Name: Cameron Peterson-Zopf **Student ID:** 57999719

SMALL ASSIGNMENT 8

This is a group assignment (up to 3 students/per group). You can also complete this assignment individually.

Discussion on a high level with your colleagues is encouraged. Make sure the work submitted is your own (or your group). When in doubt, ask a TA or the instructor. If you are not sure what constitutes academic dishonesty, please refer to the AISC web site: https://aisc.uci.edu/.

You can fill out your answers below in text, paste screenshots, and/or include images (make sure the image is right side up & legible).

This homework covers:

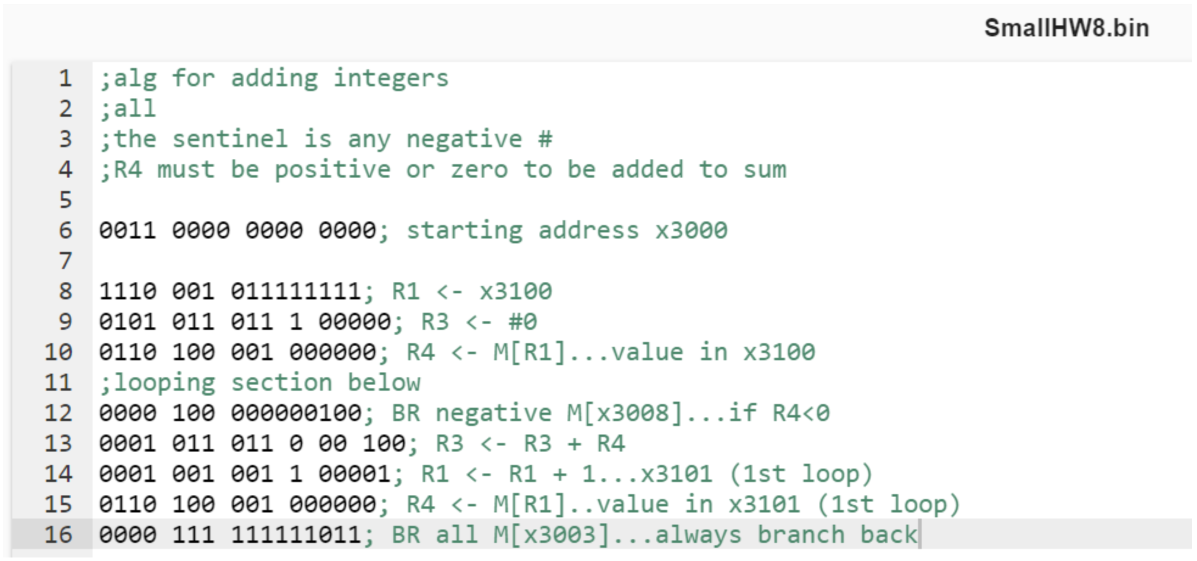
- LC-3 Simulator Debugging

AISC

Please initial here to indicate you understand UCI's Academic Integrity Policy and confirm that this is your own work (or your own group's work) you are submitting (this counts for points): CPZ

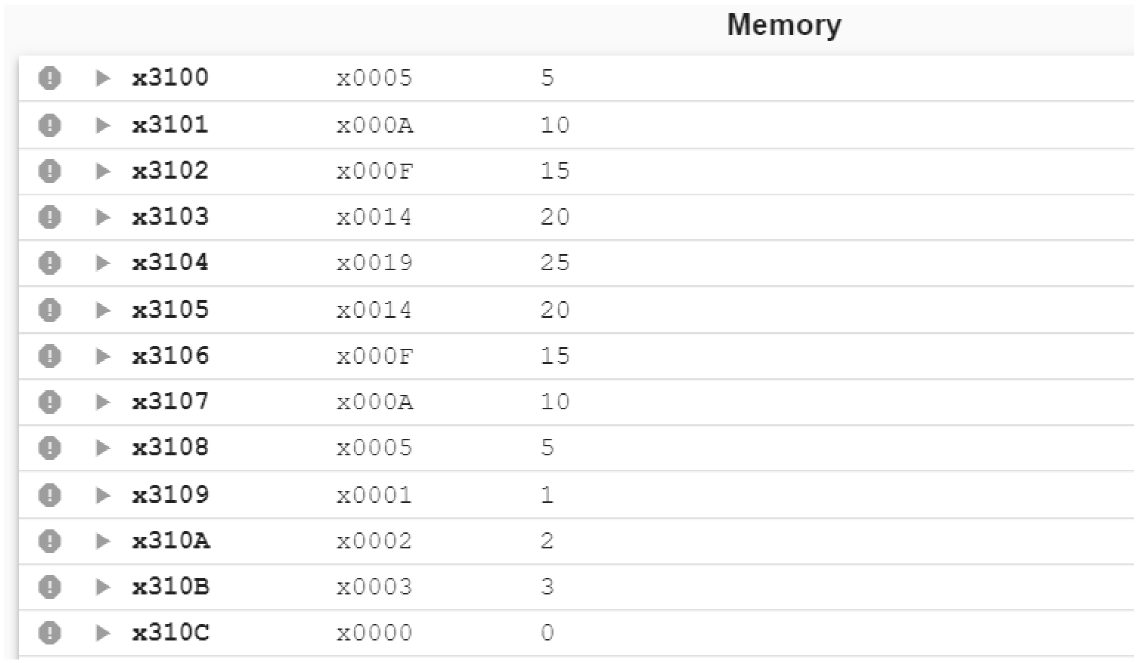
1. SCREENSHOT RECREATING PROBLEM

Below is the initial code written with comments:


SmallHW8.bin

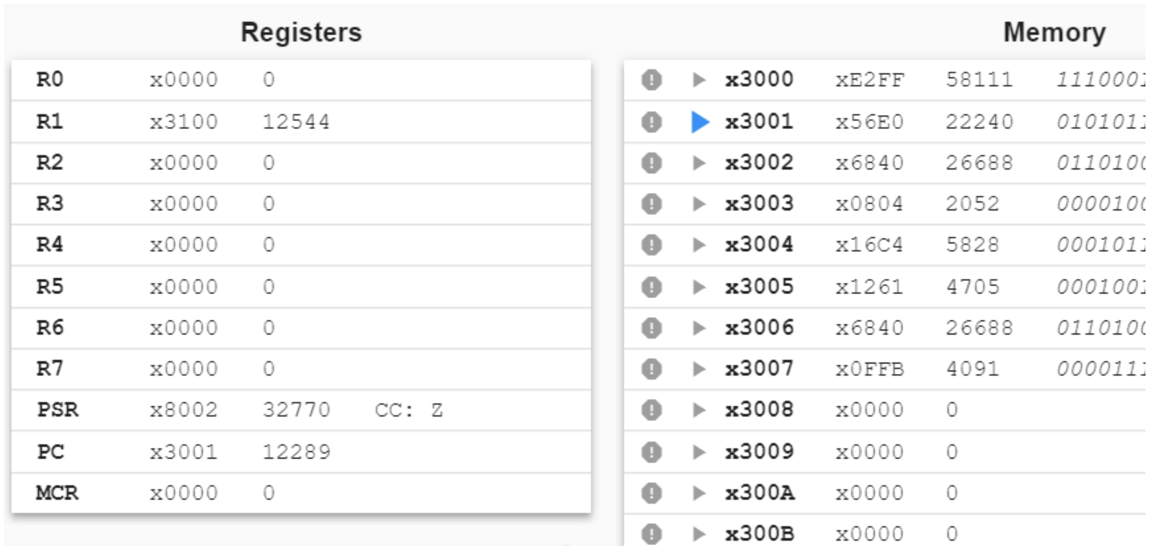
```
;alg for adding integers
;all
;the sentinel is any negative #
;R4 must be positive or zero to be added to sum

0011 0000 0000 0000; starting address x3000

1110 001 011111111; R1 <- x3100
0101 011 011 1 00000; R3 <- #0
0110 100 001 000000; R4 <- M[R1]...value in x3100
;looping section below
0000 100 000000100; BR negative M[x3008]...if R4<0
0001 011 011 0 00 100; R3 <- R3 + R4
0001 001 001 1 00001; R1 <- R1 + 1...x3101 (1st loop)
0110 100 001 000000; R4 <- M[R1]..value in x3101 (1st loop)
0000 111 111111011; BR all M[x3003]...always branch back
```

Below are the hardcoded memory values:



Memory		
x3100	x0005	5
x3101	x000A	10
x3102	x000F	15
x3103	x0014	20
x3104	x0019	25
x3105	x0014	20
x3106	x000F	15
x3107	x000A	10
x3108	x0005	5
x3109	x0001	1
x310A	x0002	2
x310B	x0003	3
x310C	x0000	0

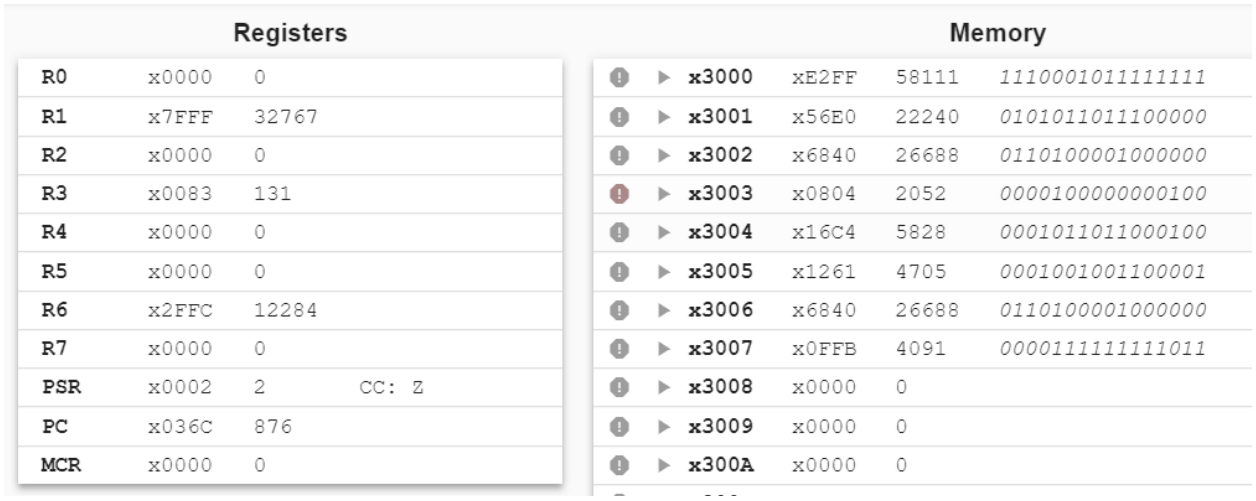
The initial value for R1 is x3100 as expected:



Registers			
R0	x0000	0	
R1	x3100	12544	
R2	x0000	0	
R3	x0000	0	
R4	x0000	0	
R5	x0000	0	
R6	x0000	0	
R7	x0000	0	
PSR	x8002	32770	CC: Z
PC	x3001	12289	
MCR	x0000	0	

Memory			
x3000	xE2FF	58111	1110001
x3001	x56E0	22240	0101011
x3002	x6840	26688	0110100
x3003	x0804	2052	0000100
x3004	x16C4	5828	0001011
x3005	x1261	4705	0001001
x3006	x6840	26688	0110100
x3007	x0FFB	4091	0000111
x3008	x0000	0	
x3009	x0000	0	
x300A	x0000	0	
x300B	x0000	0	

The final value for the sum is:



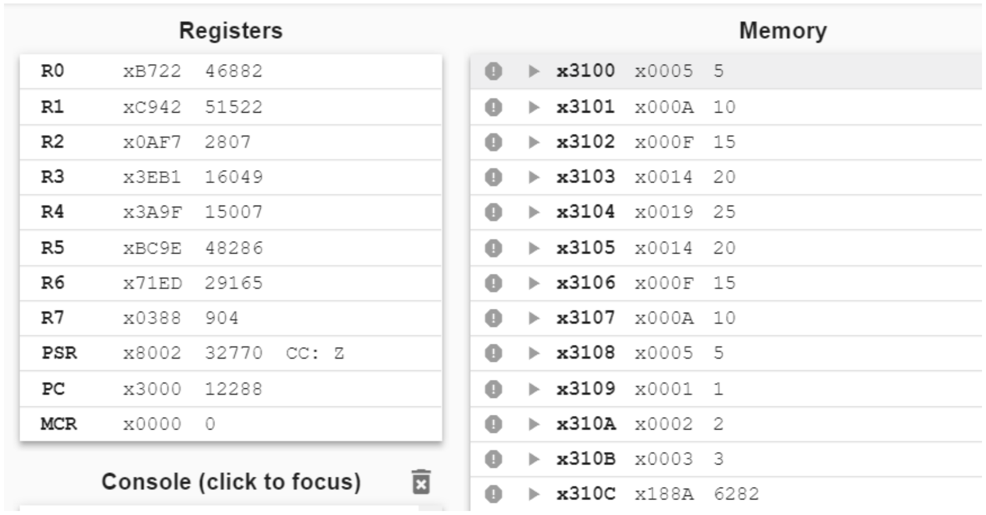
Registers			
R0	x0000	0	
R1	x7FFF	32767	
R2	x0000	0	
R3	x0083	131	
R4	x0000	0	
R5	x0000	0	
R6	x2FFC	12284	
R7	x0000	0	
PSR	x0002	2	CC: Z
PC	x036C	876	
MCR	x0000	0	

Memory			
x3000	xE2FF	58111	1110001011111111
x3001	x56E0	22240	0101011011100000
x3002	x6840	26688	0110100001000000
x3003	x0804	2052	0000100000000100
x3004	x16C4	5828	0001011011000100
x3005	x1261	4705	0001001001100001
x3006	x6840	26688	0110100001000000
x3007	x0FFB	4091	0000111111111011
x3008	x0000	0	
x3009	x0000	0	
x300A	x0000	0	

2. SCREENSHOT SHOWING LINE NUMBERS WHERE PROBLEM OCCURS

Randomize Machine:

I have now randomized the machine and inputted in the hardcoded values:

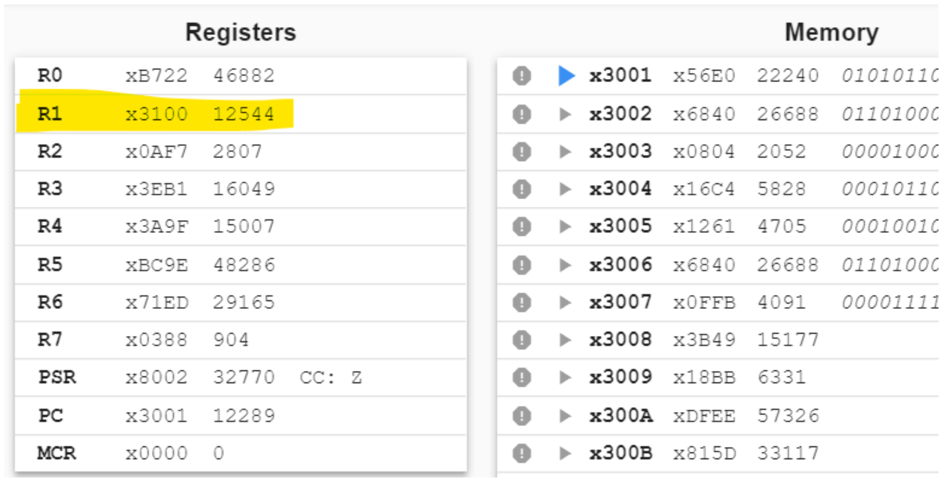


Registers			
R0	xB722	46882	
R1	xC942	51522	
R2	x0AF7	2807	
R3	x3EB1	16049	
R4	x3A9F	15007	
R5	xBC9E	48286	
R6	x71ED	29165	
R7	x0388	904	
PSR	x8002	32770	CC: Z
PC	x3000	12288	
MCR	x0000	0	

Console (click to focus)

Memory		
x3100	x0005	5
x3101	x000A	10
x3102	x000F	15
x3103	x0014	20
x3104	x0019	25
x3105	x0014	20
x3106	x000F	15
x3107	x000A	10
x3108	x0005	5
x3109	x0001	1
x310A	x0002	2
x310B	x0003	3
x310C	x188A	6282

The initial value for R1 is x3100 as expected:



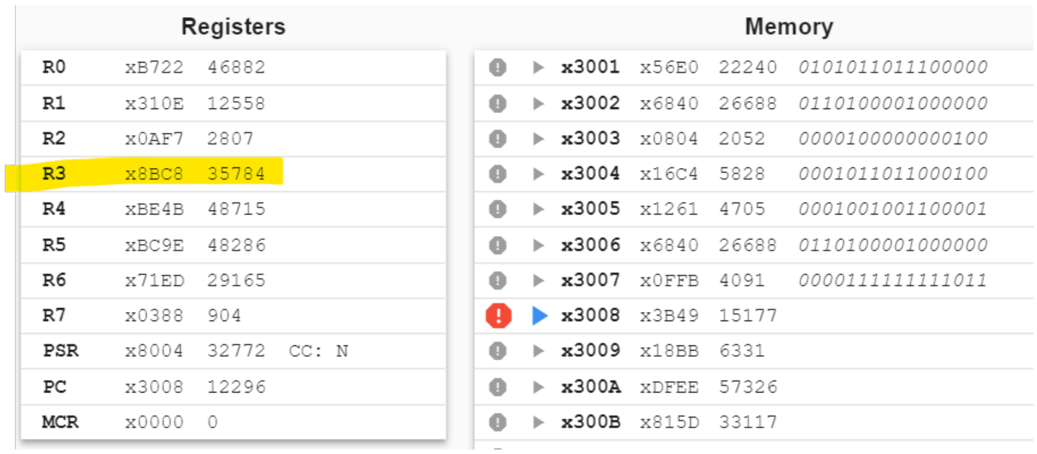
Registers

R0	xB722	46882	
R1	x3100	12544	
R2	x0AF7	2807	
R3	x3EB1	16049	
R4	x3A9F	15007	
R5	xBC9E	48286	
R6	x71ED	29165	
R7	x0388	904	
PSR	x8002	32770	CC: Z
PC	x3001	12289	
MCR	x0000	0	

Memory

x3001	x56E0	22240	01010110
x3002	x6840	26688	01101000
x3003	x0804	2052	00001000
x3004	x16C4	5828	00010110
x3005	x1261	4705	00010010
x3006	x6840	26688	01101000
x3007	x0FFB	4091	00001111
x3008	x3B49	15177	
x3009	x18BB	6331	
x300A	xDFEE	57326	
x300B	x815D	33117	

The final sum value is not 131:



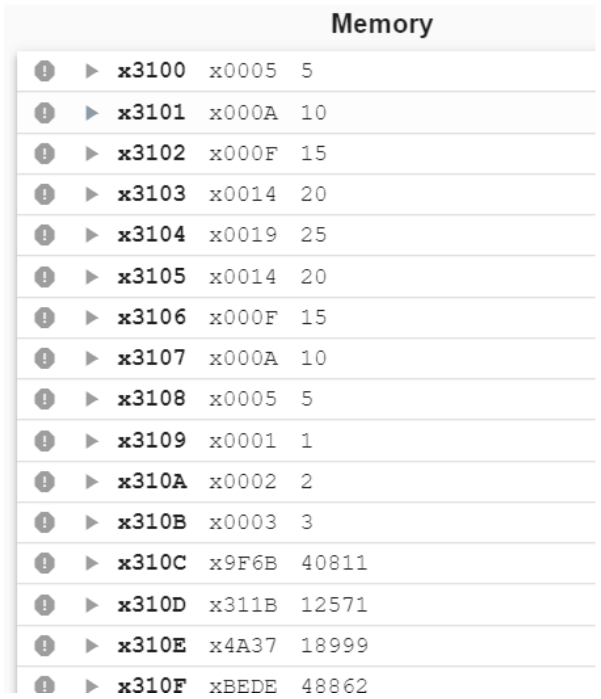
Registers

R0	xB722	46882	
R1	x310E	12558	
R2	x0AF7	2807	
R3	x8BC8	35784	
R4	xBE4B	48715	
R5	xBC9E	48286	
R6	x71ED	29165	
R7	x0388	904	
PSR	x8004	32772	CC: N
PC	x3008	12296	
MCR	x0000	0	

Memory

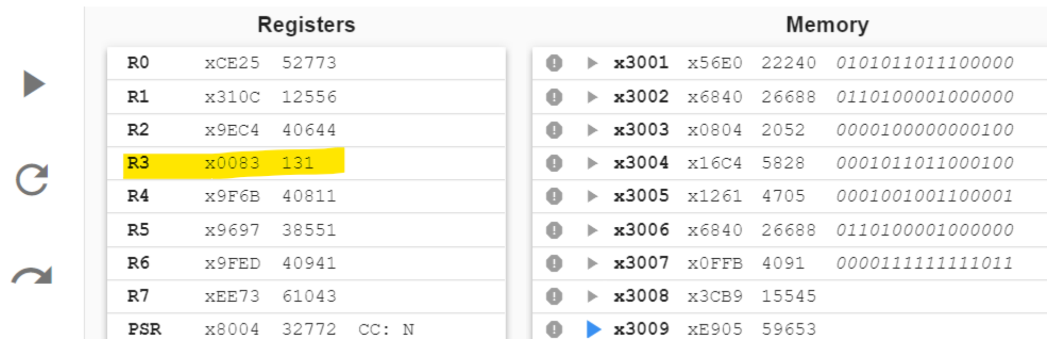
x3001	x56E0	22240	0101011011100000
x3002	x6840	26688	0110100001000000
x3003	x0804	2052	0000100000000100
x3004	x16C4	5828	0001011011000100
x3005	x1261	4705	0001001001100001
x3006	x6840	26688	0110100001000000
x3007	x0FFB	4091	0000111111111011
x3008	x3B49	15177	
x3009	x18BB	6331	
x300A	xDFEE	57326	
x300B	x815D	33117	

The second randomization worked to get 131:



Memory

x3100	x0005	5
x3101	x000A	10
x3102	x000F	15
x3103	x0014	20
x3104	x0019	25
x3105	x0014	20
x3106	x000F	15
x3107	x000A	10
x3108	x0005	5
x3109	x0001	1
x310A	x0002	2
x310B	x0003	3
x310C	x9F6B	40811
x310D	x311B	12571
x310E	x4A37	18999
x310F	xBEDE	48862



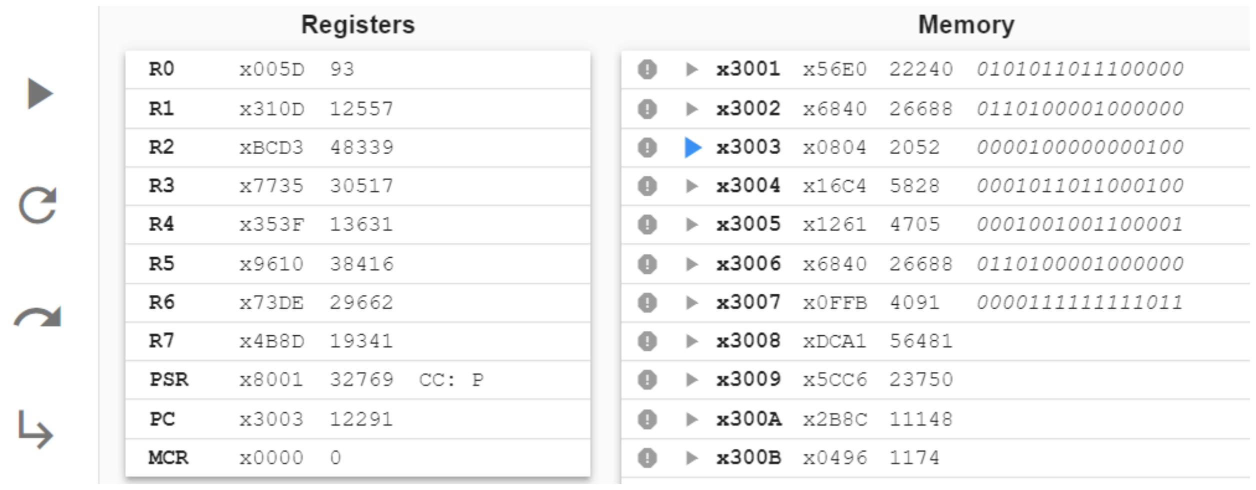
Registers

R0	xCE25	52773	
R1	x310C	12556	
R2	x9EC4	40644	
R3	x0083	131	
R4	x9F6B	40811	
R5	x9697	38551	
R6	x9FED	40941	
R7	xEE73	61043	
PSR	x8004	32772	CC: N

Memory

x3001	x56E0	22240	0101011011100000
x3002	x6840	26688	0110100001000000
x3003	x0804	2052	0000100000000100
x3004	x16C4	5828	0001011011000100
x3005	x1261	4705	0001001001100001
x3006	x6840	26688	0110100001000000
x3007	x0FFB	4091	0000111111111011
x3008	x3CB9	15545	
x3009	xE905	59653	

In another case where it didn't work, we see that when we get 131, it doesn't stop, but instead keeps going since the branch at x3003 is only checking if the value is negative, not if we are in the range of number we wanted to count (from x3100 to x310B).



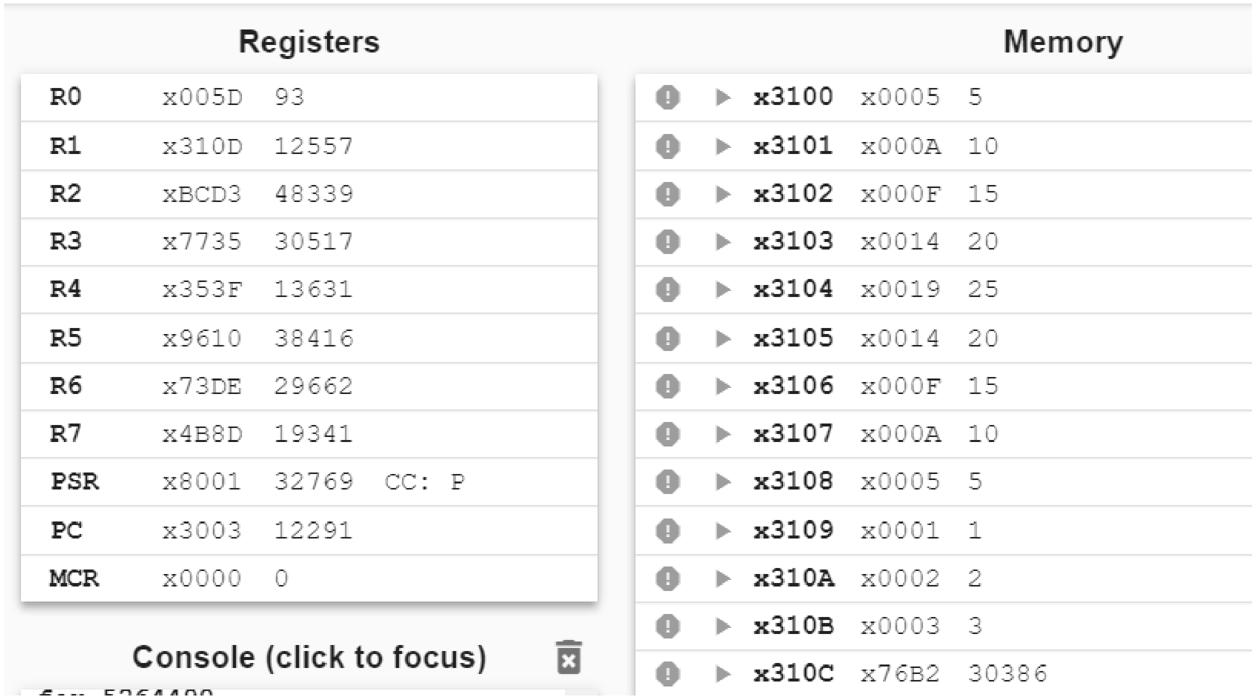
Registers

R0	x005D	93	
R1	x310D	12557	
R2	xBCD3	48339	
R3	x7735	30517	
R4	x353F	13631	
R5	x9610	38416	
R6	x73DE	29662	
R7	x4B8D	19341	
PSR	x8001	32769	CC: P
PC	x3003	12291	
MCR	x0000	0	

Memory

x3001	x56E0	22240	0101011011100000
x3002	x6840	26688	0110100001000000
x3003	x0804	2052	0000100000000100
x3004	x16C4	5828	0001011011000100
x3005	x1261	4705	0001001001100001
x3006	x6840	26688	0110100001000000
x3007	x0FFB	4091	0000111111111011
x3008	xDCA1	56481	
x3009	x5CC6	23750	
x300A	x2B8C	11148	
x300B	x0496	1174	

Now we have a large R3 value. This will continue summing as well. Looking at the values stored in x310C



Registers

R0	x005D	93
R1	x310D	12557
R2	xBCD3	48339
R3	x7735	30517
R4	x353F	13631
R5	x9610	38416
R6	x73DE	29662
R7	x4B8D	19341
PSR	x8001	32769 CC: P
PC	x3003	12291
MCR	x0000	0

Console (click to focus)

Memory

x3100	x0005	5
x3101	x000A	10
x3102	x000F	15
x3103	x0014	20
x3104	x0019	25
x3105	x0014	20
x3106	x000F	15
x3107	x000A	10
x3108	x0005	5
x3109	x0001	1
x310A	x0002	2
x310B	x0003	3
x310C	x76B2	30386

We see that it is also positive. Hence the break statement will include it. We see that it was added since the current value of R3 = 30517 – 30386 = 131 which is what we wanted for the sum.

3. DESCRIPTION OF PROBLEM AND WHY IT ONLY HAPPENS SOMETIMES

This problem occurs when the x310C value is positive, or when it is zero and then x310D is positive. We want the program to end when we reach x310C. This only happens sometimes, because when x310C is randomized to a negative value, then we get out of the loop. If x310C and subsequent address values are positive, we will add them to the sum. When we had initialized all values outside of the ones pertinent to the code to zero, they didn't effect the sum as they were zero, but the code also never stopping running as there was no negative address value.

4. PROPOSAL OF A FIX/MODIFICATION THAT WILL ADDRESS PROBLEM

Program Instead of having a branch when negative statement by itself, we should add an additional branch statement. The branch when negative checks whether values within x3100 to x310B are positive or negative. Now, if we know that these values will always be positive or zero, then we could remove that statement. If we want to design a program where possibly values in this range are negative and we want to stop the sum, then we would leave it. Given this problem says all values in this range are always positive, we can remove the branch statement. We will instead use a counter, with 12, where a branch statement will check if it is zero. If it is zero, then we will exit the loop, as we have counted all of the numbers in the specified range.