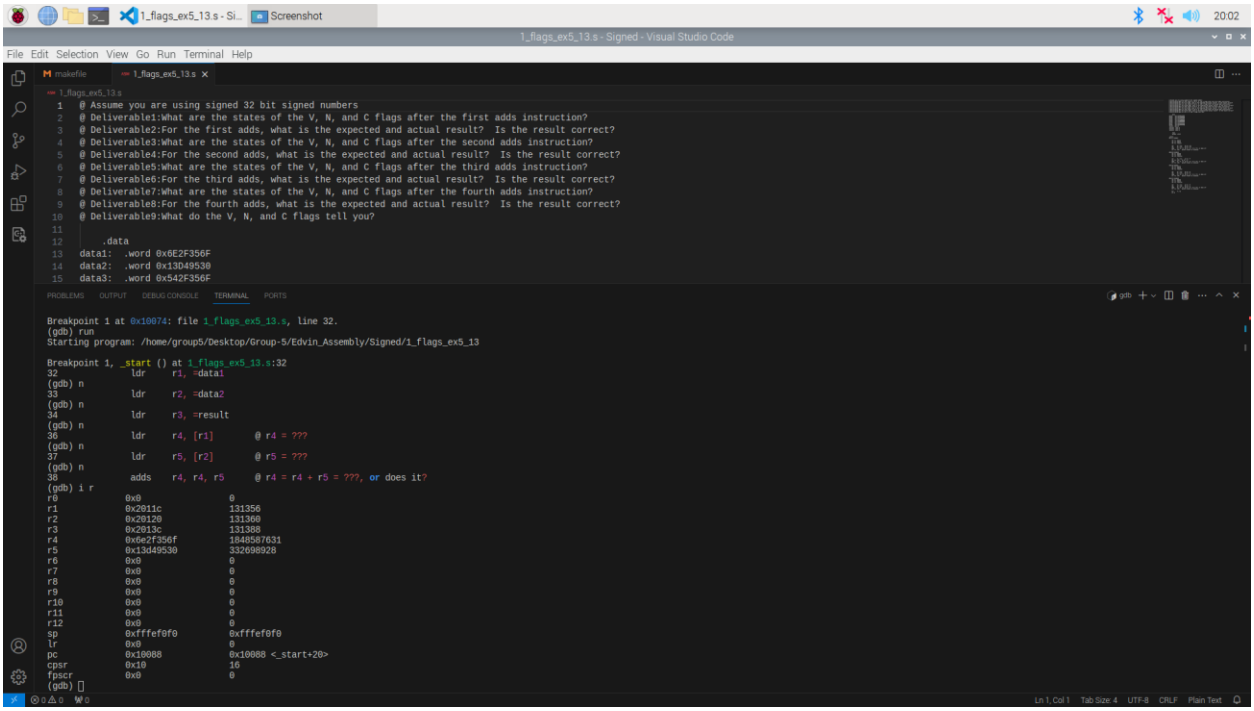
1. Flags
    a. What are the states of the V, N, and C flags after the first adds instruction?

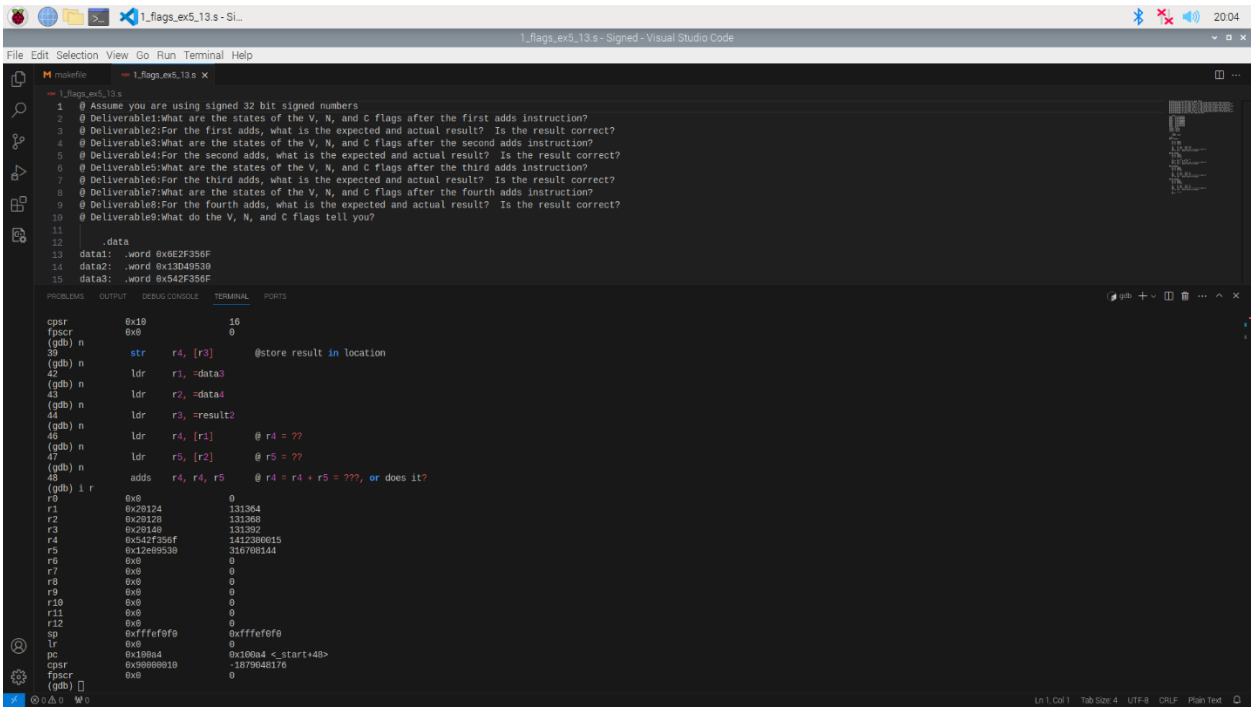
    The flags are all set to zero.



    b. For the first adds, what is the expected and actual result? Is the result correct?

    You would expect to see the sum to be 262,744, but is given as the much higher 1848587631.

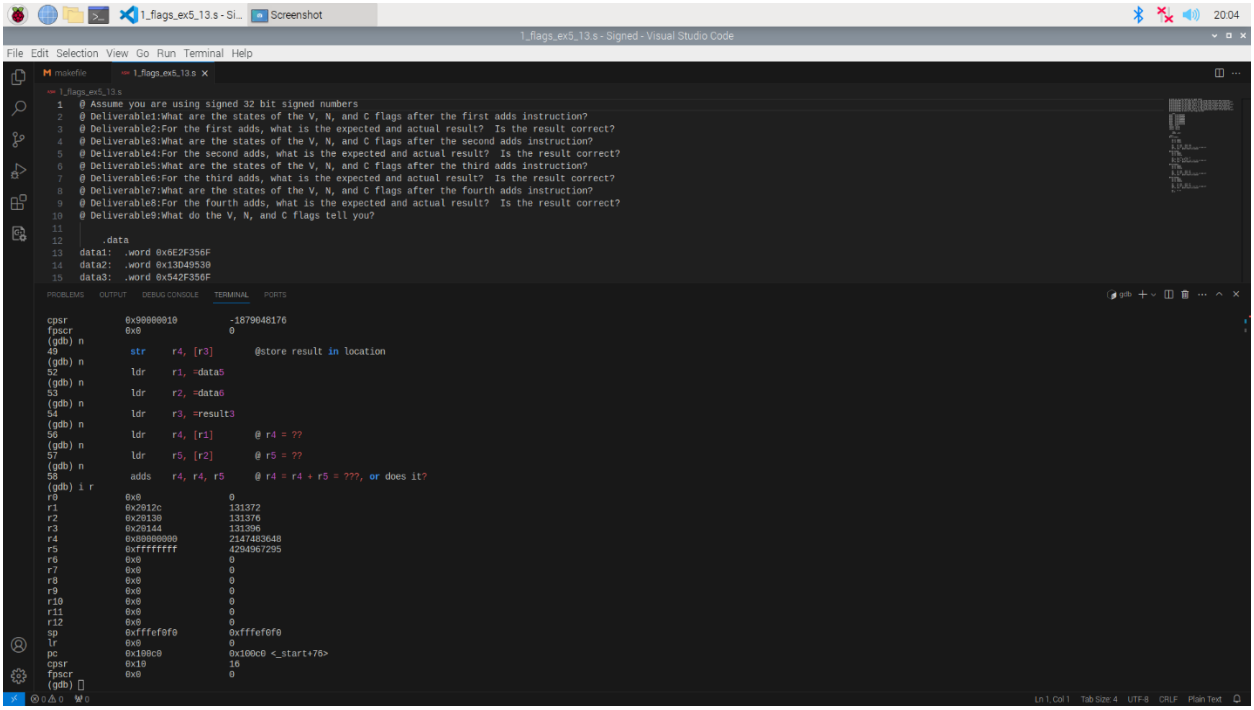
    c. What are the states of the V, N, and C flags after the second adds instruction?



d. For the second adds, what is the expected and actual result? Is the result correct?

The expected result is 262732, but the given result is 1412380015. This is not correct.

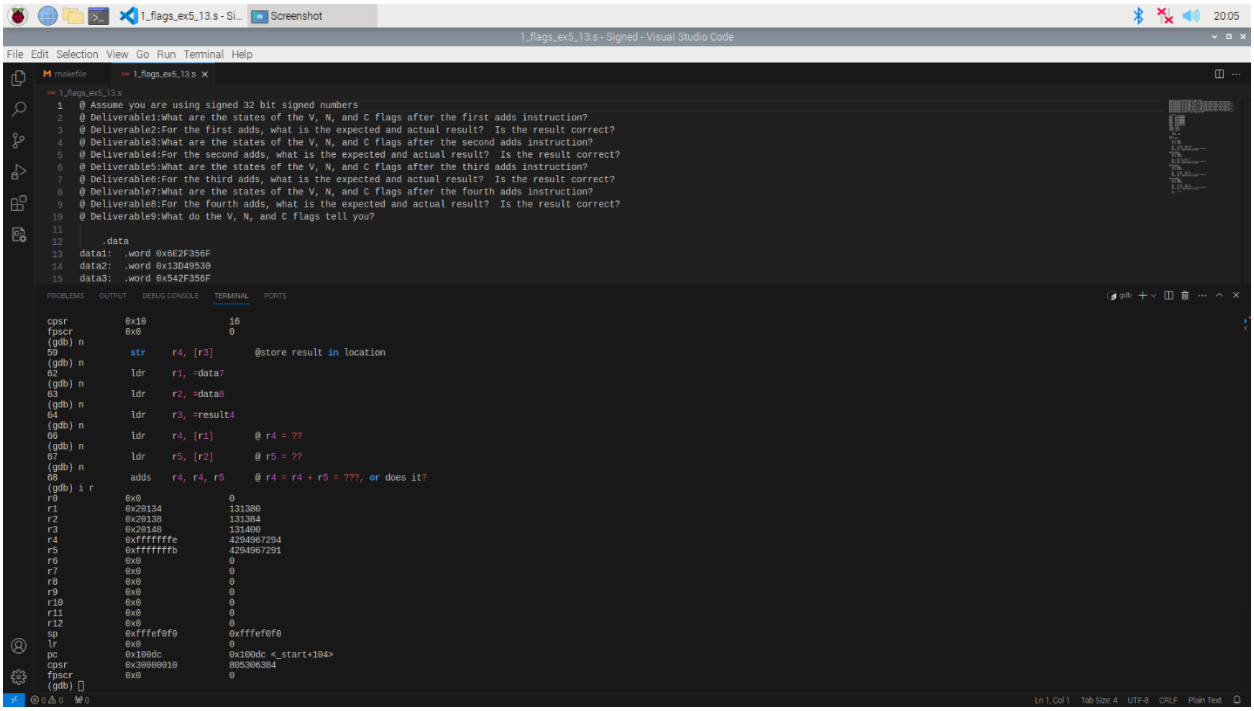
e. What are the states of the V, N, and C flags after the third adds instruction?



f. For the third adds, what is the expected and actual result? Is the result correct?

The expected result is 262,748, but is instead the much higher 2147483648.

g. What are the states of the V, N, and C flags after the fourth adds instruction?



h. For the fourth adds, what is the expected and actual result? Is the result correct?

Expected: 262764

Actual: 4294967294

i. What do the V, N, and C flags tell you?

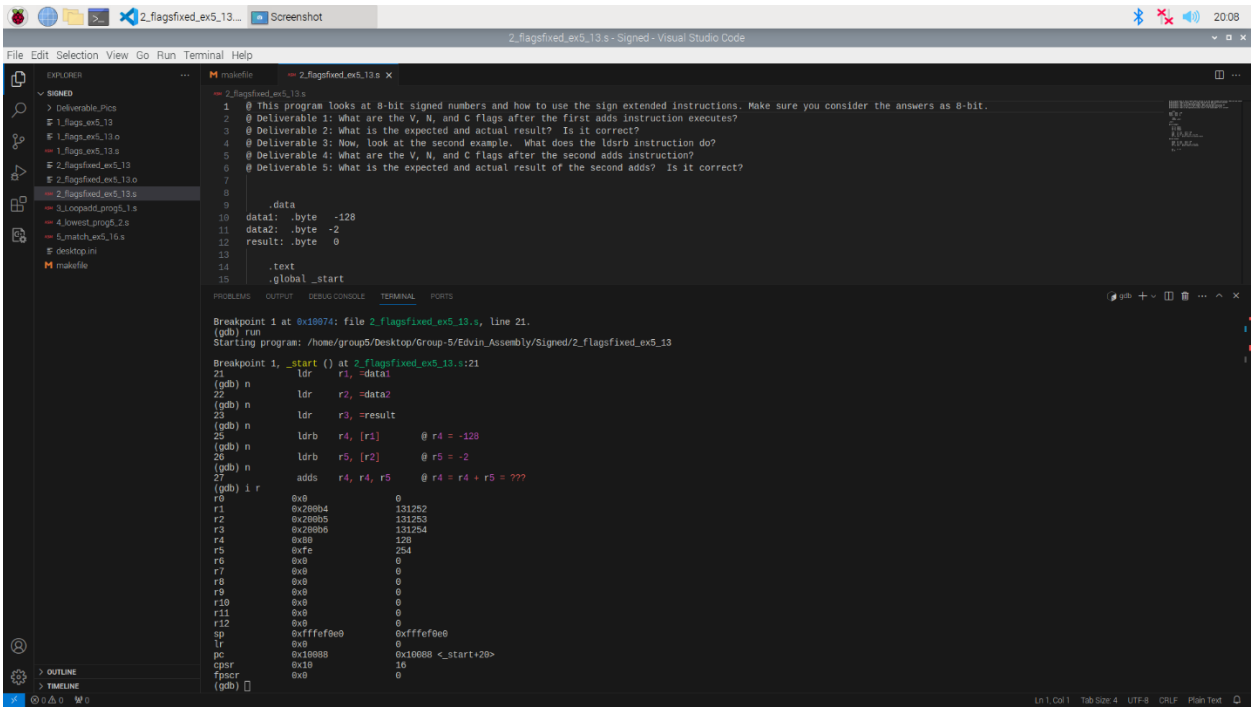
N is the negative flag, being set to 1 when negative and 0 when not.

V is the overflow indicator.

C is the carry indicator.

2. Flags Fixed

a. What are the V, N, and C flags after the first adds instruction executes?

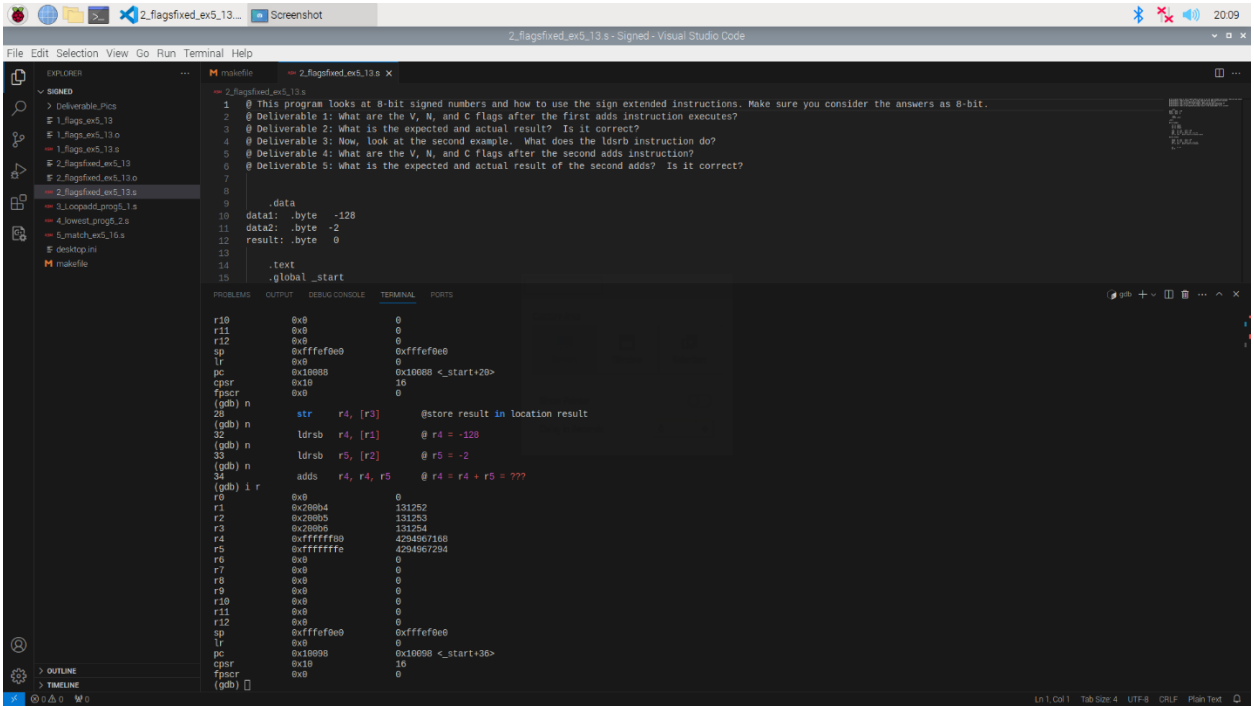


b. What is the expected and actual result? Is it correct?

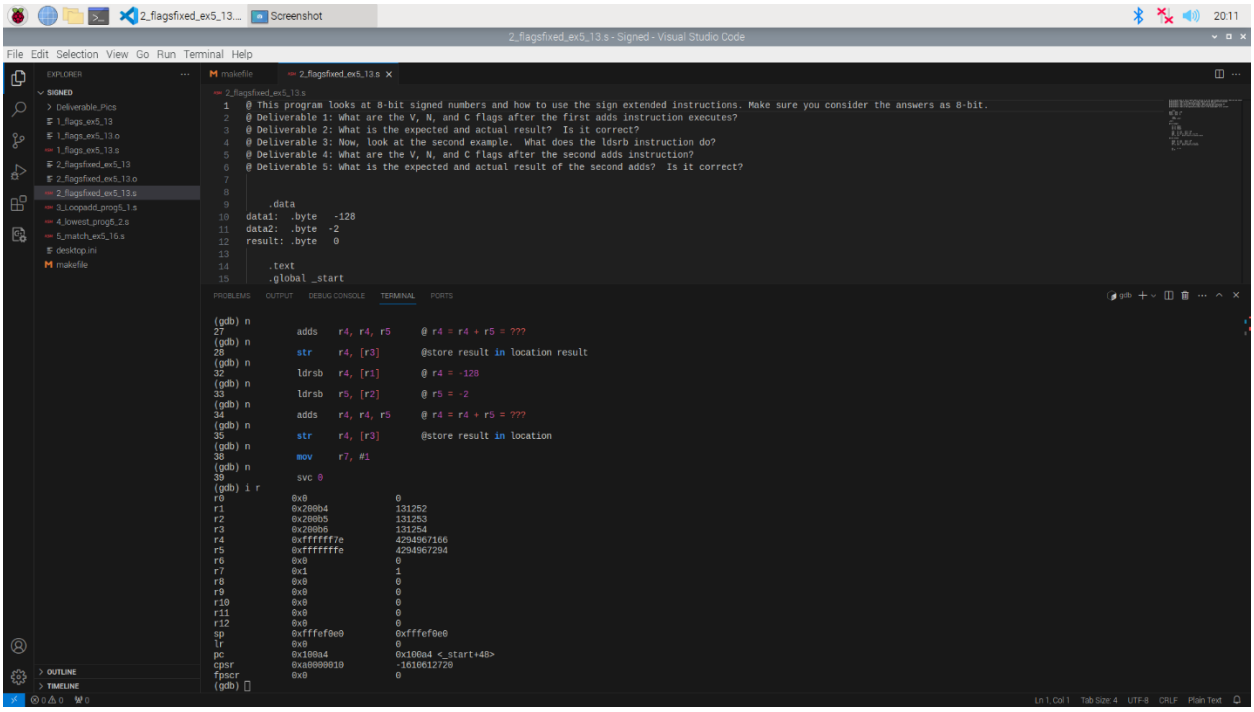
The expected result is -130, but is given as 128.

c. Now, look at the second example. What does the ldsrb instruction do?

The LDSRB loads a double-word instruction into the registers.



d. What are the V, N, and C flags after the second adds instruction?



e. What is the expected and actual result of the second adds? Is it correct?

The actual result is 4294967166, instead of the expected 262505

3. Loop Add

a. What is the expected result of the sum of the numbers?

b. Describe how the program works.

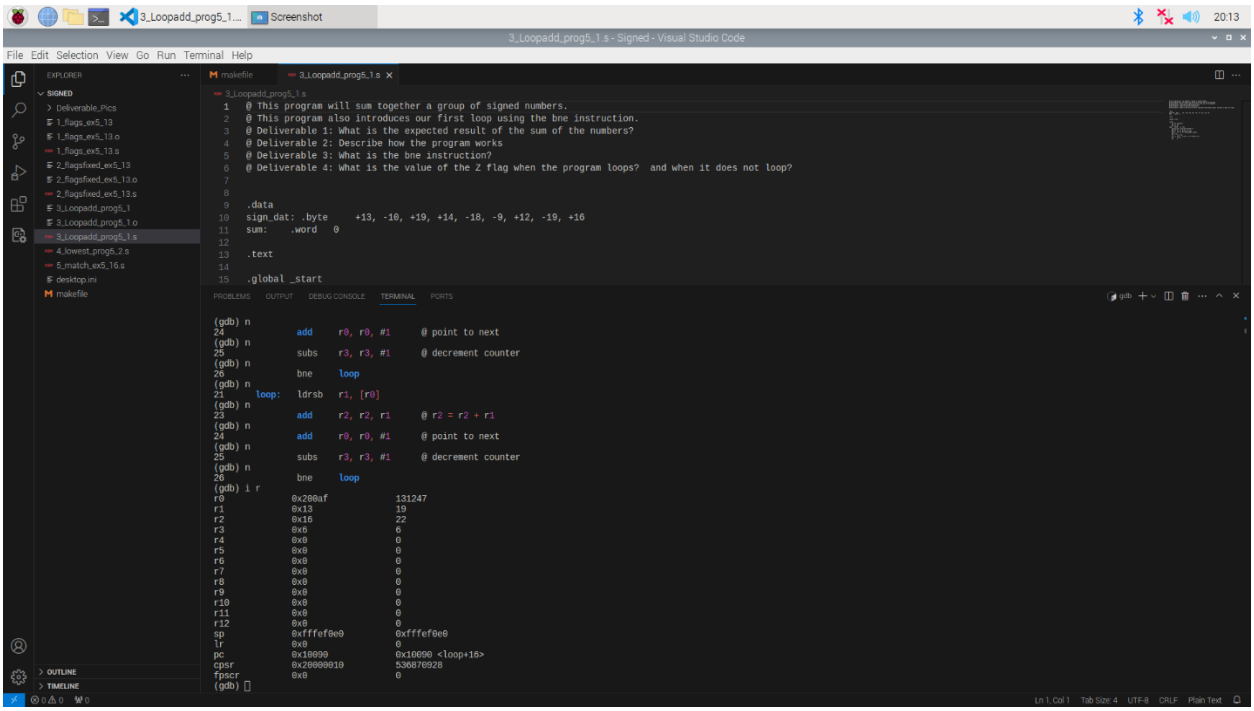
The program sums the signed values in the array by adding each value to an accumulator.

c. What is the bne instruction?

The BNE is a call-back instruction which calls back to a previous branch. This makes it useful for creating loops.

d. What is the value of the Z flag when the program loops? and when it does not loop?

The z-flag is 0 when it loops and stops looping when the counter at register 3 reaches zero and trips the z-flag.



4. Lowest Progs
    a. Describe how the program works. Make sure you describe the pointer and counter function. For each of 2-5, make sure you include flags in your discussion of "how does it work"
    b. What is the beq instruction and how does it work?

The BEQ tests if the z flag is set to 1, indicating the previous arithmetic resulted in zero. If it is the branch is taken, if not the instruction following BEQ is taken.

    c. What is the cmp instruction and how does it work?

CMP subtracts the second operand from the first operand but does not store the result, only updates the appropriate flags based on the results.

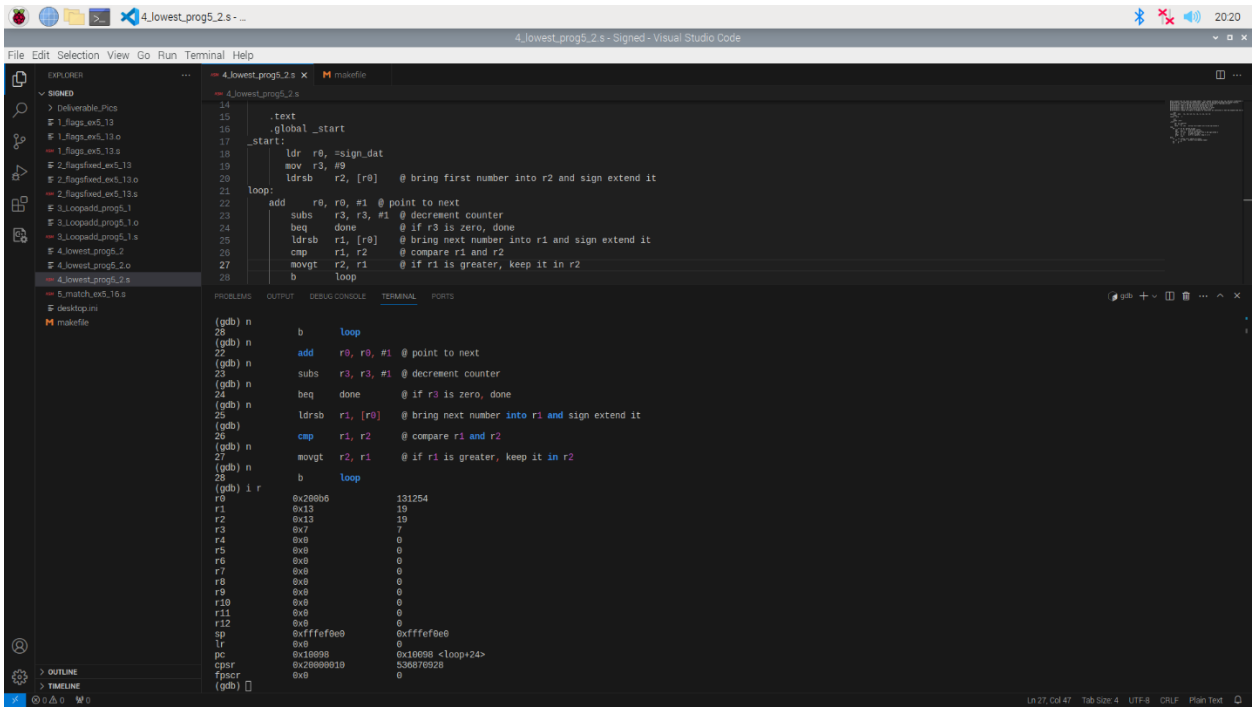
    d. What is the movlt instruction and how does it work?

MOVLT checks the negative flag and overflow flag. If negative is set but overflow isn't the MOVLT instruction is taken. Otherwise it is skipped.

    e. What is the b instruction and how does it work?

The B instruction causes the program to jump to a specified target address.

    f. Change the program by changing the conditional mov instruction so that the program finds the largest of the signed numbers. Show your final code.



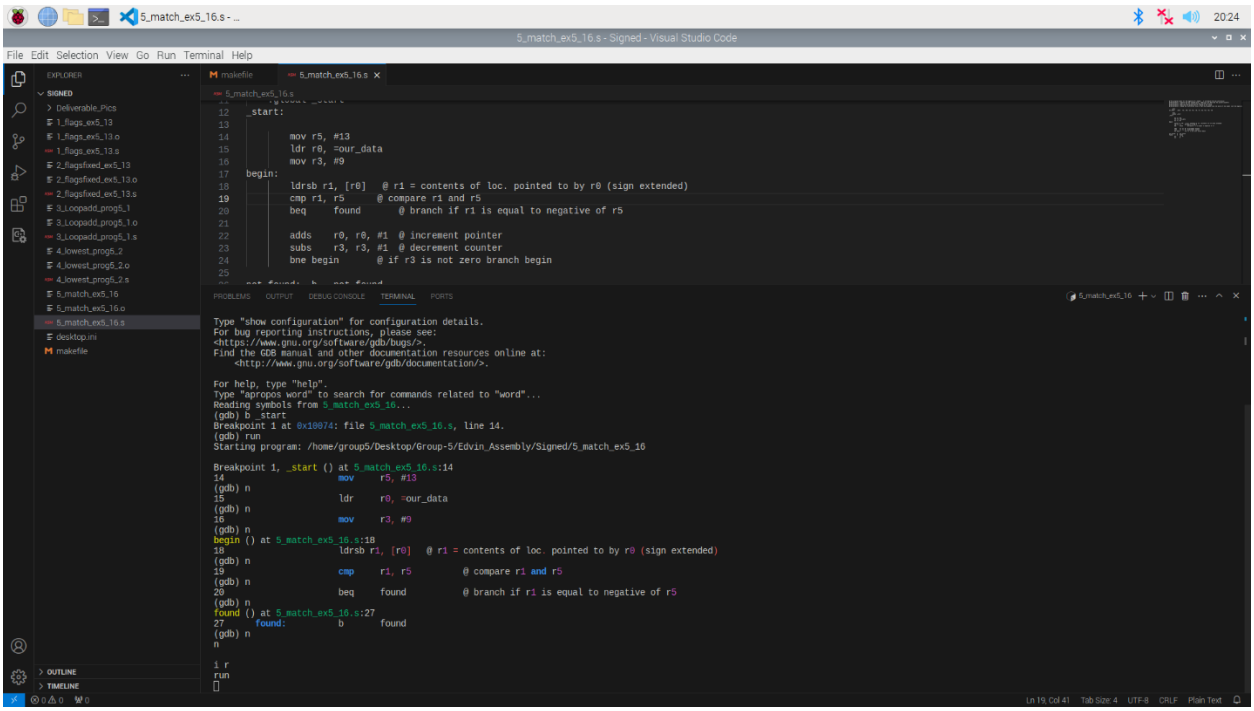
5. Match
    a. Explain how the program works. Make sure you describe the pointer/counters
    b. What is the cmn instruction and how does it work?

    The CMN is similar to CMP in that it subtracts two operands and updates the register flags without saving the result. The updated flags are N, Z, V, and C flags.

    c. What does the program do if a match is not found?

    IF a match is not found the programs increments a counter and loops back to check. When a match is found the program ends.

    d. Change one instruction only so that the program finds the match of the number (not the negative of the number). Show your program.


5_match_ex5_16.s - ...
2024
5_match_ex5_16.s - Signed - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
SIGNED
Deliverable_Pics
1_flags_ex5_13
1_flags_ex5_13.o
1_flags_ex5_13.s
2_flagsfixed_ex5_13
2_flagsfixed_ex5_13.o
2_flagsfixed_ex5_13.s
3_Loopadd_prog5_1
3_Loopadd_prog5_1.o
3_Loopadd_prog5_1.s
4_lowest_prog5_2
4_lowest_prog5_2.o
4_lowest_prog5_2.s
5_match_ex5_16
5_match_ex5_16.o
5_match_ex5_16.s
desktop.ini
makefile
OUTLINE
TIMELINE
makefile
5_match_ex5_16.s
5_match_ex5_16.s
12 _start:
13
14 mov r5, #13
15 ldr r0, =our_data
16 mov r3, #9
17 begin:
18 ldrsb r1, [r0] @ r1 = contents of loc. pointed to by r0 (sign extended)
19 cmp r1, r5 @ compare r1 and r5
20 beq found @ branch if r1 is equal to negative of r5
21
22 adds r0, r0, #1 @ increment pointer
23 subs r3, r3, #1 @ decrement counter
24 bne begin @ if r3 is not zero branch begin
25
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
5_match_ex5_16
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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"...
Reading symbols from 5_match_ex5_16...
(gdb) b _start
Breakpoint 1 at 0x10074: file 5_match_ex5_16.s, line 14.
(gdb) run
Starting program: /home/group5/Desktop/Group-5/Edvin_Assembly/Signed/5_match_ex5_16
Breakpoint 1, _start () at 5_match_ex5_16.s:14
14 mov r5, #13
(gdb) n
15 ldr r0, =our_data
(gdb) n
16 mov r3, #9
(gdb) n
begin () at 5_match_ex5_16.s:18
18 ldrsb r1, [r0] @ r1 = contents of loc. pointed to by r0 (sign extended)
(gdb) n
19 cmp r1, r5 @ compare r1 and r5
(gdb) n
20 beq found @ branch if r1 is equal to negative of r5
(gdb) n
found () at 5_match_ex5_16.s:27
27 found: b found
(gdb) n
n
i r
run
Ln 19, Col 41 Tab Size: 4 UTF-8 CRLF Plain Text