Name: Minh Binh Nguyen
PantherID: 002-46-4288

Project 2 – Task 4

ARM Assembly Programming

Part 1

I started with getting my Raspberry connected to my computer and redirected to “Project” folder where I store all my Project files for this class (Figure 1).

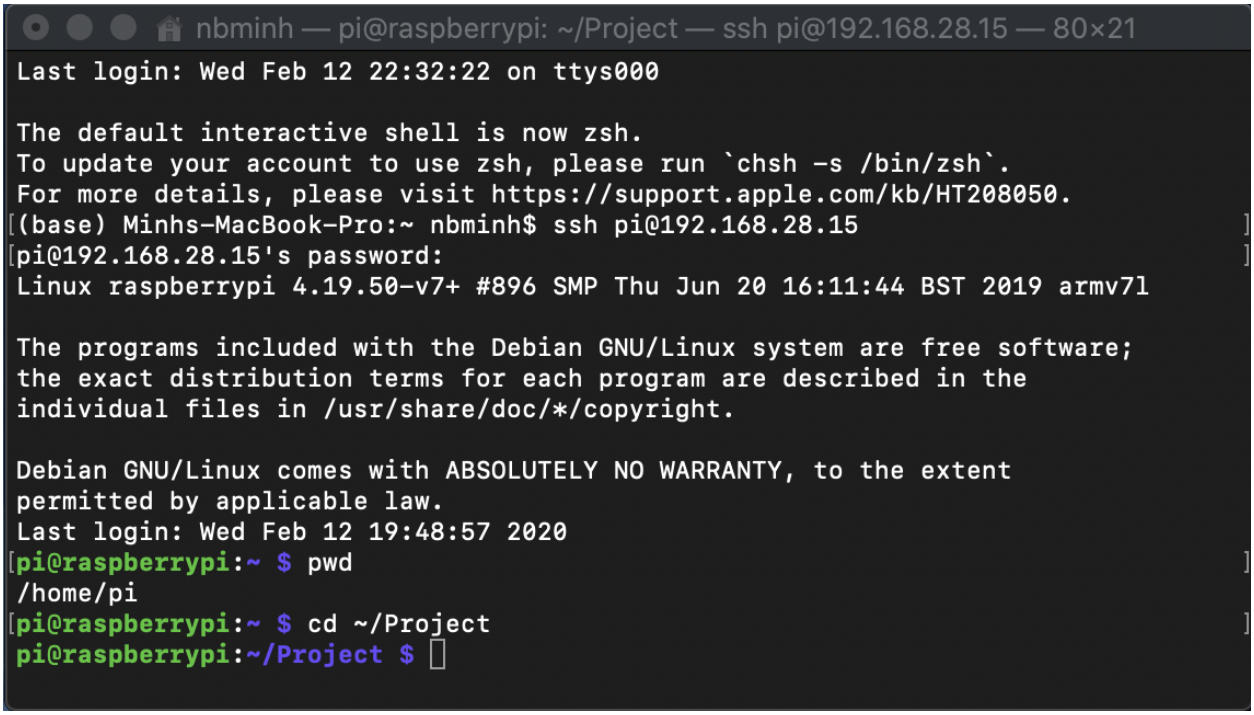

Figure 1

Then I created a file called second.s and typed in the code (Figure 2).

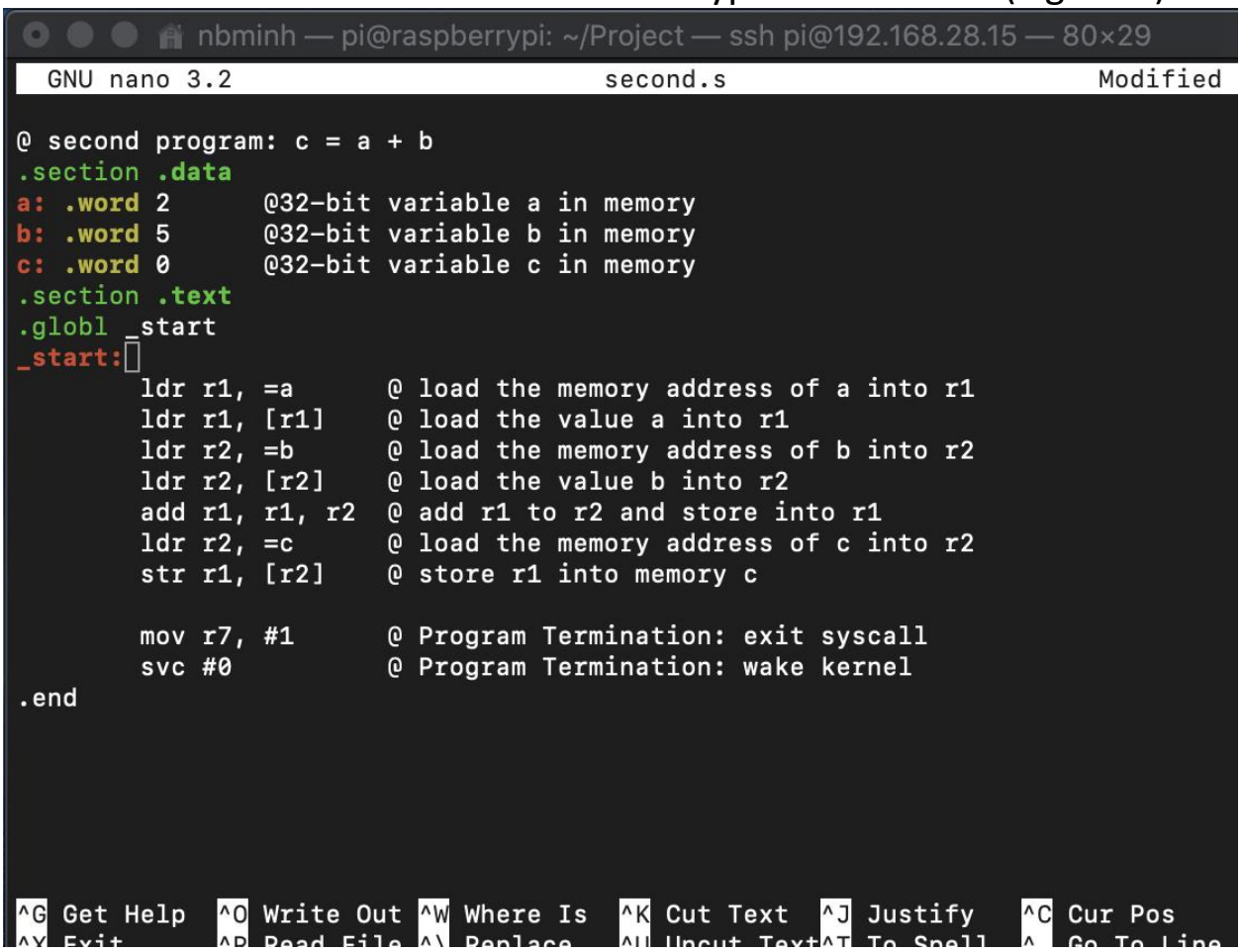

Figure 2

Then I assembled and linked the file to get an executable file (without using the flag “-g”). When running it, I didn’t see any output. This is to be expected because the code doesn’t have any instructions to produce any outputs (Figure 3).

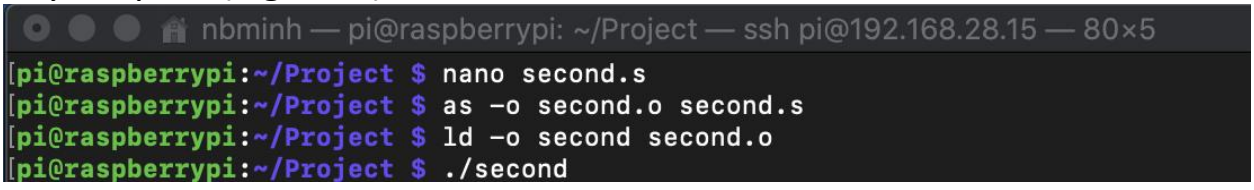

Figure 3

Then I re-assembled and re-linked the file again (this time adding the "-g" flag for debug purpose), then use the GDB to debug the program (Figure 4).

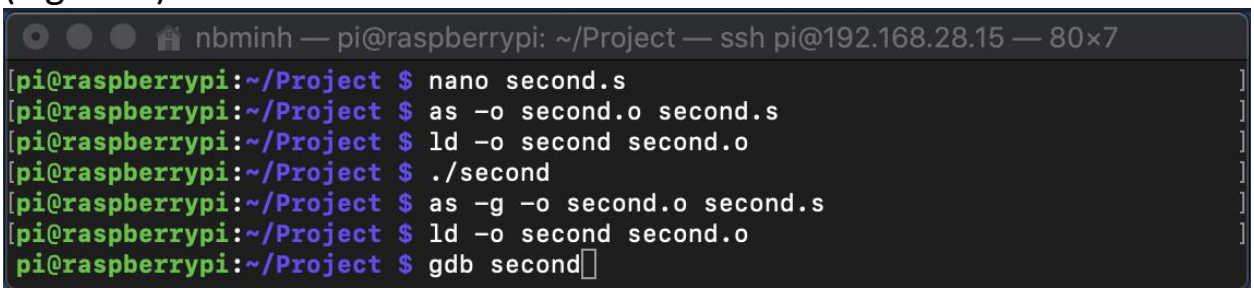

Figure 4

Here is the code being listed by the gdb. I set a breakpoint at line 9 and started debugging it line by line using stepi to understand what is going on with the registers and the memory (Figure 5).

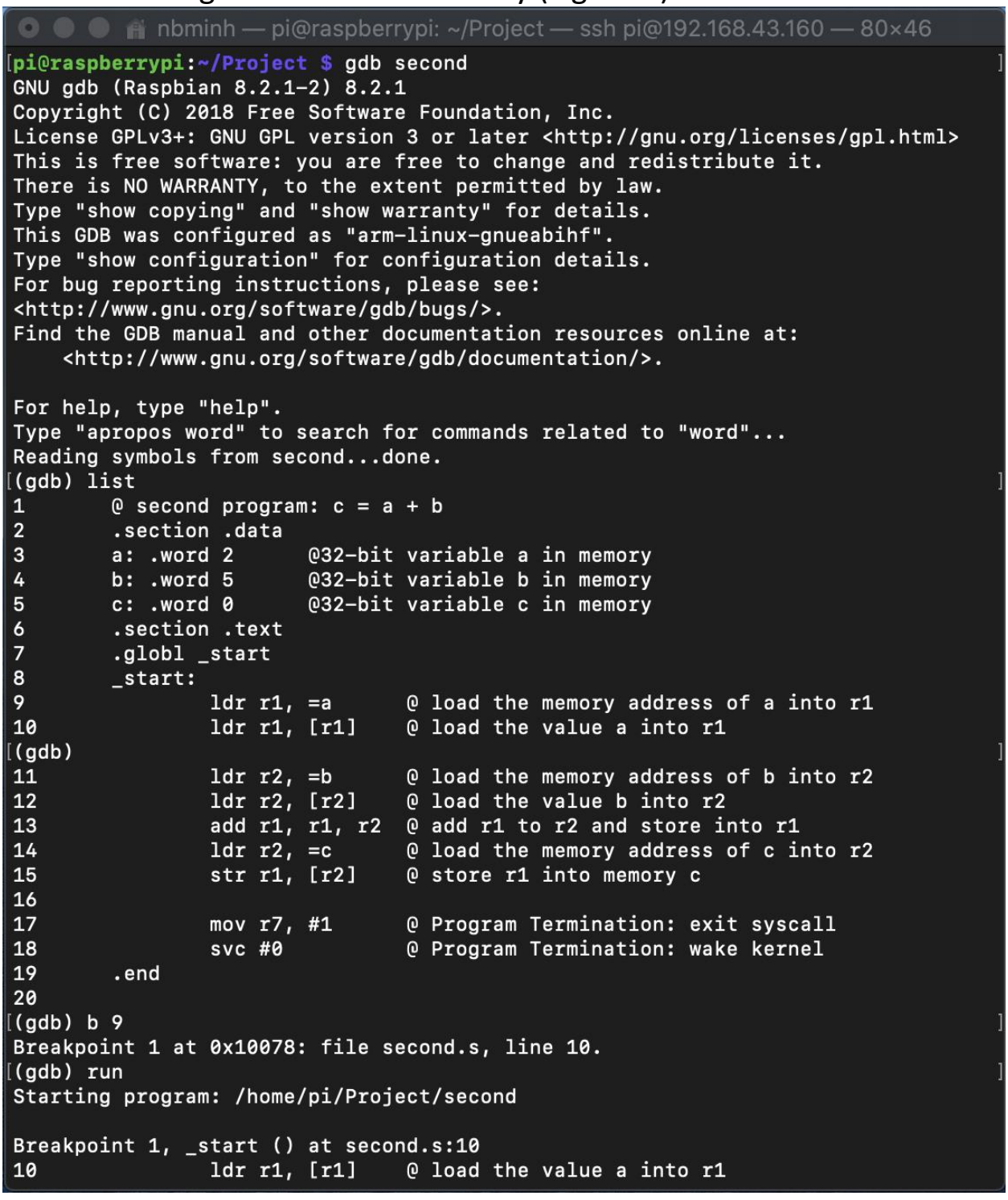

Figure 5

After loading the address of a into r1, I can see the actual address being stored in r1 using the “info registers” command. It’s 0x200a4. By using the examining command “x/1xw 0x200a4”, I can see 1 item word in hex at that address. And it holds the value 2h which is 2d as expected (Figure 6).

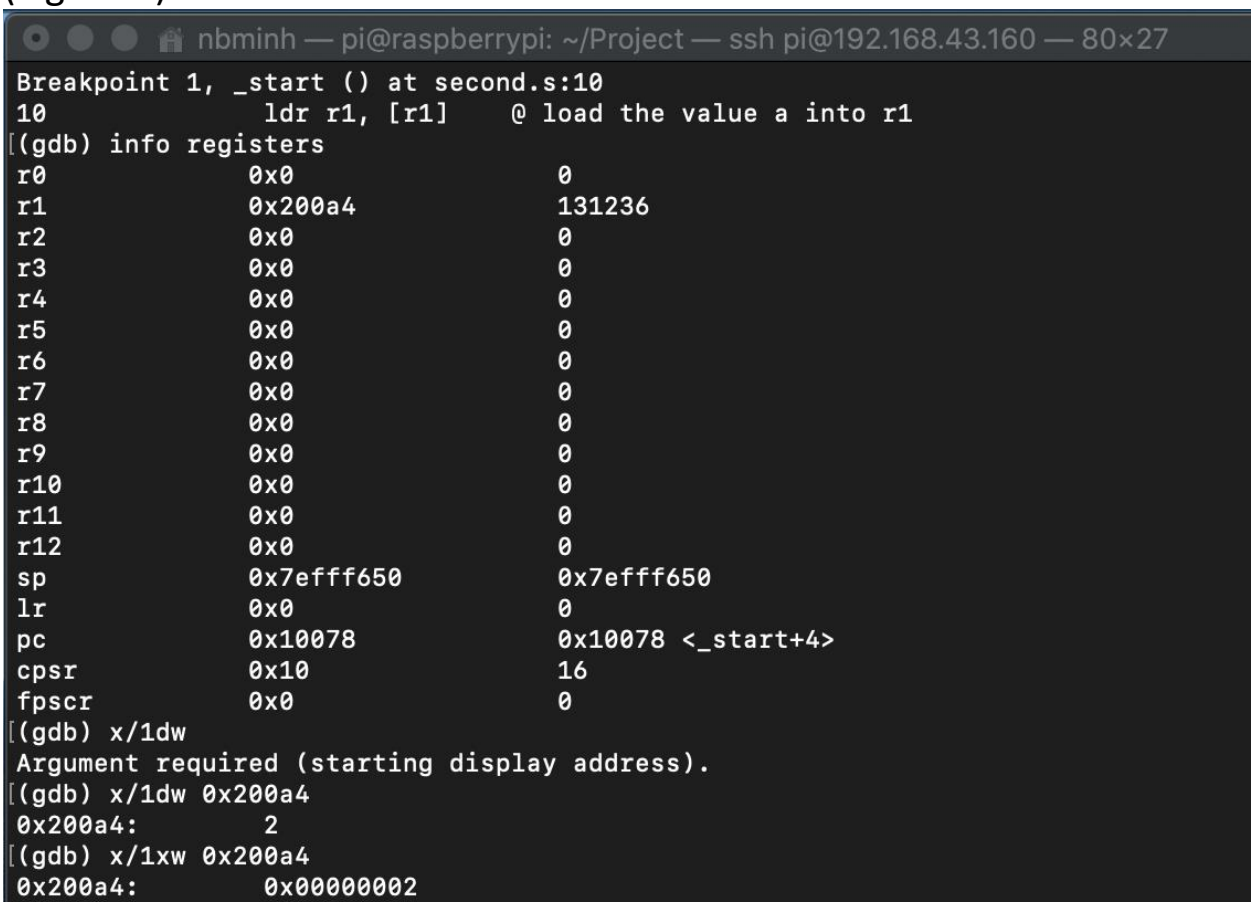

Figure 6

After executing line 10, the value of a has been stored into r1 register (Figure 7)

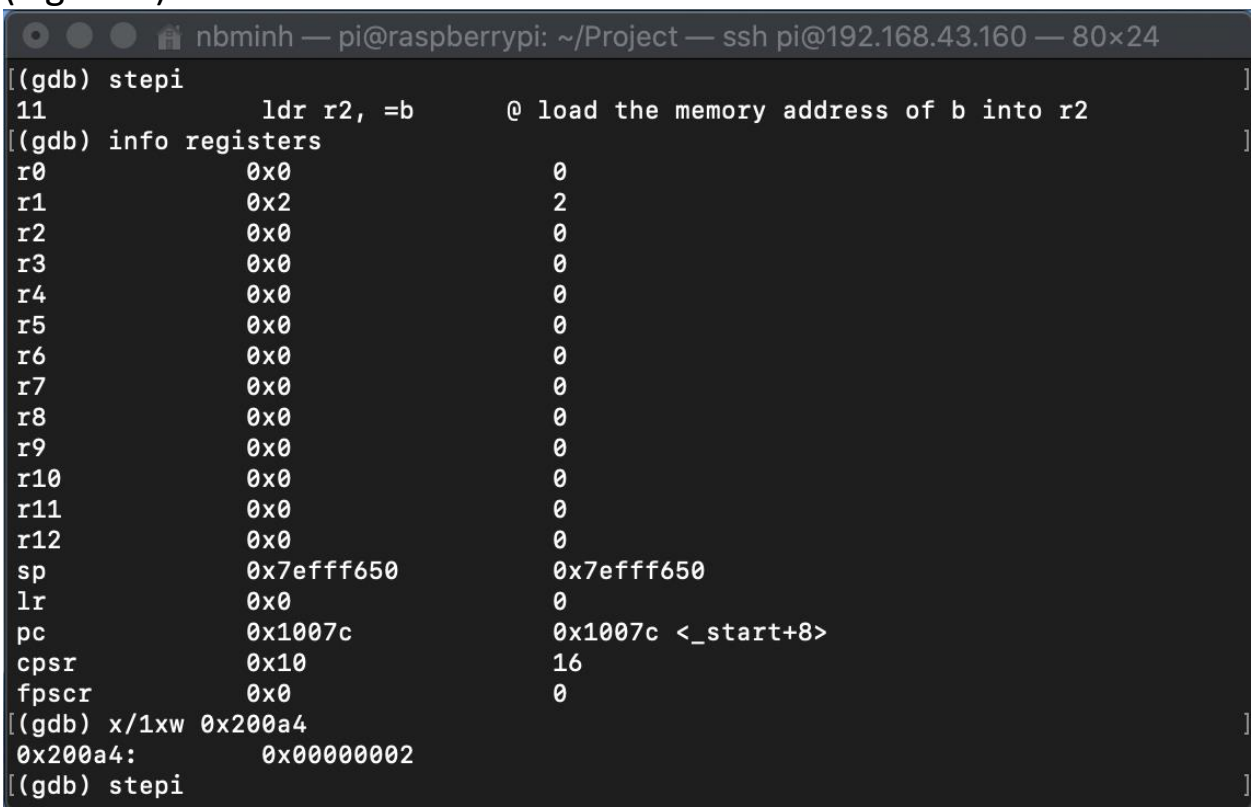

Figure 7

Executing line 11 gives me the address of b in r2 register. The address is 0x200a8 which is a 4-byte step from address of a. This is because a is a 4-byte word. By examining this address, I got the value of b in the memory (which is 5h, also 5d) as expected (Figure 8).

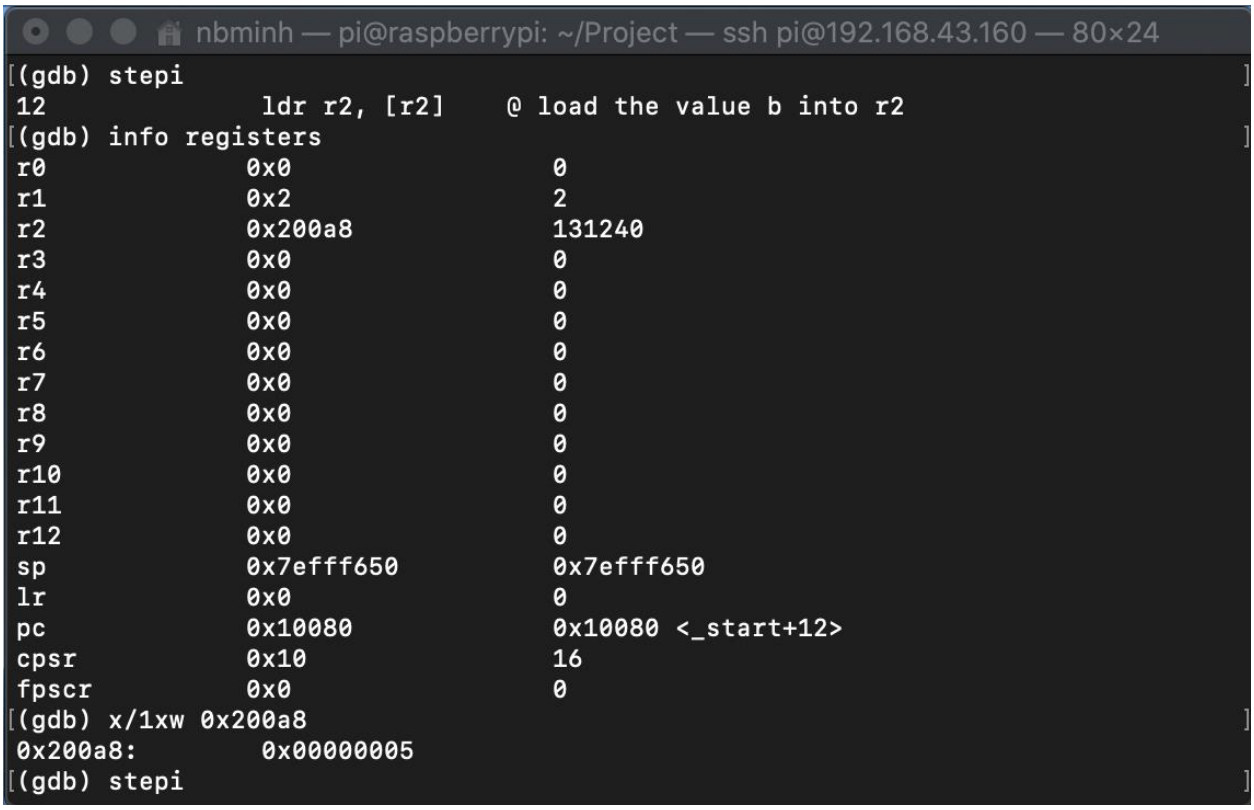

Figure 8

I keep stepping line by line and getting the results as expected. Notice that the address of c is 0x200ac, which is also 4-byte different with the address of b. This is also because b is a 4-byte word (Figure 9).

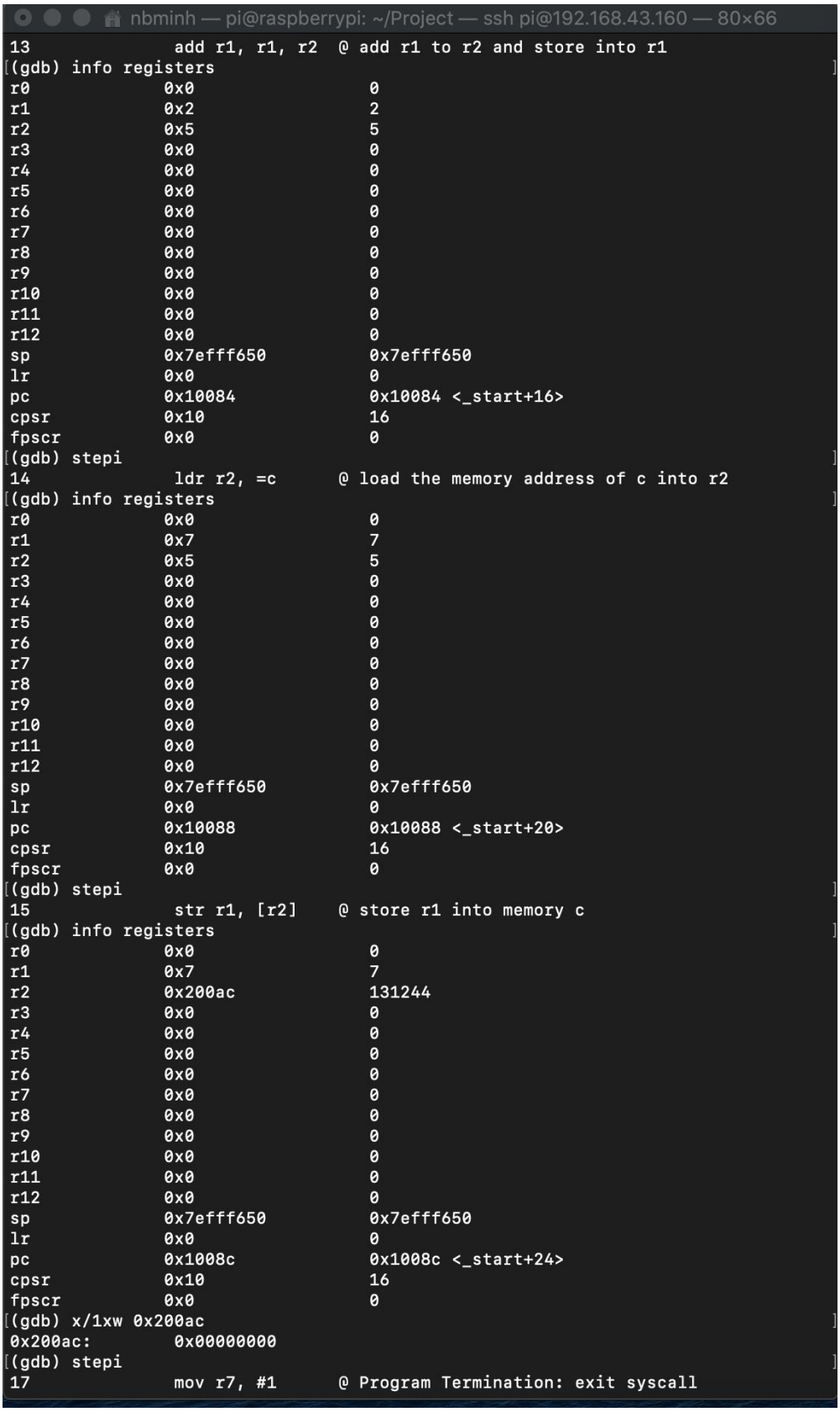
nbminh — pi@raspberrypi: ~/Project — ssh pi@192.168.43.160 — 80×66
13 add r1, r1, r2 @ add r1 to r2 and store into r1
(gdb) info registers
r0 0x0 0
r1 0x2 2
r2 0x5 5
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x0 0
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0
sp 0x7efff650 0x7efff650
lr 0x0 0
pc 0x10084 0x10084 <_start+16>
cpsr 0x10 16
fpscr 0x0 0
(gdb) stepi
14 ldr r2, =c @ load the memory address of c into r2
(gdb) info registers
r0 0x0 0
r1 0x7 7
r2 0x5 5
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x0 0
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0
sp 0x7efff650 0x7efff650
lr 0x0 0
pc 0x10088 0x10088 <_start+20>
cpsr 0x10 16
fpscr 0x0 0
(gdb) stepi
15 str r1, [r2] @ store r1 into memory c
(gdb) info registers
r0 0x0 0
r1 0x7 7
r2 0x200ac 131244
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x0 0
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0
sp 0x7efff650 0x7efff650
lr 0x0 0
pc 0x1008c 0x1008c <_start+24>
cpsr 0x10 16
fpscr 0x0 0
(gdb) x/1xw 0x200ac
0x200ac: 0x00000000
(gdb) stepi
17 mov r7, #1 @ Program Termination: exit syscall

Figure 9

Before terminating the program, I went ahead and checked the memory at c's address. It stored the value 7, which is the correct result of the addition (Figure 10).

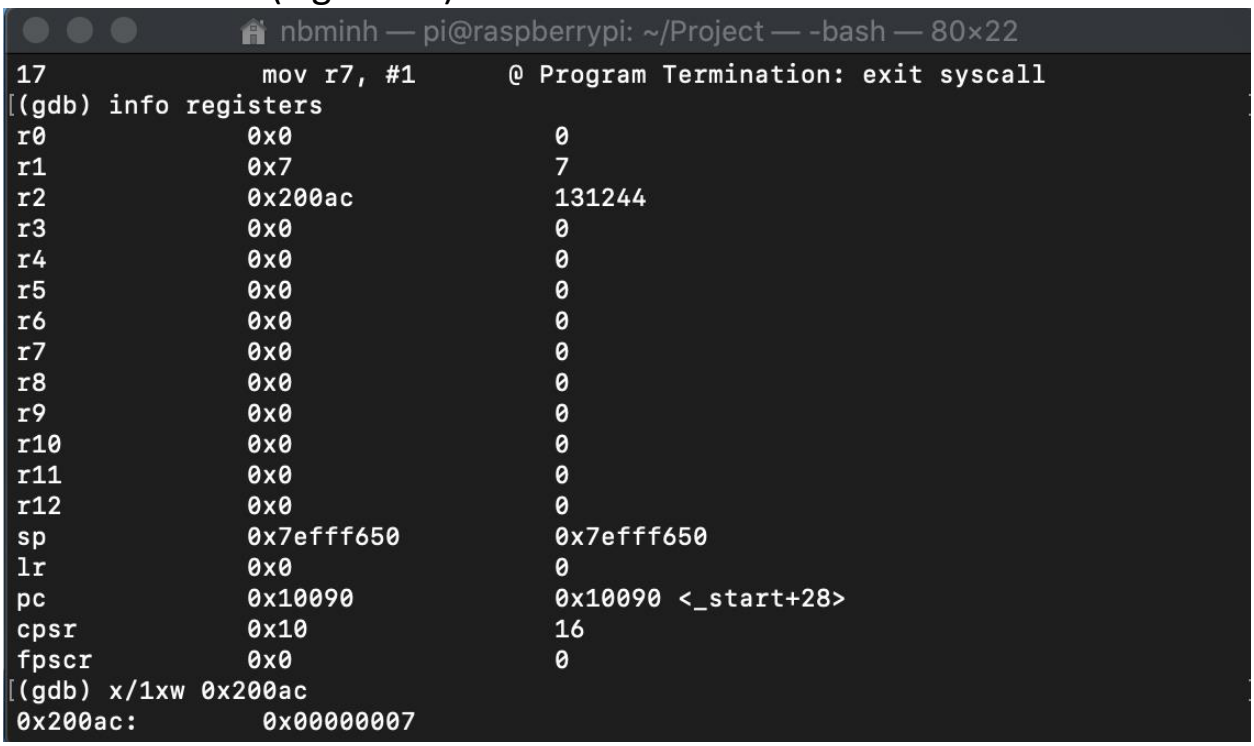

Figure 10

Part 2

I created a arithmetic2.s file and type in the code for this program using the editor (Figure 11).

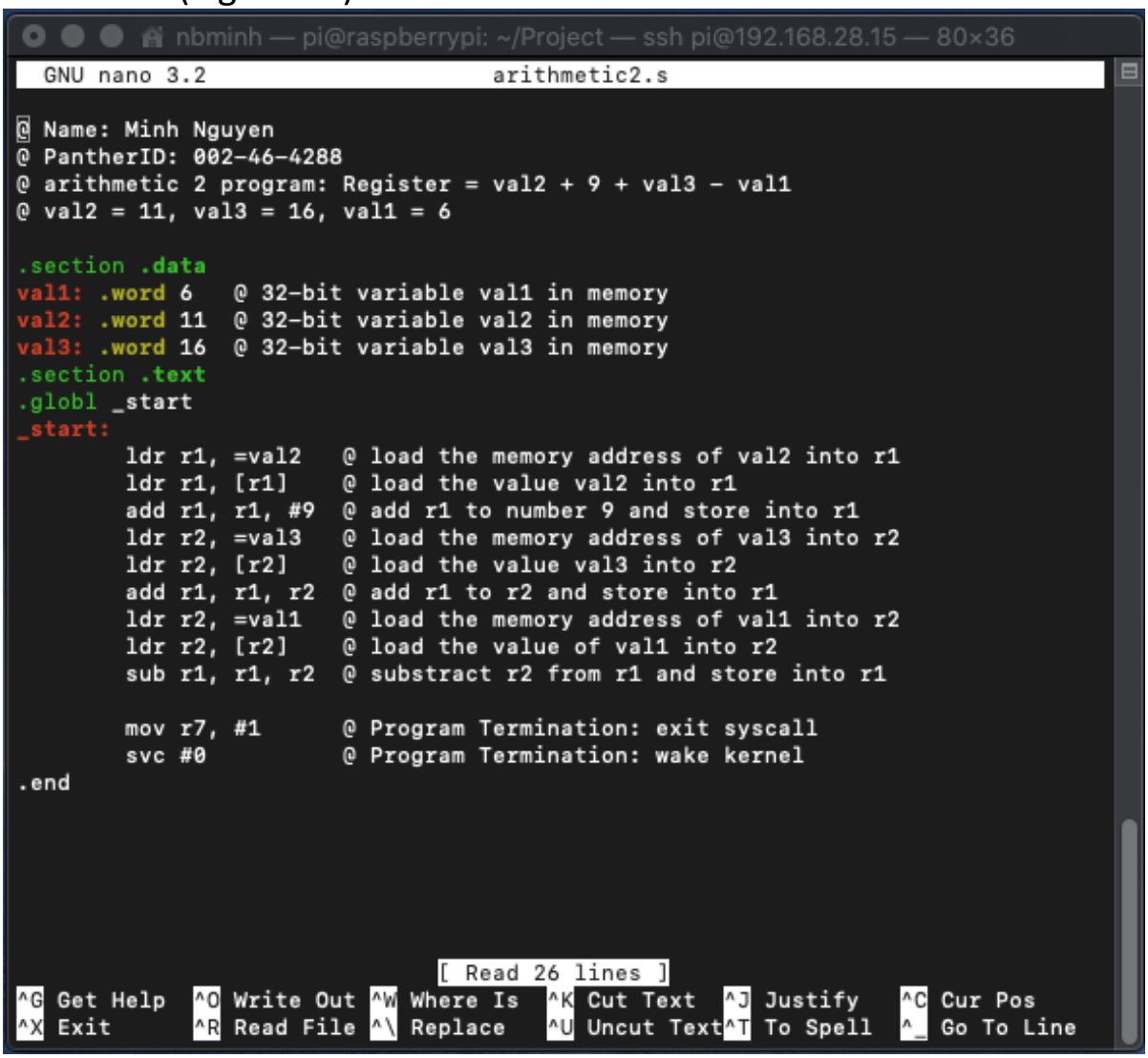

Figure 11

Running it gives me no output just like part 1. Because there is no instruction to produce the output for this program (Figure 12).

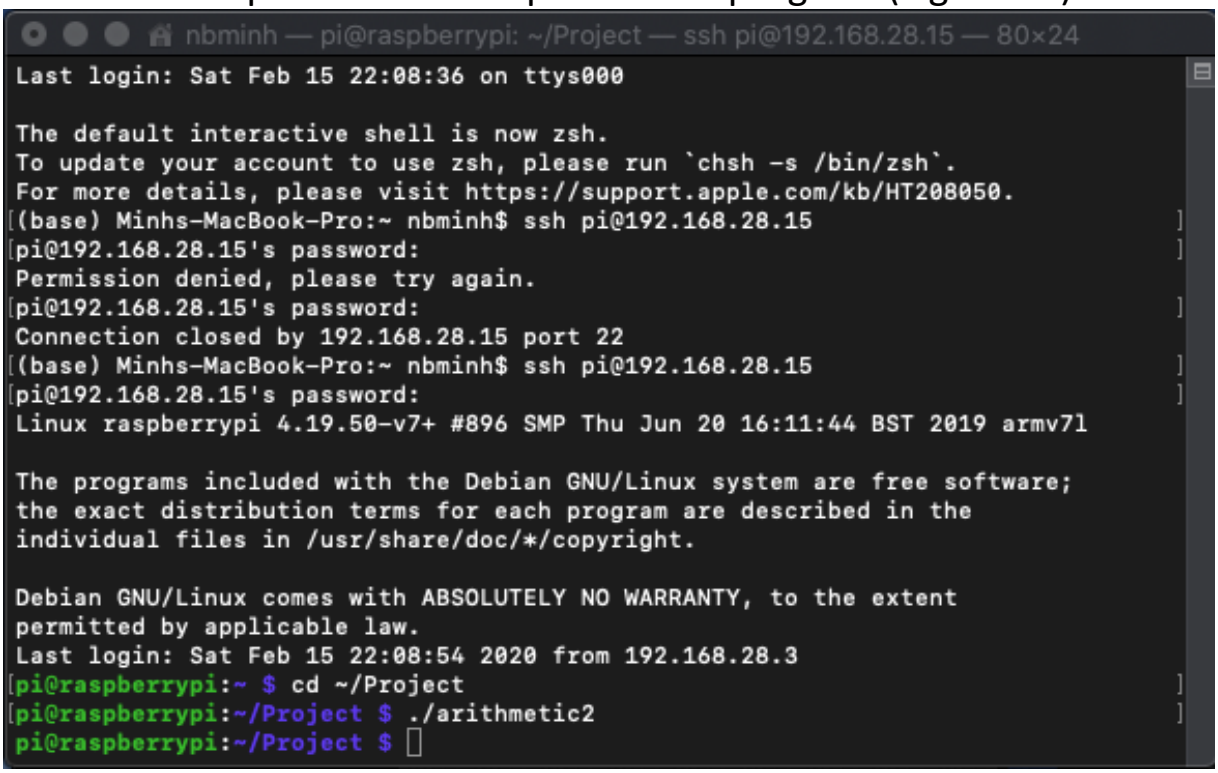

Figure 12

I started debugging it using GDB and set the breakpoint at line 14 (Figure 13).

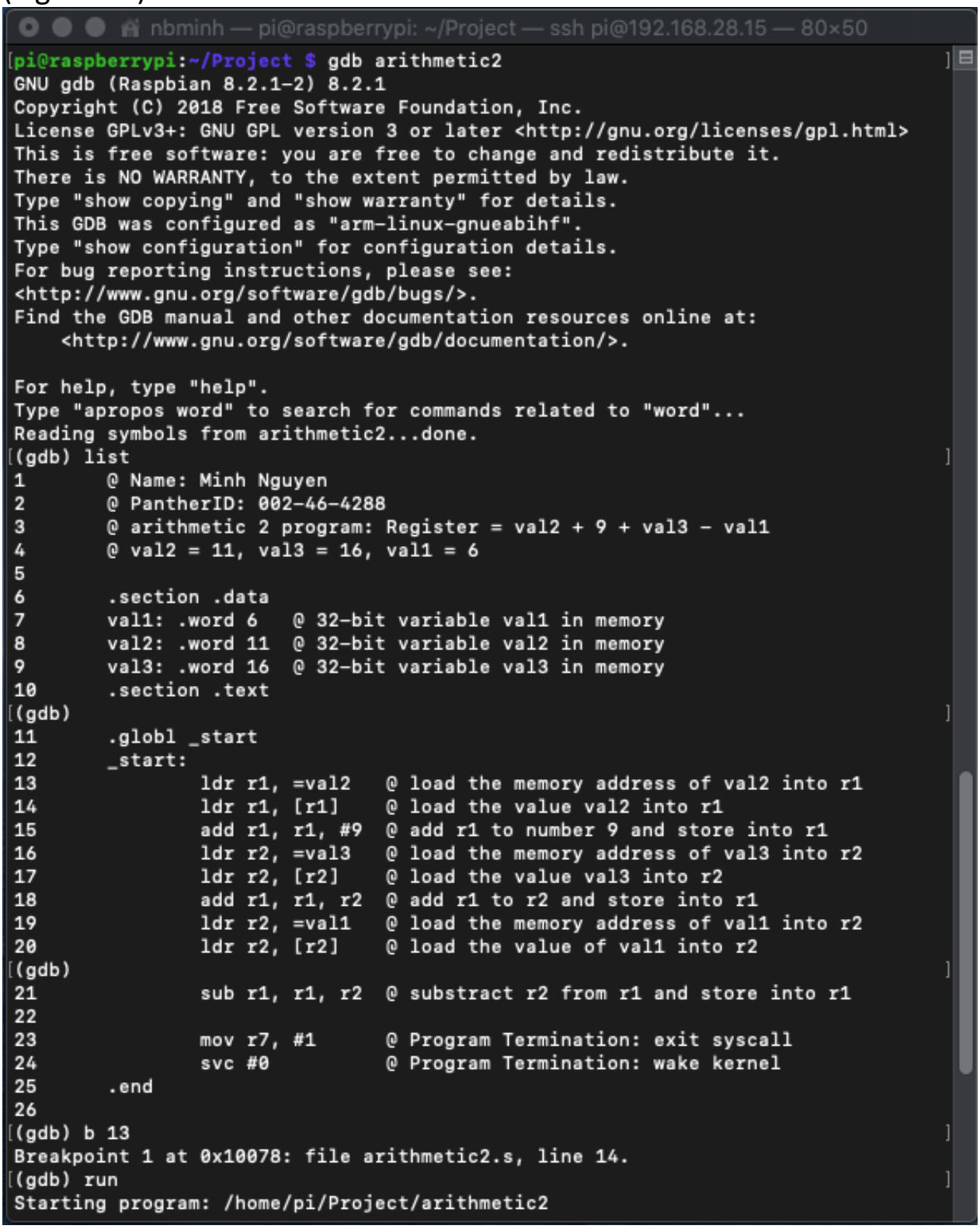

Figure 13

By using the same commands as part 1, I can step through the program line by line to examine the registers and memory data. After executing line 13, I got the address of val2 which is 0x200b0. Examining this address gave me the value b in hex (which is 11 in decimal) as expected (Figure 14).

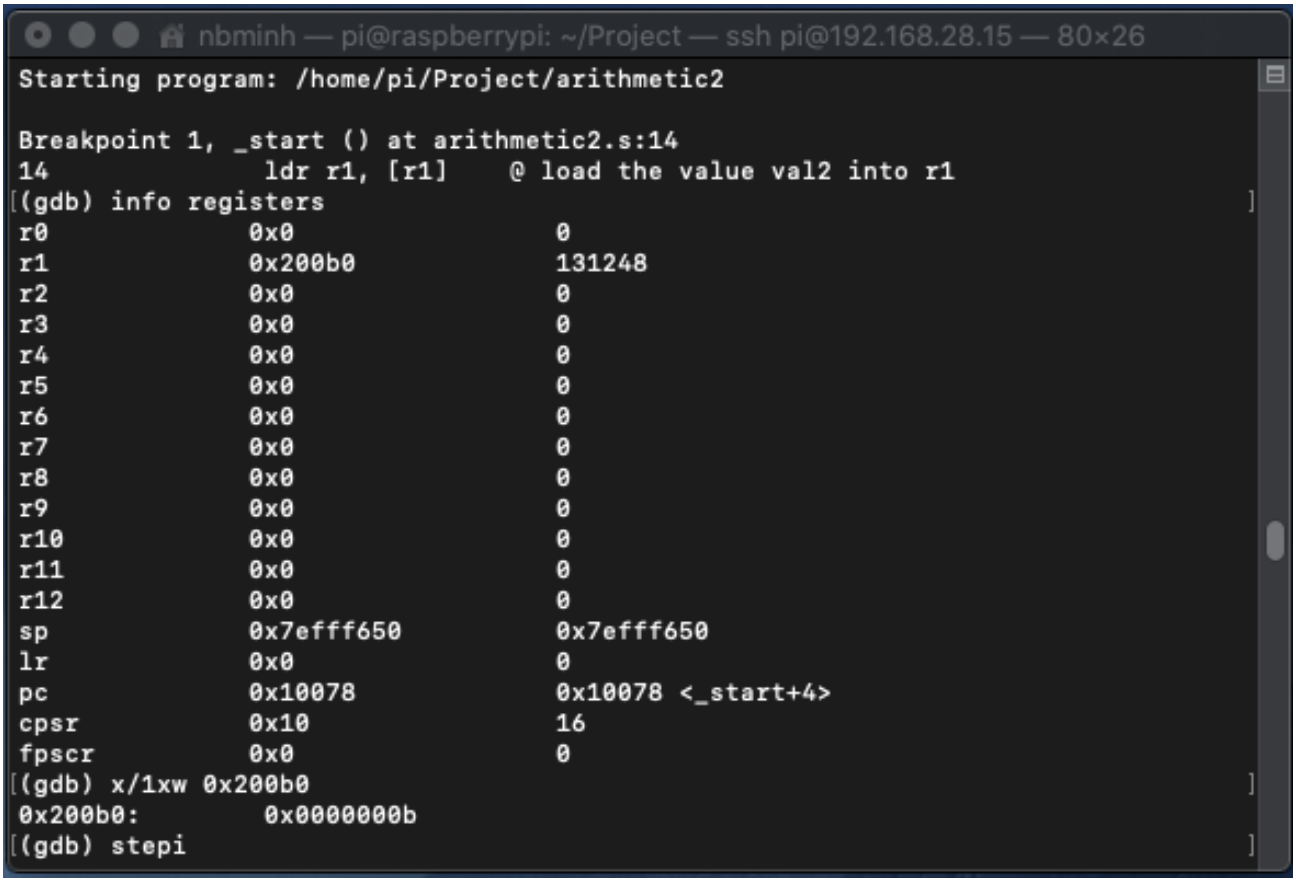

Figure 14

After executing line 15, I got the value 20 stored in r1 register as expected (adding val2 to number 9) (Figure 15).

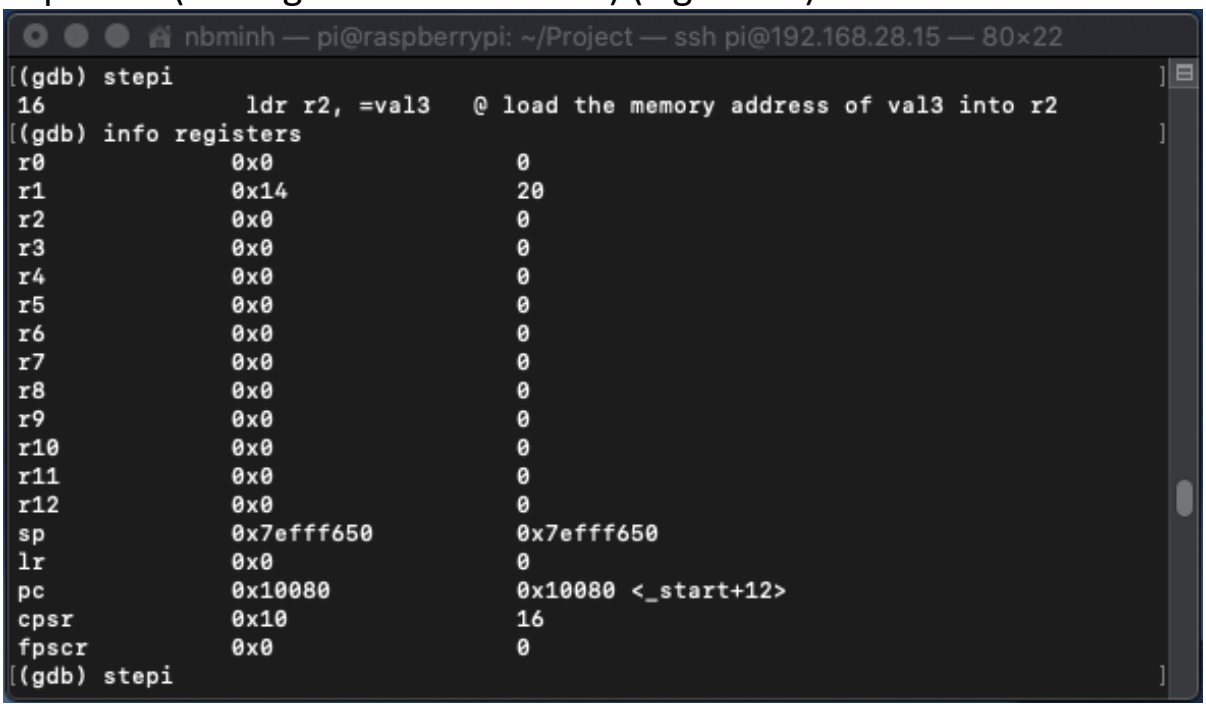

Figure 15

Executing line 16 gave me the address of val3, which is 0x200b4. The value this address holds is 10 in hex (16 in decimal) (Figure 16).

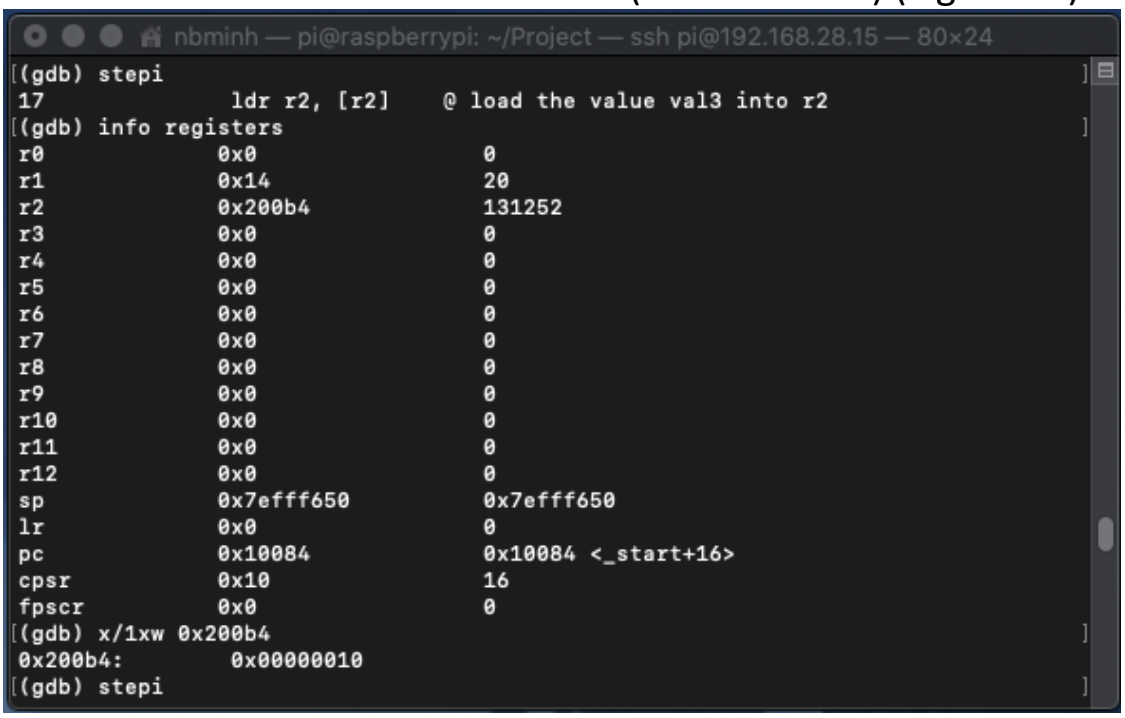

Figure 16

After the addition in line 18, r1 register holds the value 36 (val2 + 9 + val3) (Figure 17).

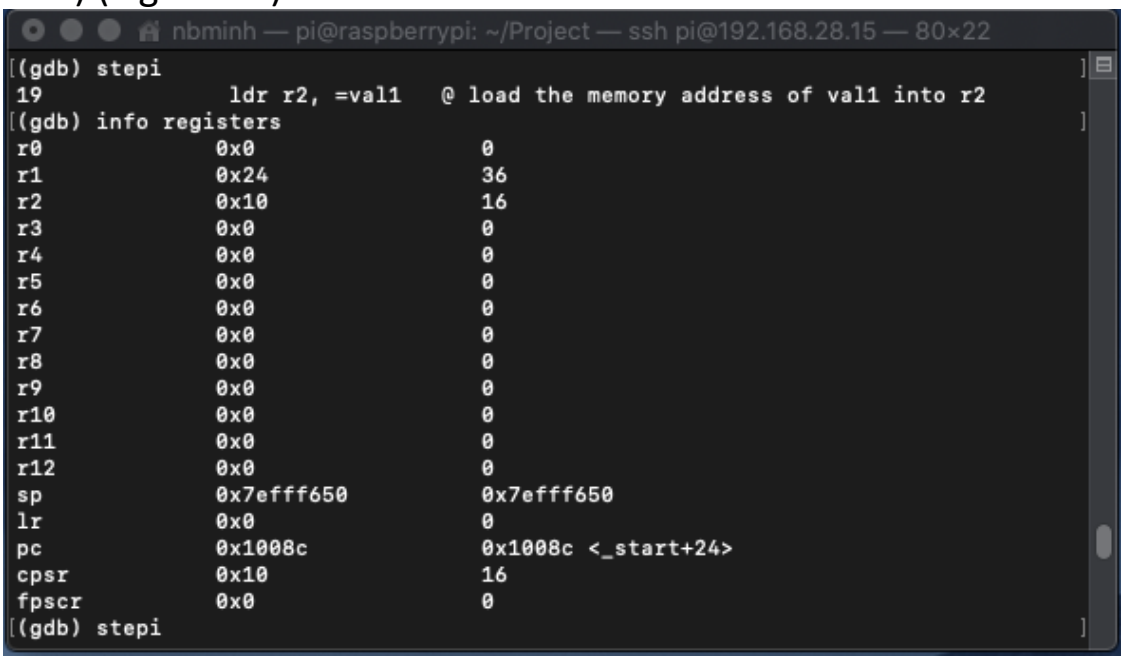

Figure 17

After executing line 19, I had the address of val1 in r2 which is 0x200ac. Reading from this address gave me the value 6 in hex (also 6 in decimal) (Figure 18).

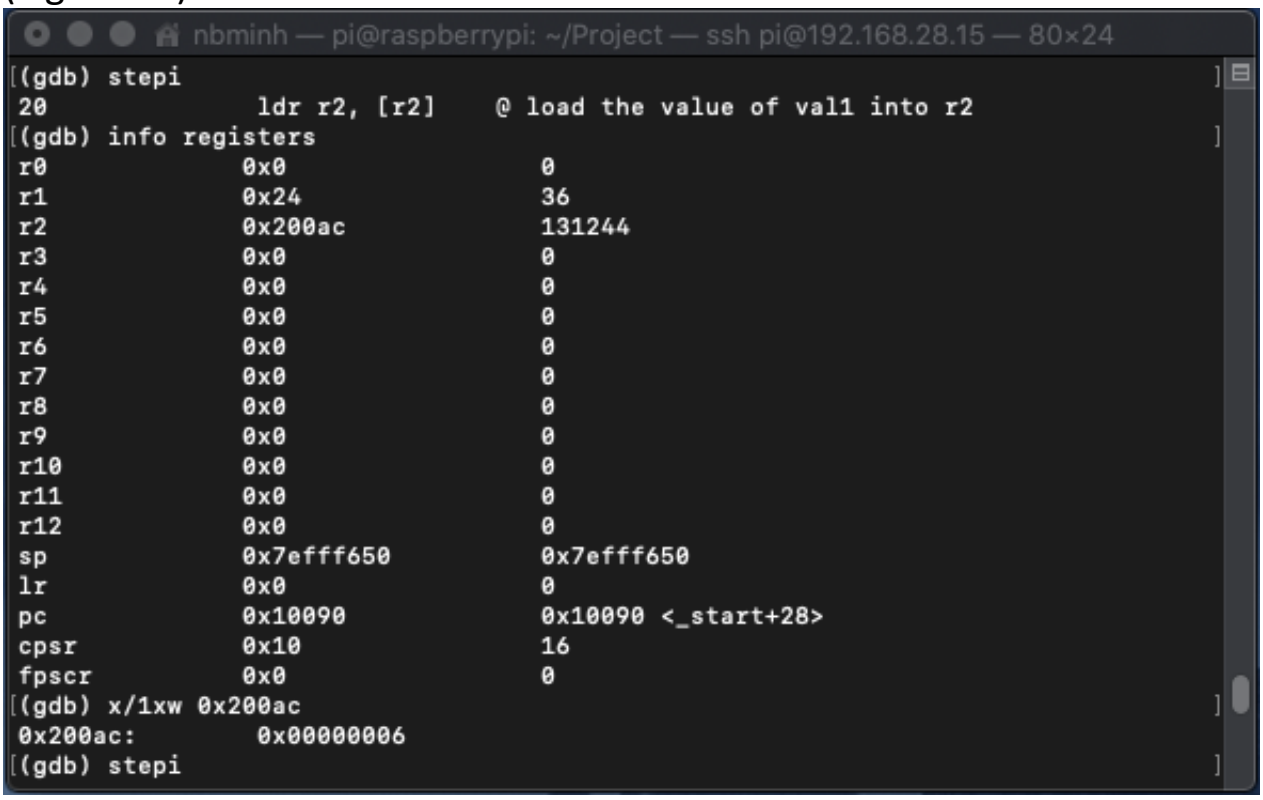

Figure 18

The result of the arithmetic expression is 30 and it was stored in a register (register r1) as required (Figure 19).

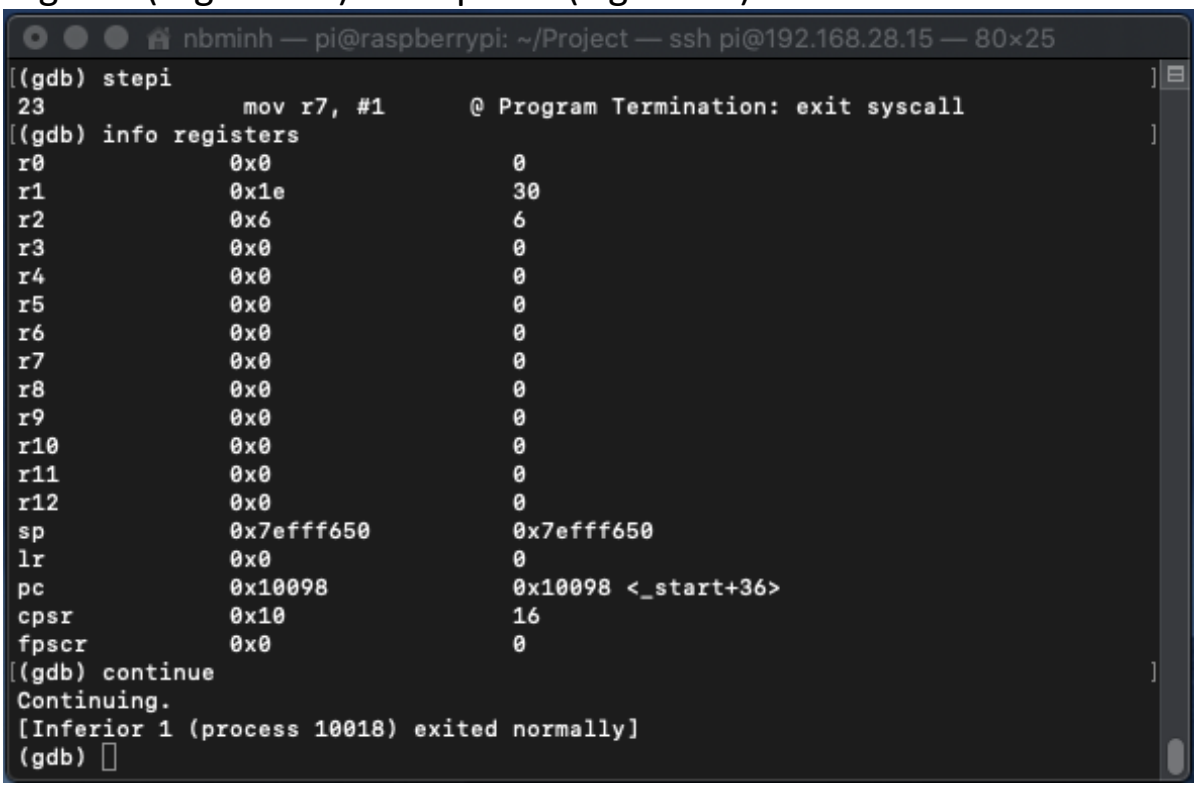

Figure 19

And here is what my Project folder in Raspberry looks like. Noted that the selected files are all the files that were created in this project (Figure 20).

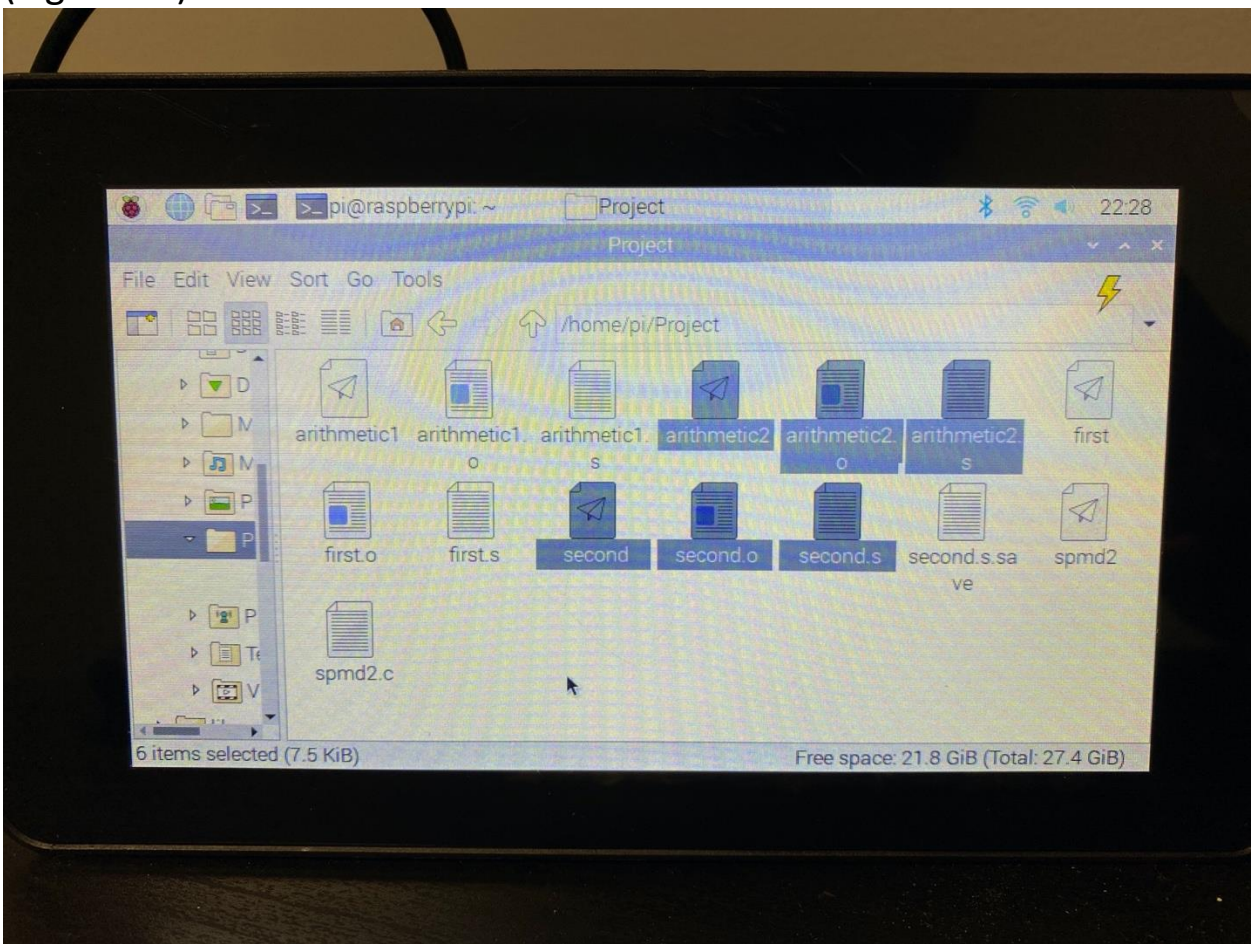

Figure 20