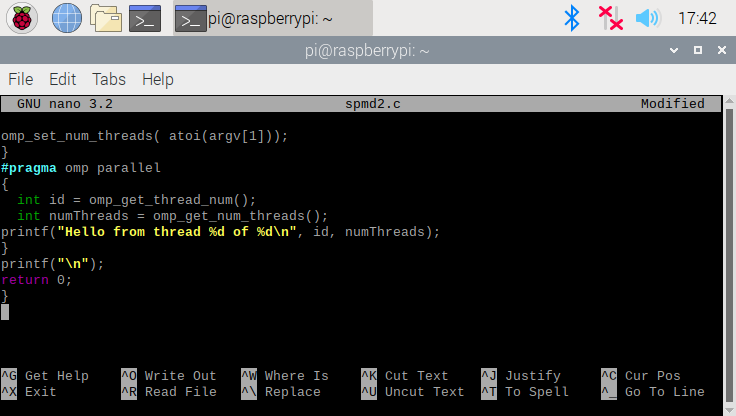
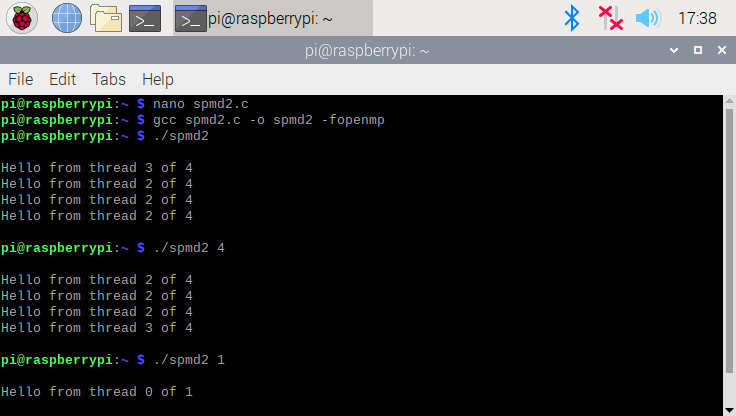
ROOHI Beginning of Parallel Programming:

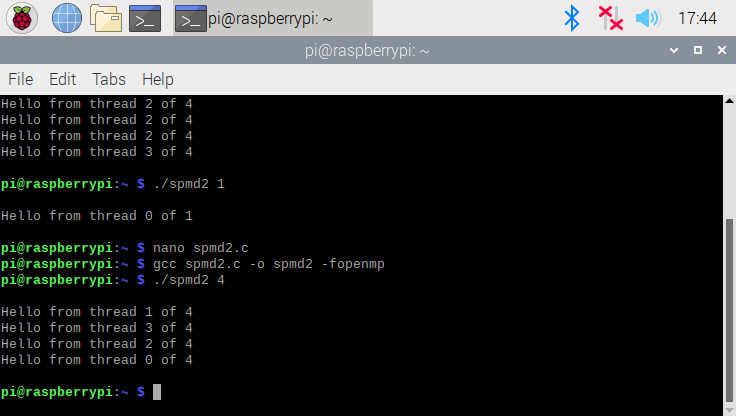




The above 2 screenshots show the source code for spmd2.c. The later changes made to fix the code are also reflected above, where each code now has its private copy of id and numThreads variables.



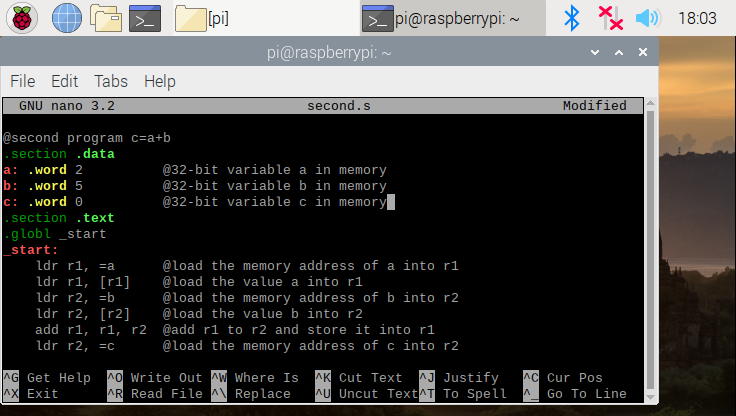
This is the output observed with the original source code, where one thread is repeated multiple times because the cores are sharing one memory bank. In addition, the variables declared outside of the block will be run parallel on separate threads, and all threads are sharing the variable’s memory location.

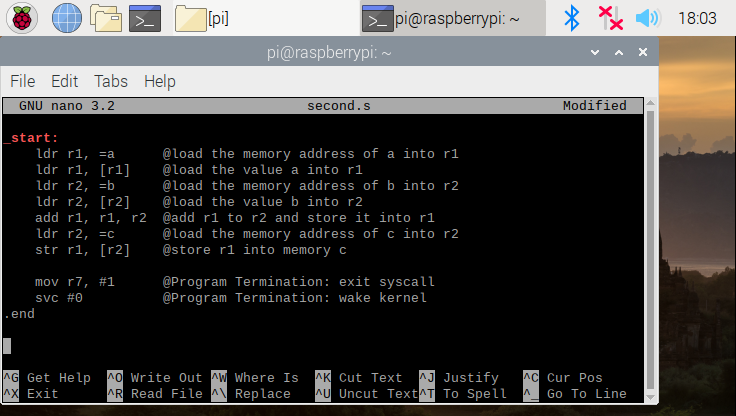


This output reflects the fixed code. Now that each thread has its own copy of the variable’s memory location and value, so they are all aligned.

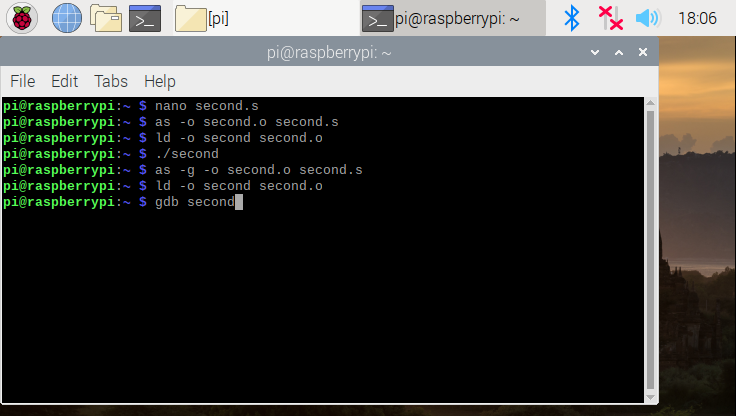
ROOHI END OF PARALLEL PROGRAMMING

BEGINNING OF ARM PROGRAMMING ROOHI:

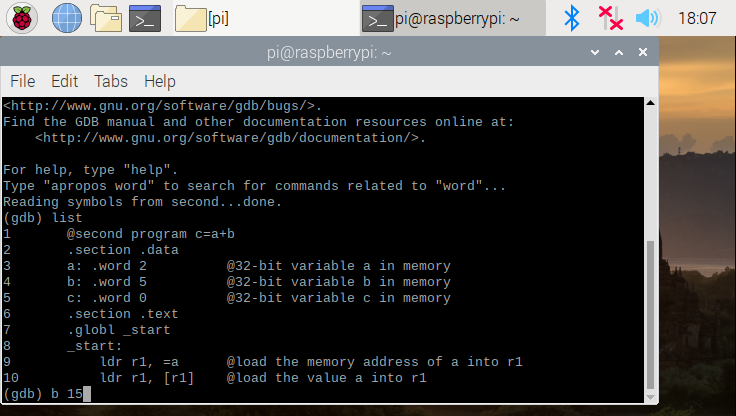
Second.s  




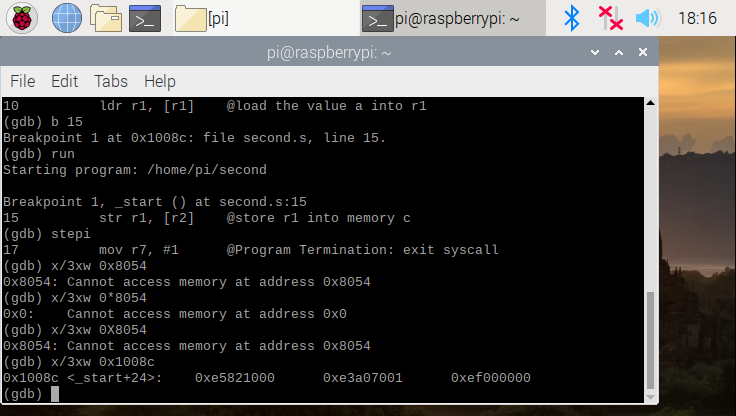
The above 2 screenshots are the source code for the second.s program. Under the .data section, the memory addresses a, b, and c have the size of a word (which is 32-bit) which have the values 2, 5, and 0 stored in them respectively. Under the .text section, we are trying to add the values a and b and store it in c by indirectly storing the values in registers using the memory addresses.



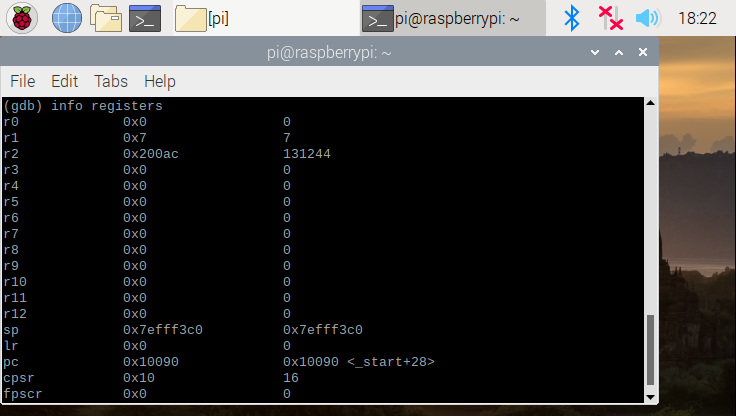
The commands above are basically assembling and linking the second.s file into an executable called second. There is no output while running the program because manipulations between the CPU and the registers are not shown since there is not I/O devices between them.



We can take a closer look at the registers and how they behave using the gdb command and setting a breakpoint, so that the program stops prematurely and does not exit while trying to look at the registers.

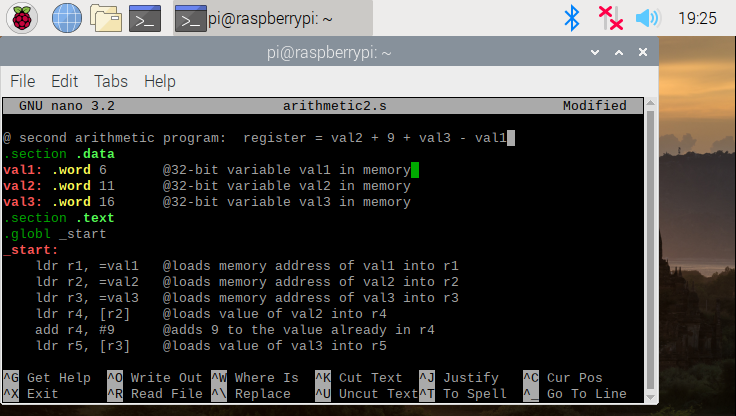


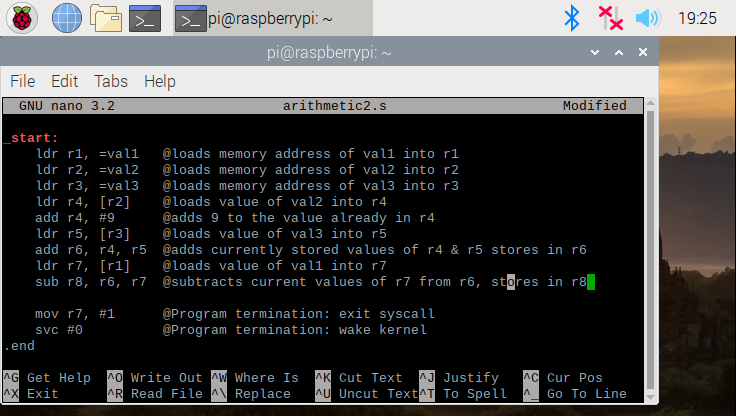
The stepi command in gdb lets us step over one instruction at a time, and the x/3xw 0x1008c command displays three words in hexadecimal starting at the location/address 1008c.



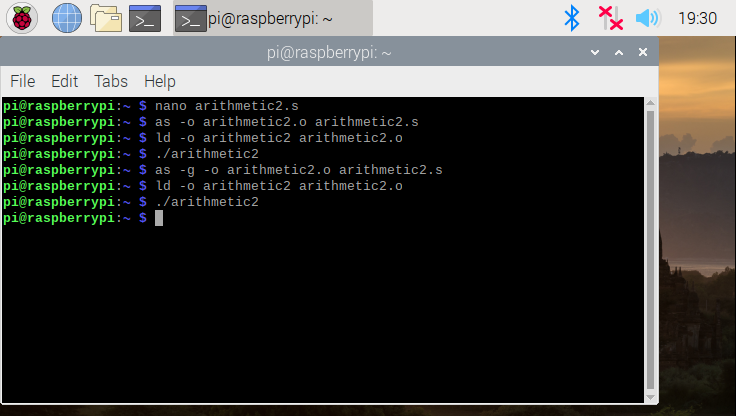
The registers show the results of the operation that was done in the program.

Arithmetic2

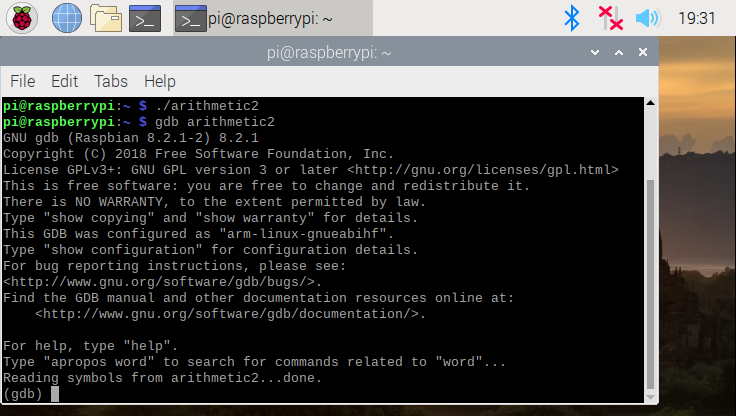


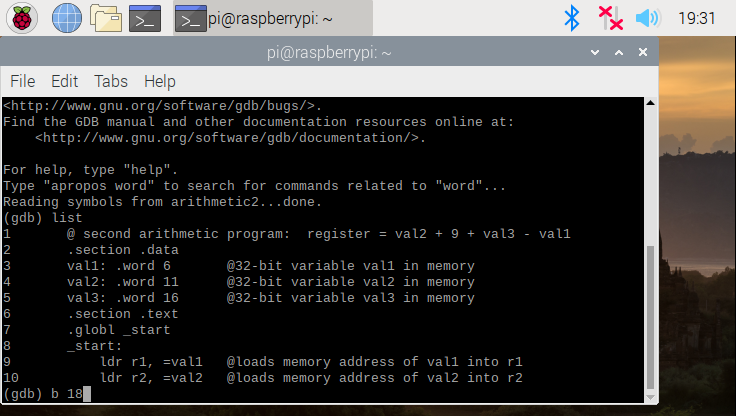


The above 2 screenshots are the source code for the arithmetic2 program. Under the .data section, we are declaring memory size of 32 bits (word) to store the values 6, 11, and 16 in val1, val2, and val3 respectively. The .text section is trying to perform the operation: 11+9+16-6.

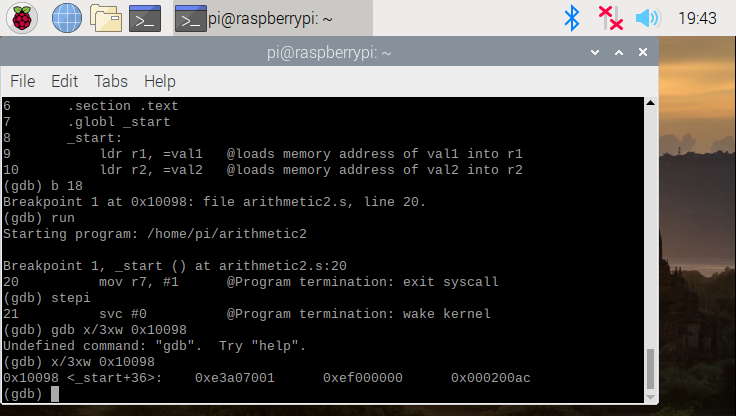


The arithmetic2 program is being assembled and linked to produce an executable, and does not produce an ouput when trying to execute because manipulations between the CPU and the registers are not shown since there is not I/O devices between them.

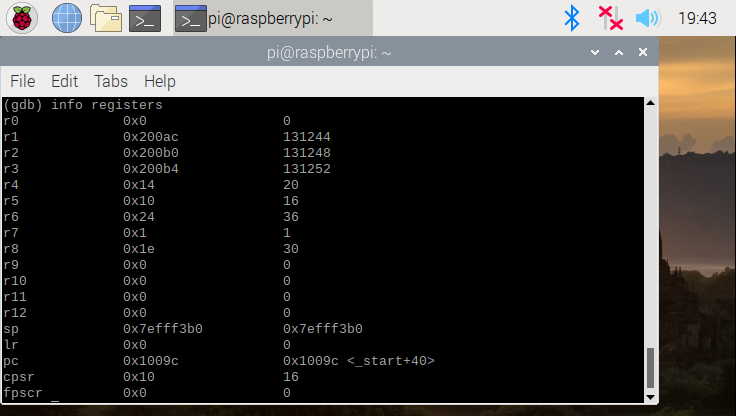




Since we cannot see the ouput directly, we need to use the gdb command to take a look at what is stored in the registers.



The stepi command in gdb lets us step over one instruction at a time, and the x/3xw 0x10098 command displays three words in hexadecimal starting at the location/address 10098.



The registers reflect the results of the operation/arithmetic being done in the arithmetic2 program. The final answer for the arithmetic is 30, which can be seen in r8.

END OF ROOHI ARM PROGRAMMING