Lam Ngo
Final Project
Testing

**Note: Since there are a lot of screen shots, and this has gotten too long. You should go into my codes to see the implementation of my test. I will try to keep my screen shot labels as clear as possible.Also all these tests are done with a carry look ahead adder, so that makes sure my carry look ahead adder works (I also use it to increment PC).*

I. Test Basic R Type Instructions (since these are straight forward, I am only showing the results):

1. Professor Rieffel’s Test: *this program should put the values 0..7 into registers 0..7*

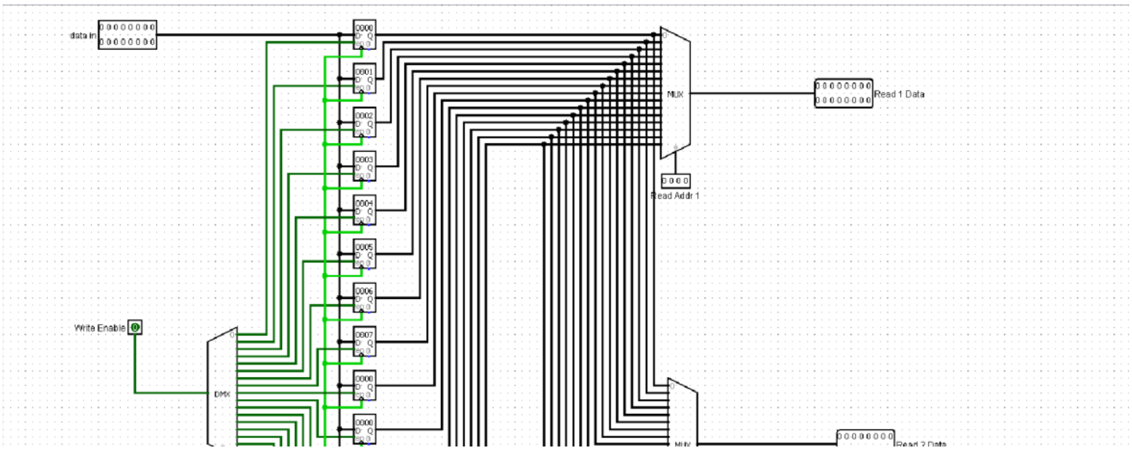


Figure 1: Basic R instructios test code: putting the values 0-7 into registers 0-7.

2. Lam Ngo Test:
# $2 should contain: 5
# $3 should contain: 3
# $4 should contain: 7
# $5 should contain: 2

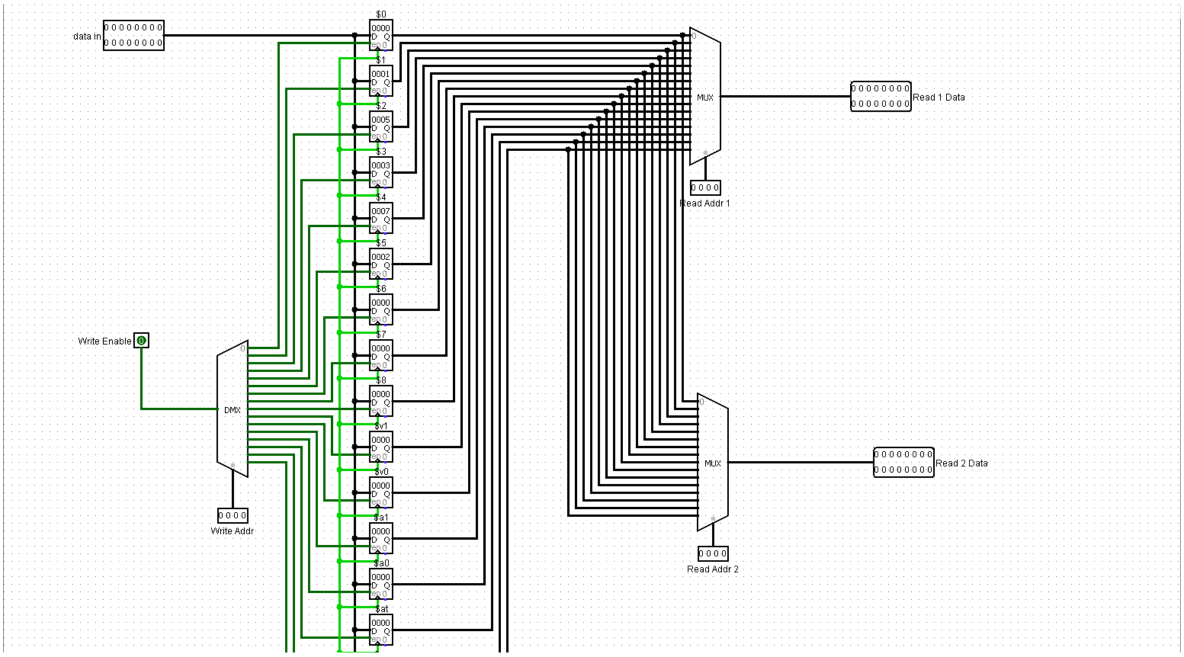


Figure 2: Lam Ngo’s test: Adding and subbing values into registers.

II. Test basic I-type Instructions (since these are straight forward, I am only showing the results):

1. Professor Rieffel’s test: *values 0..7 should be in memory locations 0. Values 7..0 should be in registers 0 … 7(switched)*

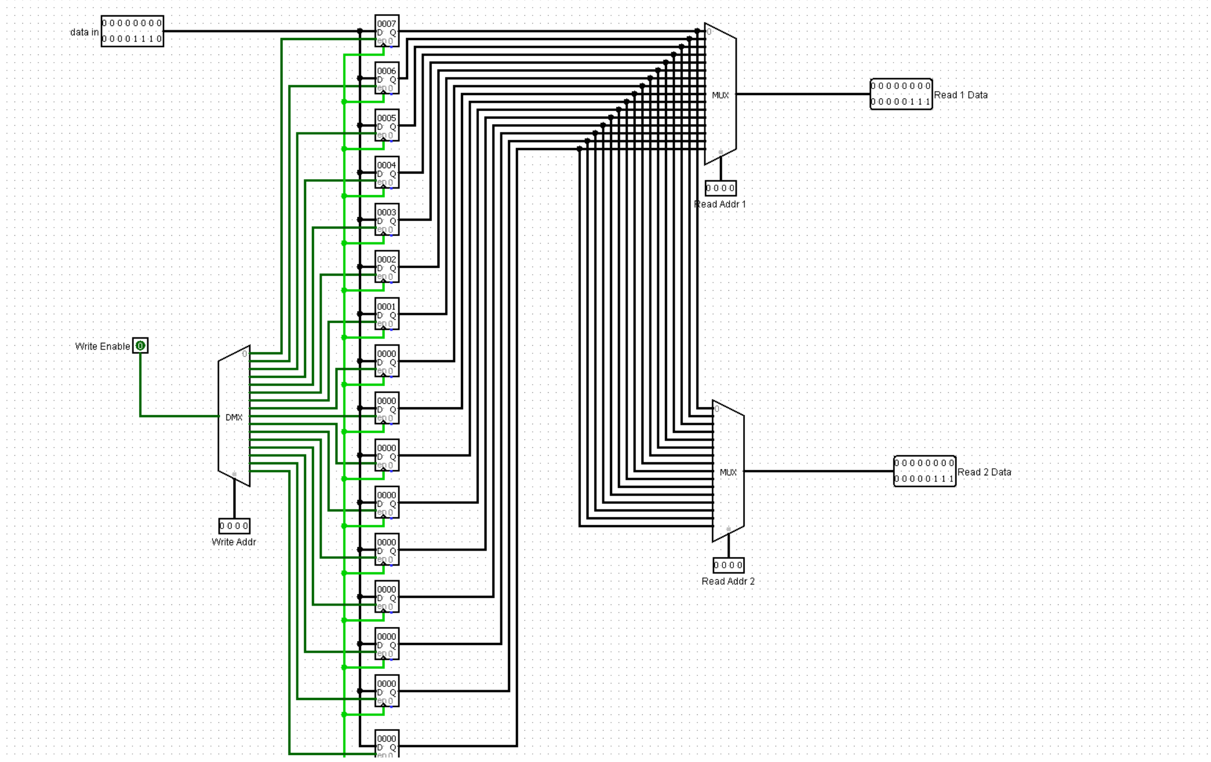
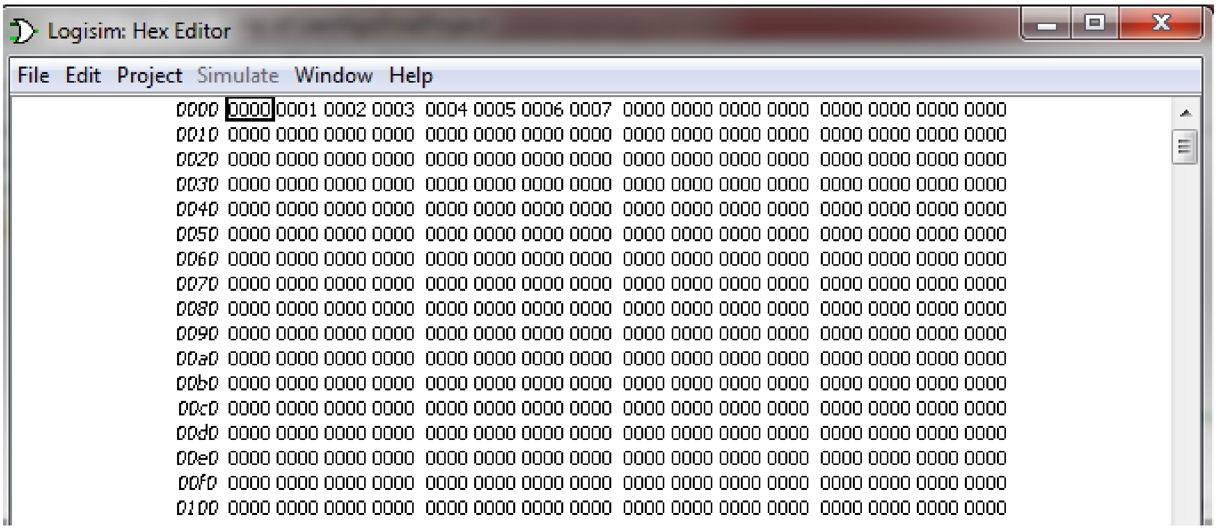


Figure 3: Values from 7 to 0 are in registers from 0 to 7.



```
0000 0000 0001 0002 0003  0004 0005 0006 0007  0000 0000 0000 0000  0000 0000 0000 0000
0010 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0020 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0030 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0040 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0050 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0060 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0070 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0080 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0090 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00a0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00b0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00c0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00d0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00e0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00f0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0100 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
```

Figure 4: Values from 0 to 7 are stored in memory location from 0 to 7.

2. Lam Ngo’s Test:

   a. First test: test subtract a negative number. Results should be the same as Professor Rieffel’ test. (Instead of addi $1 $0 1, it is subi $1 $0 -1, I’m testing substract at the same time):

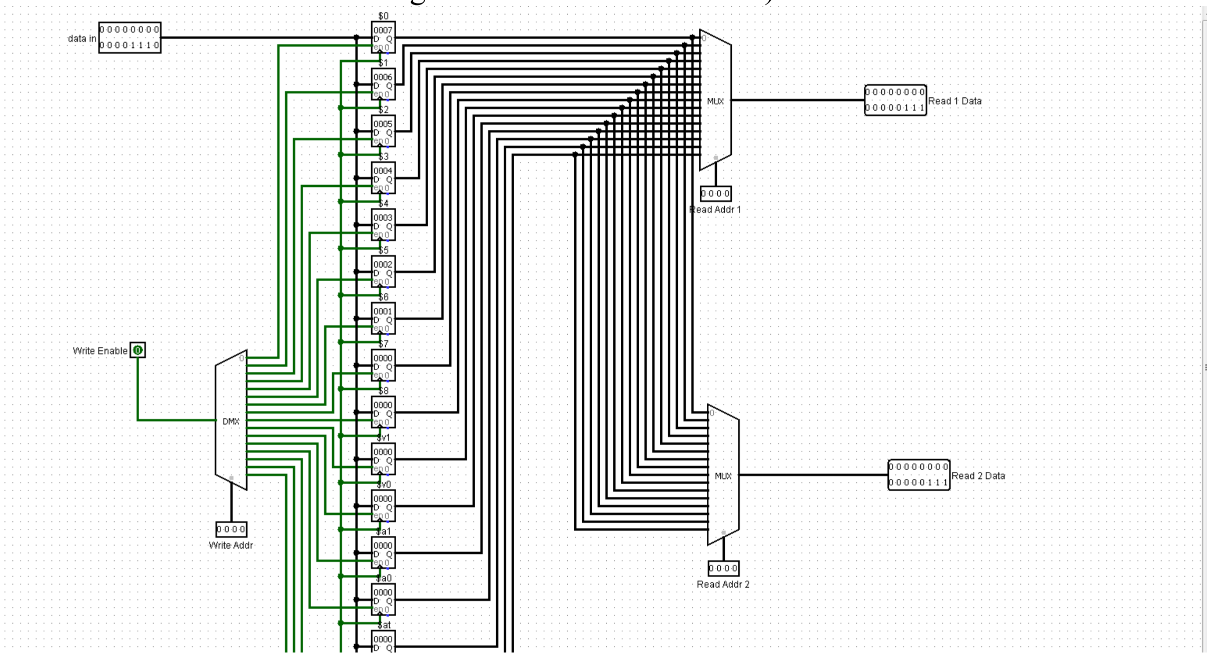
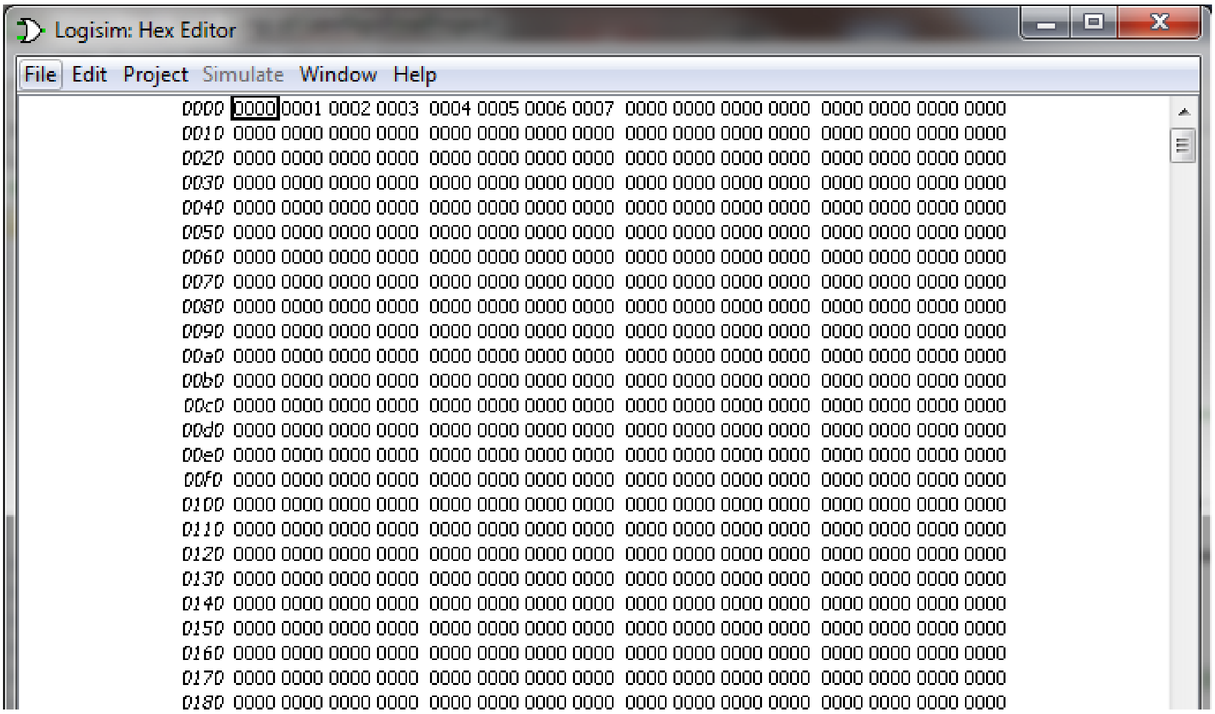


Figure 5: Values from 7 to 0 are in registers from 0 to 7.



```
0000 0000 0001 0002 0003 0004 0005 0006 0007 0000 0000 0000 0000 0000 0000 0000 0000
0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0030 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0040 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0050 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0070 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00a0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00b0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00c0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00d0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00e0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00f0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0120 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0130 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0140 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0150 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0160 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0170 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0180 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 6: Values from 0 to 7 are stored in memory location from 0 to 7.

b. Lam Ngo Test 2: Same as first two test, but reverse the order of the array in memory, values 7..0 are stored from memory location 0 to 7(Adapted from Lab 4):

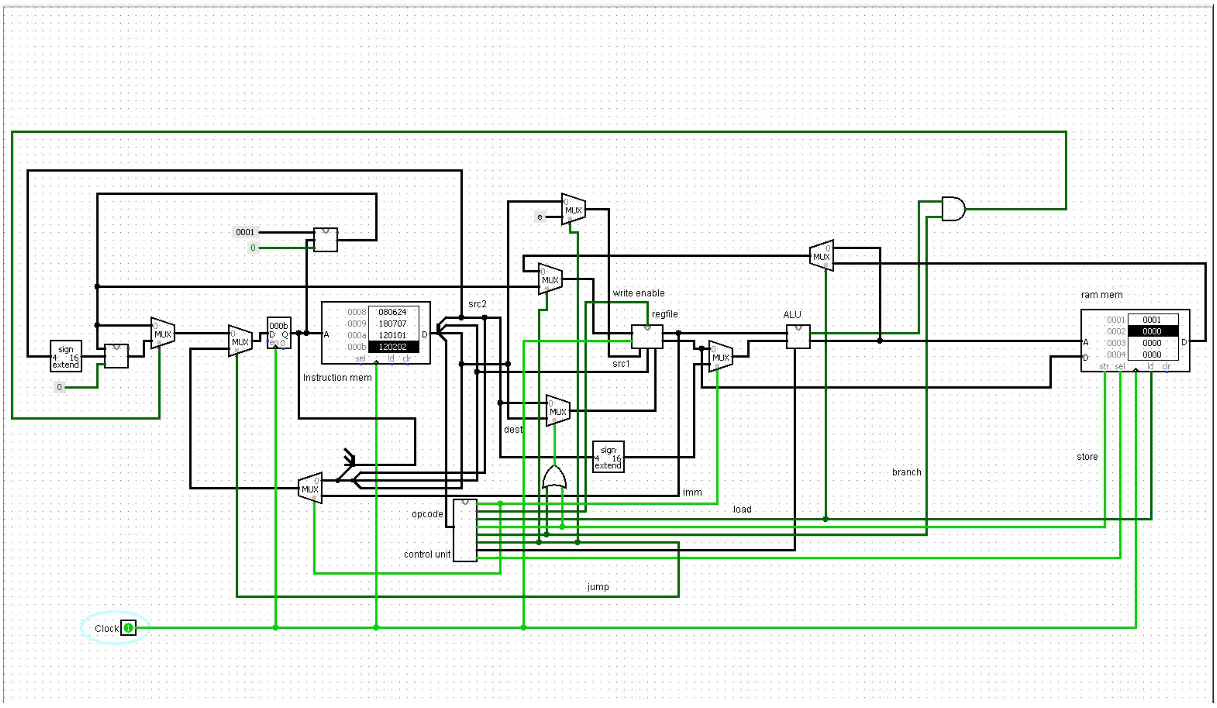
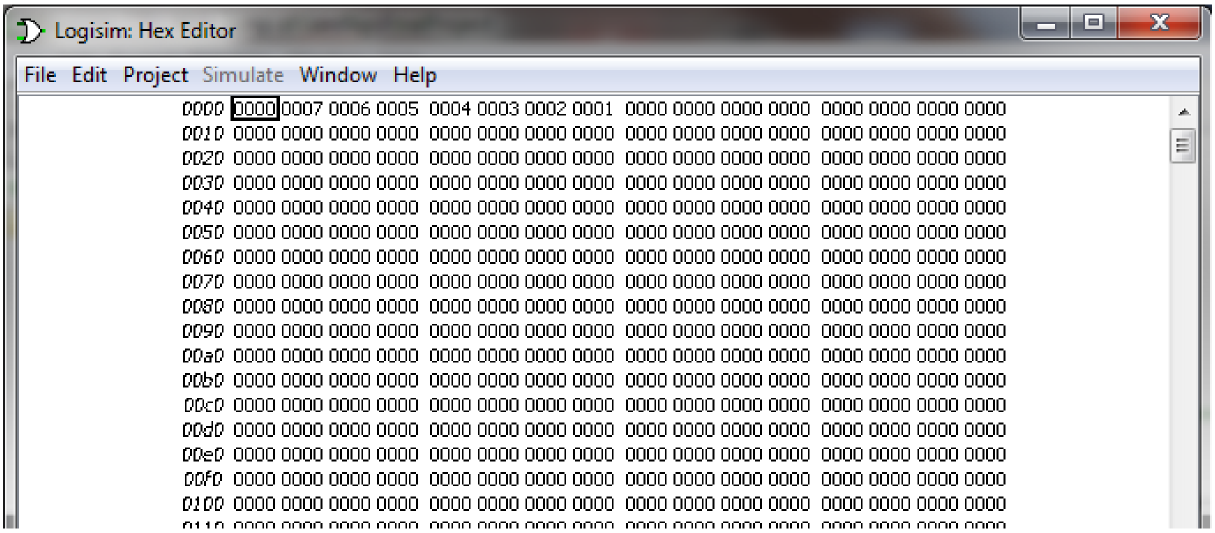


Figure 7: Load Word Control Signal



```
0000 0000 0007 0006 0005 0004 0003 0002 0001 0000 0000 0000 0000 0000 0000 0000 0000
0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0030 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0040 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0050 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0070 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00a0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00b0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00c0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00d0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00e0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00f0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 8: Values 7…0 are reversed and stored from 1 to 7.

III. Test Branch If Equal:

1. Professor Rieffel Test:
   a. Test Positive branch offset: results: $4 should contain 4 and $5 should contain 5.

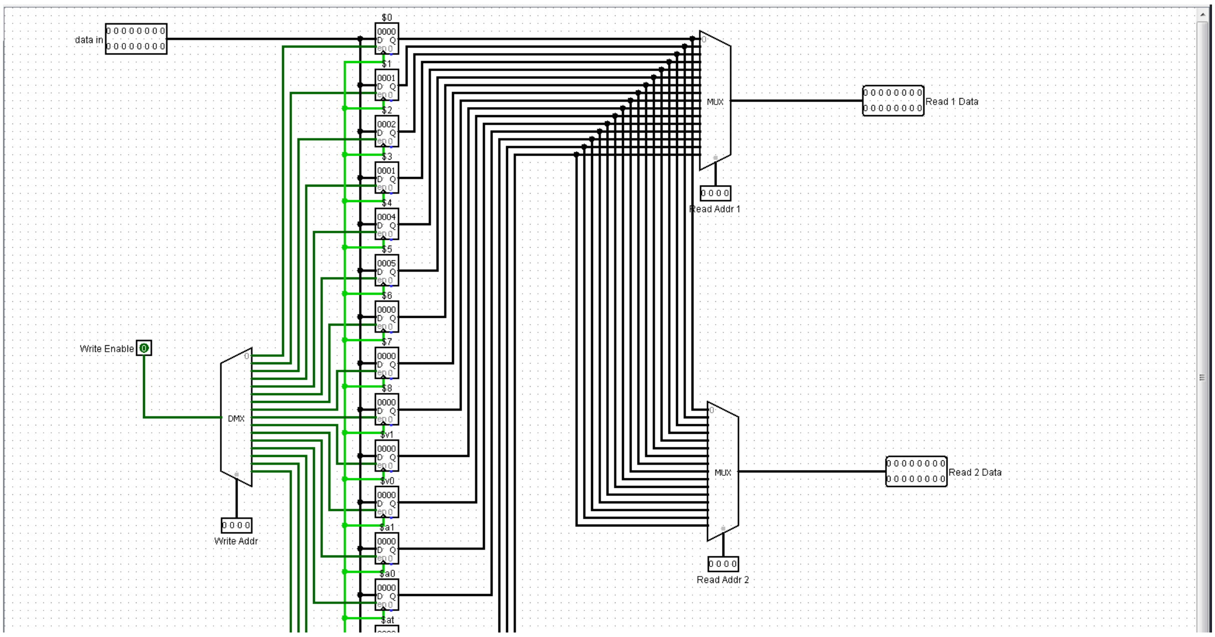


Figure 9: Register 4 contains value 4 and Register 5 contains value 5.

   b. Test Back Branch: registers from 1 to 5 should all contain 5.

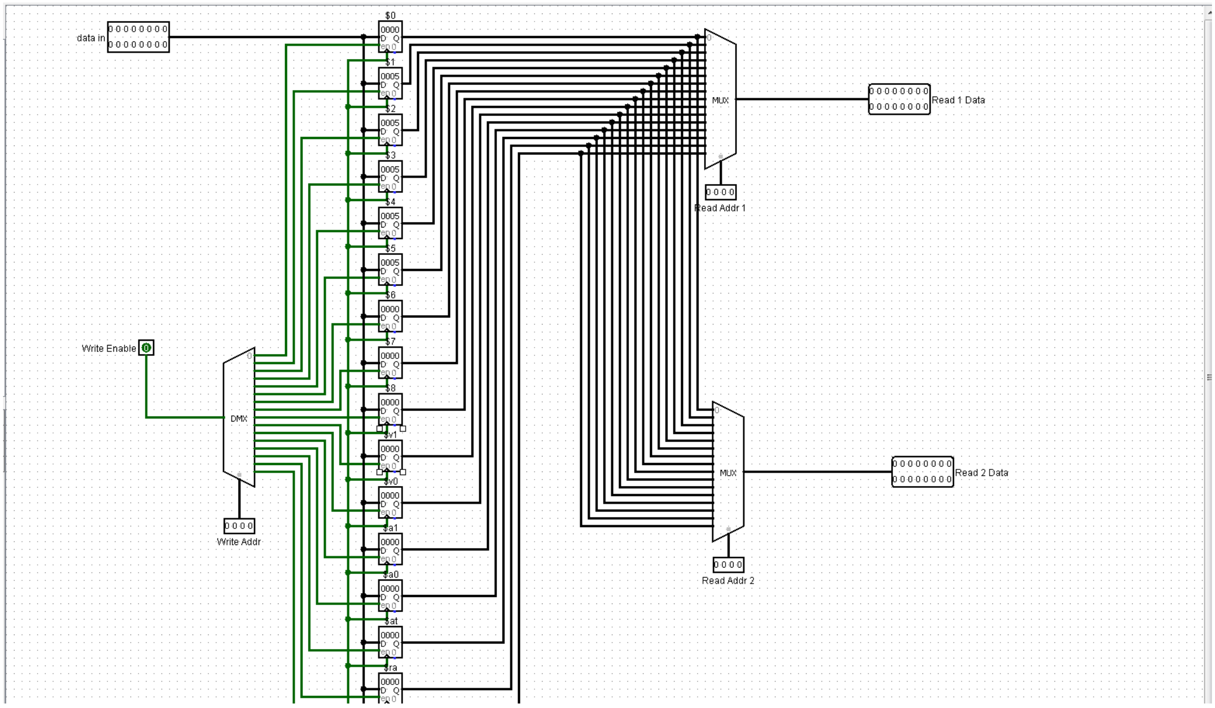


Figure 10: Registers from 1 to 5 all contains 5.

2. Lam Ngo’s Test: True if registers 0-4 are 0, and first branch not taken and second branch

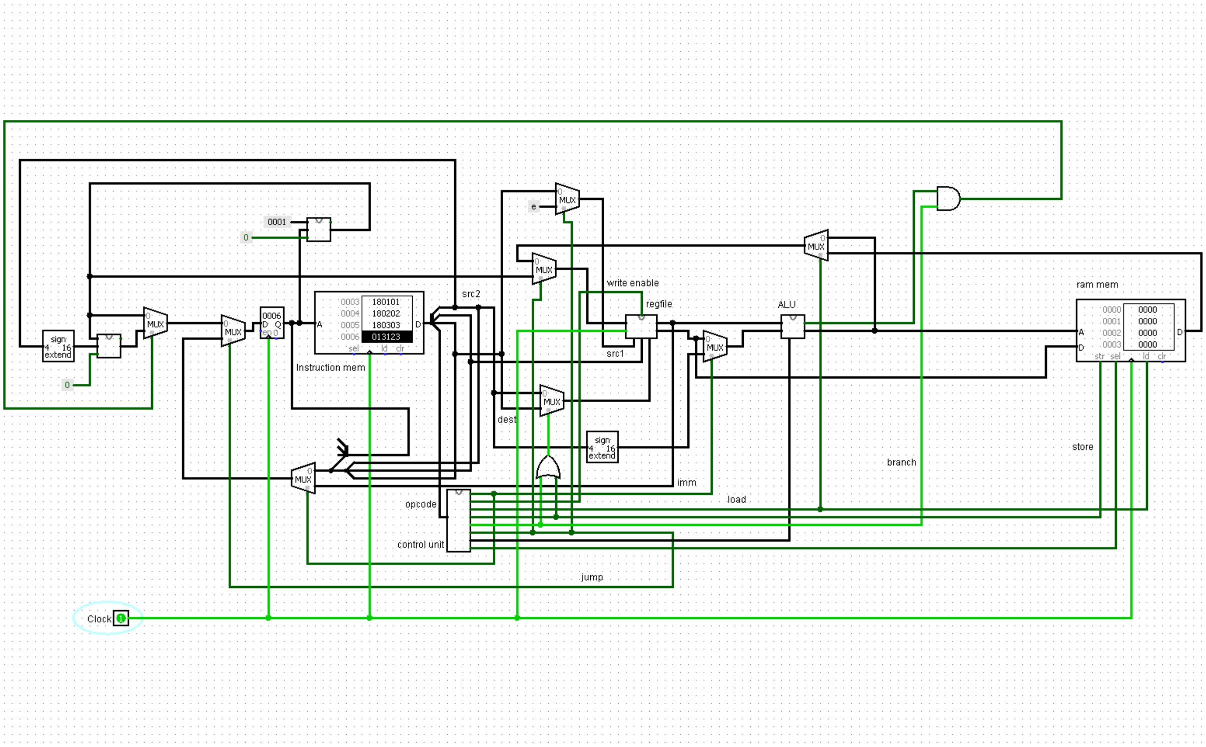


Figure 11: Branch Not Taken Control Signal

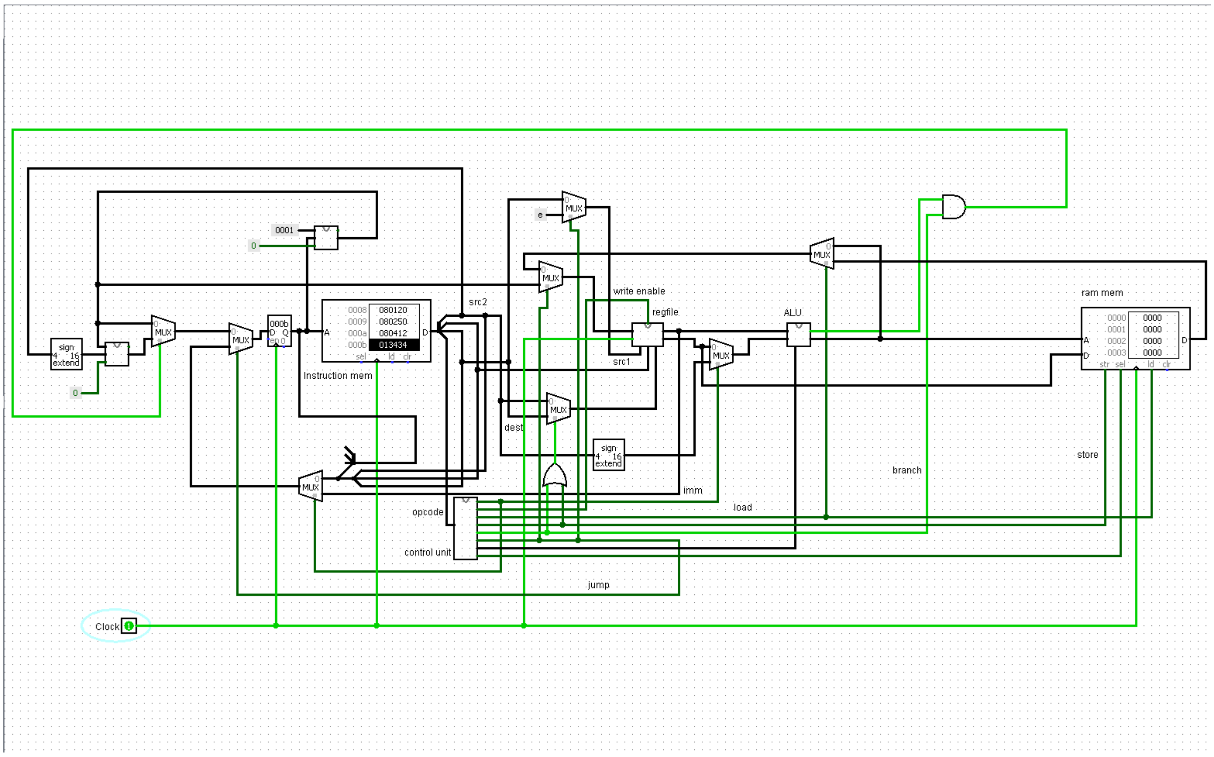


Figure 12: Branch Taken Control Signal.

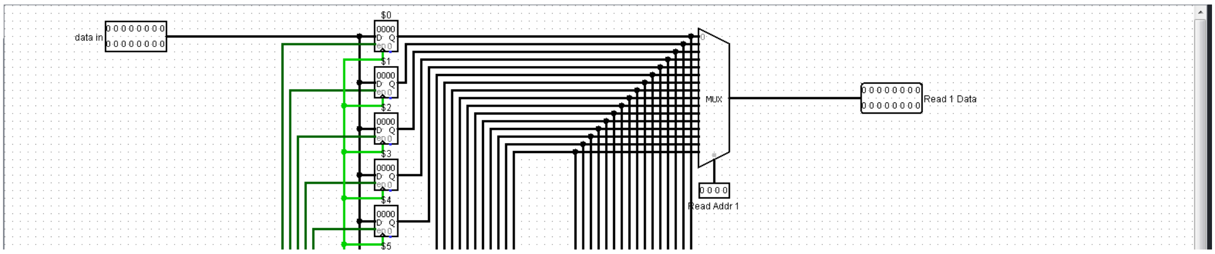


Figure 13: Registers from 0 to 4 all contain 0.

IV. Test Jump Instructions (Jal and Jr will be tested in the next section):

1. Professor Rieffel's test: $1 should have the value 3 if this passes test.

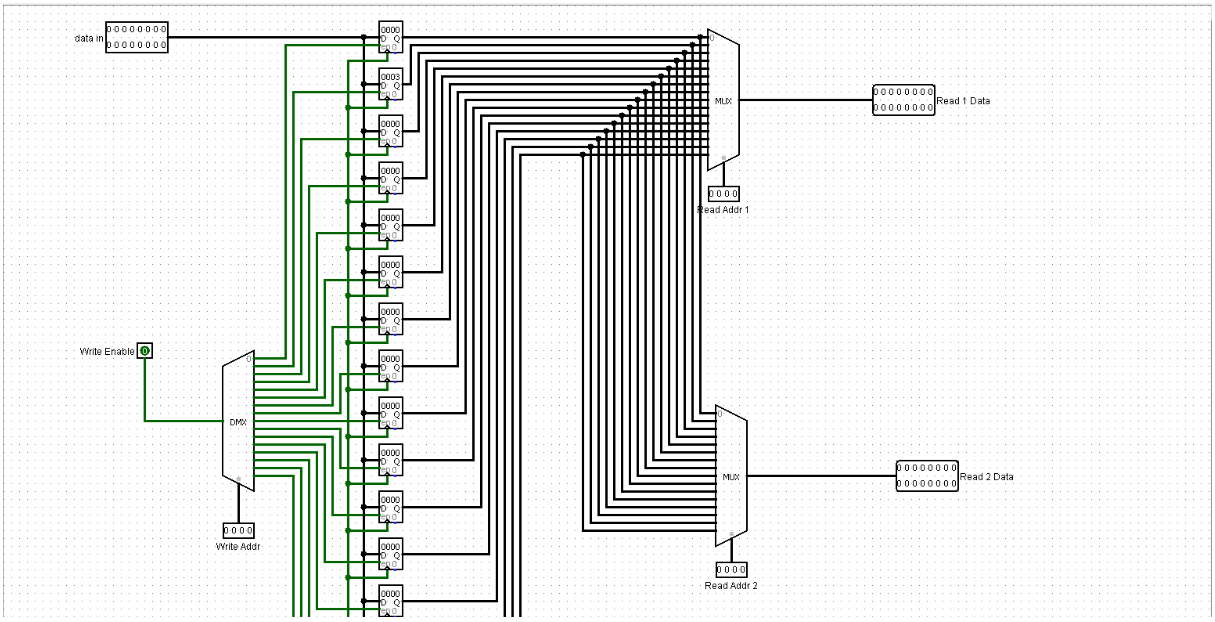


Figure 14: Register 1 has the value 3 after finishing the test.

2. Lam Ngo's test: $1 should have the value 4 if this passes test.

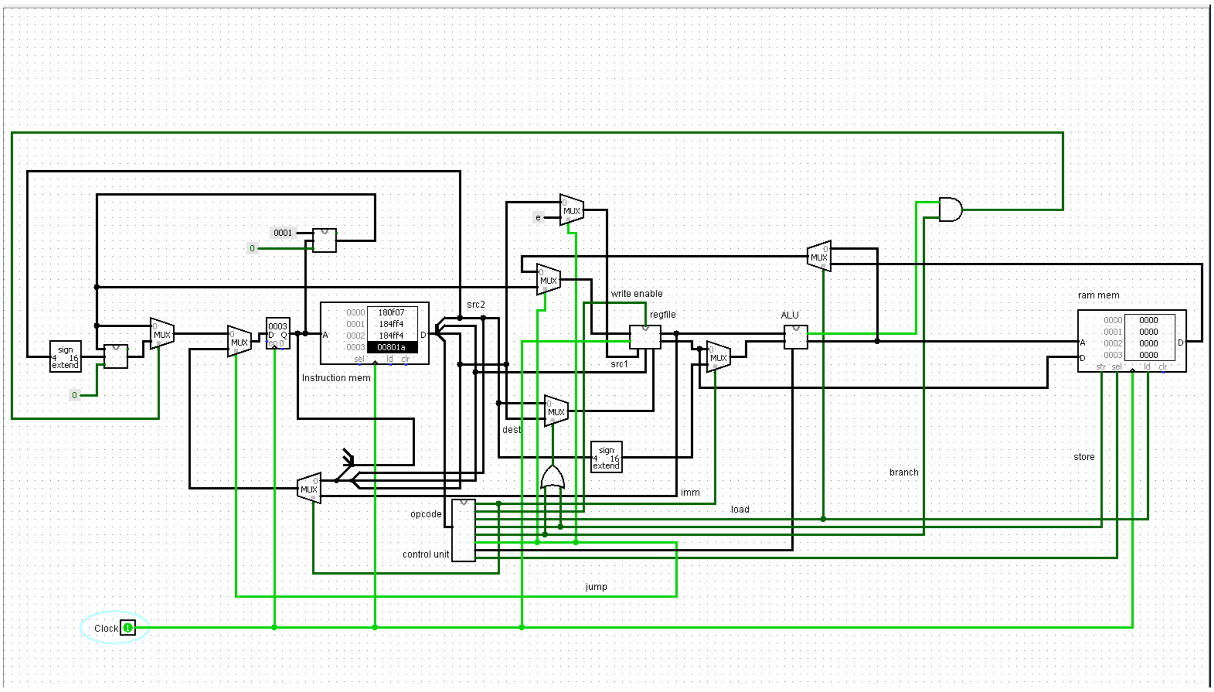


Figure 15: Jump Instruction control signal.

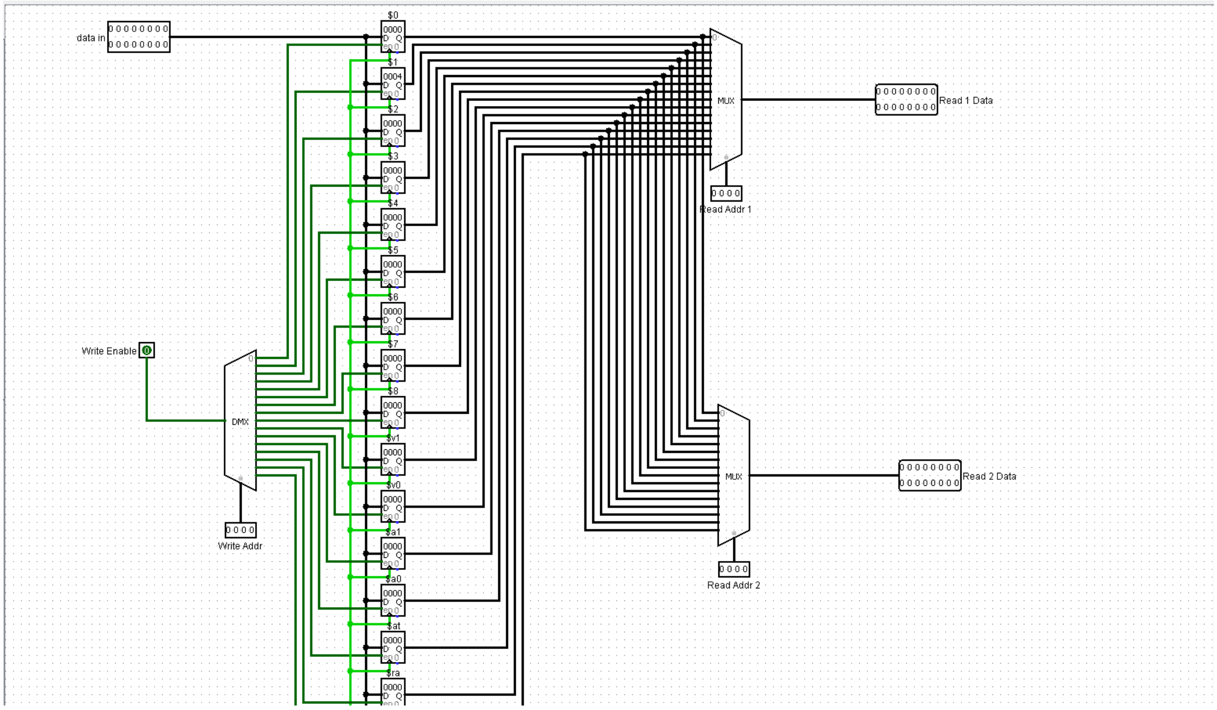


Figure 16: $1 contains the value 4.

V. Test Function calls and Jump And Link and Jump Return:

1. First test: Call 1 function min, return 1 if the first argument is smaller than the second argument. 1 should be stored at address 1.

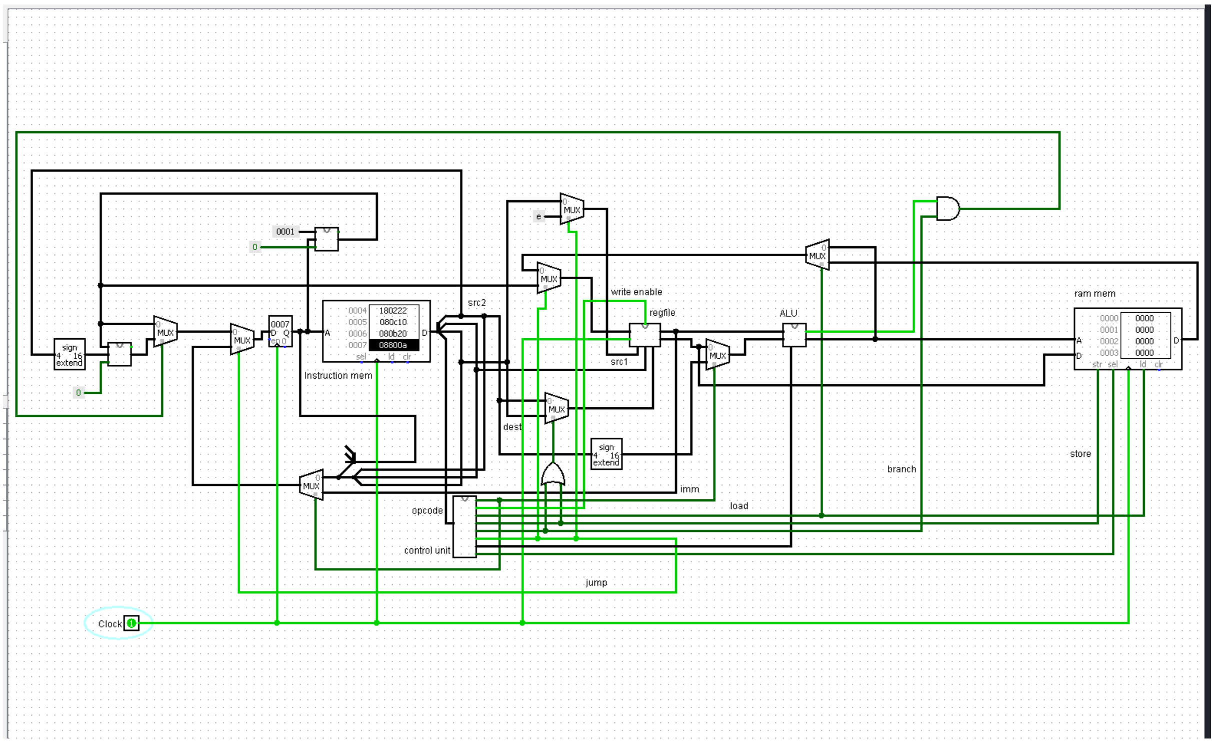


Figure 17: Jump And Link Control Signal (should return to line 8).

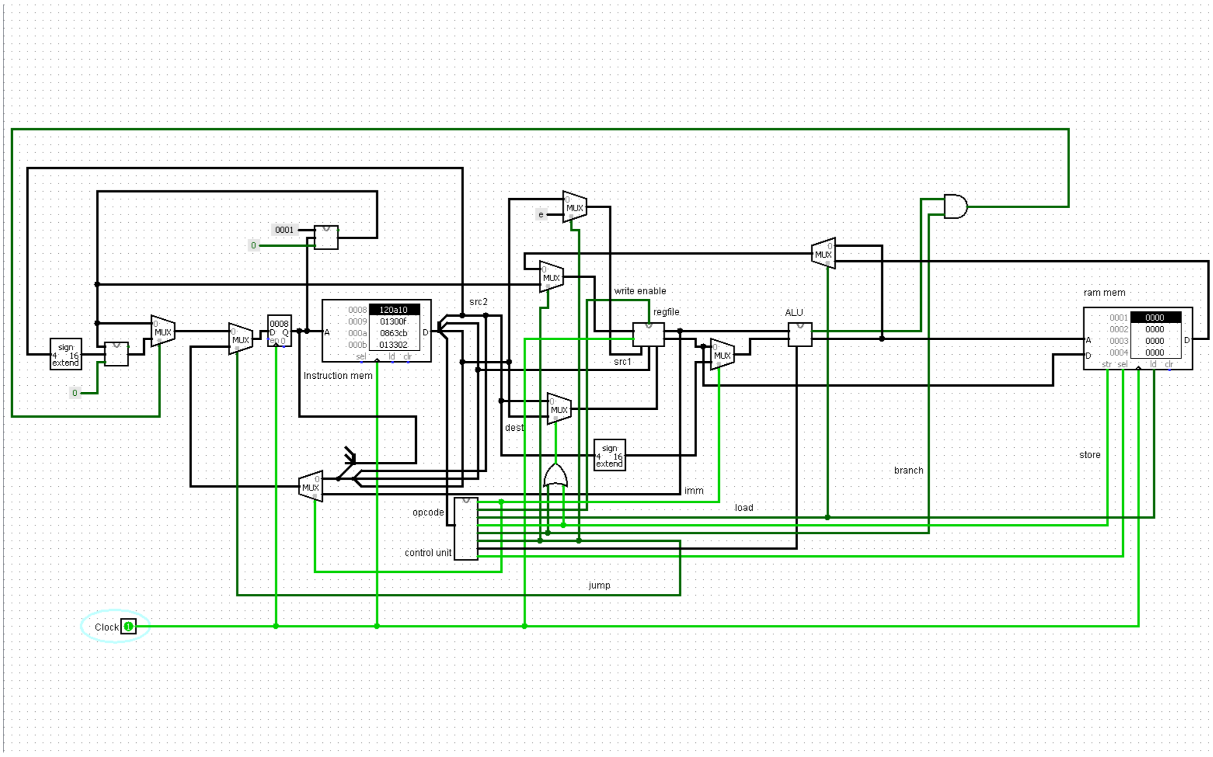
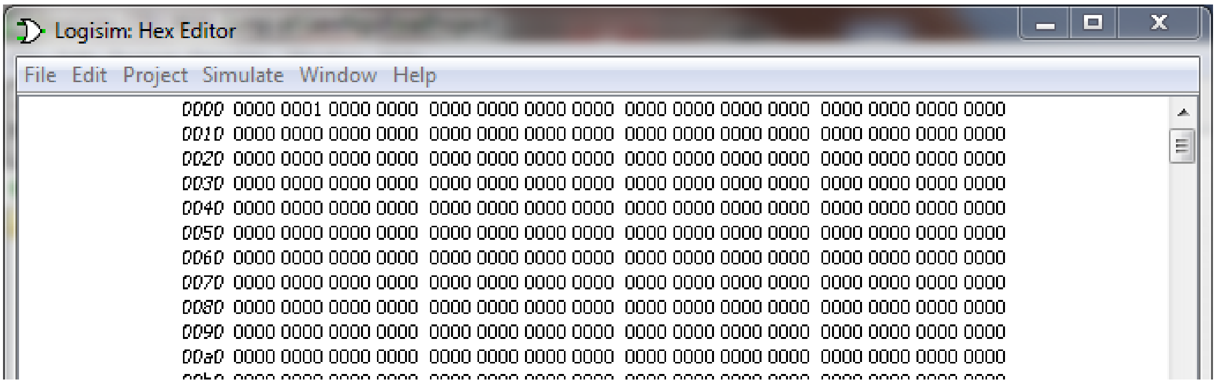


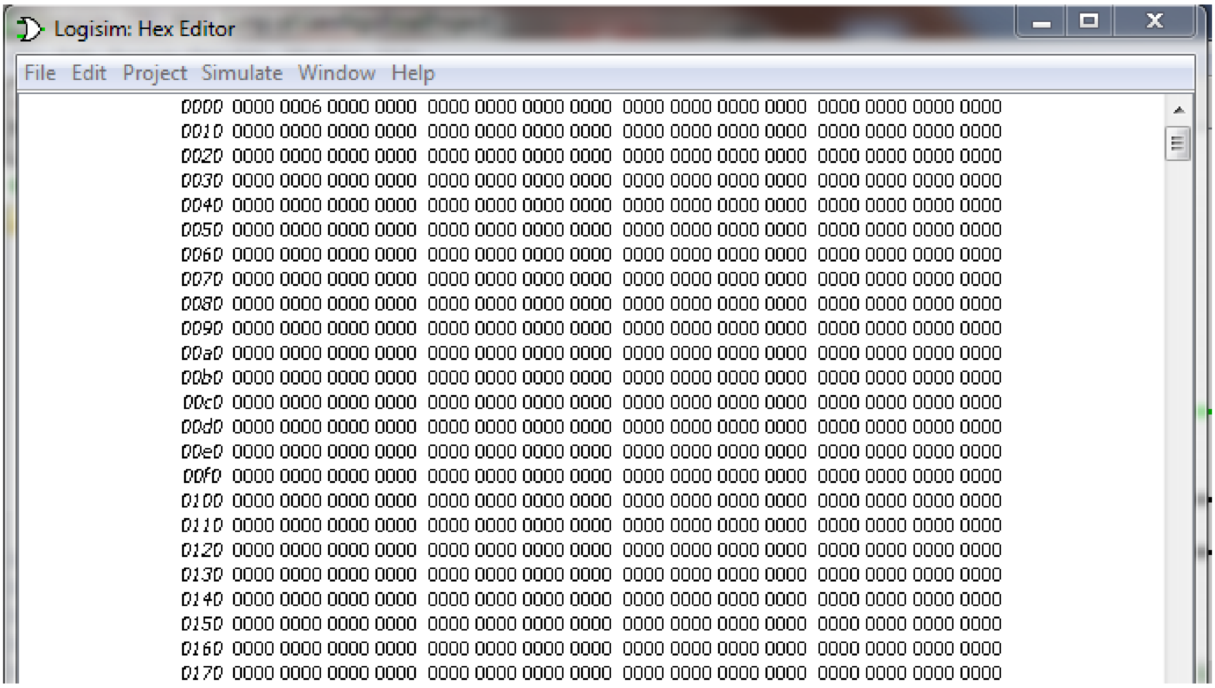
Figure 18: After doing jr $ra, PC now has returned to line 8.



```
0000 0000 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0030 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0040 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0050 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0070 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00a0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 19: Value 1 is stored at address 1.

2. Second Test: Calls the function Add to add two arguments, then calls function Double to double the result. Should store the value 6 at memory location 1.



```
0000 0000 0006 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0030 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0040 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0050 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0070 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00a0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00b0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00c0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00d0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00e0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00f0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0120 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0130 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0140 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0150 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0160 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0170 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 20: After calling the two functions, value 6 is stored at address 1.

VI. Test Function calls with Stacks (nested function calls and recursion):

1. Professor Riffel’s test: Recursion: this program should stores values 0…5 into registers 0…5 then revursively add them together, storing the result in address 1.

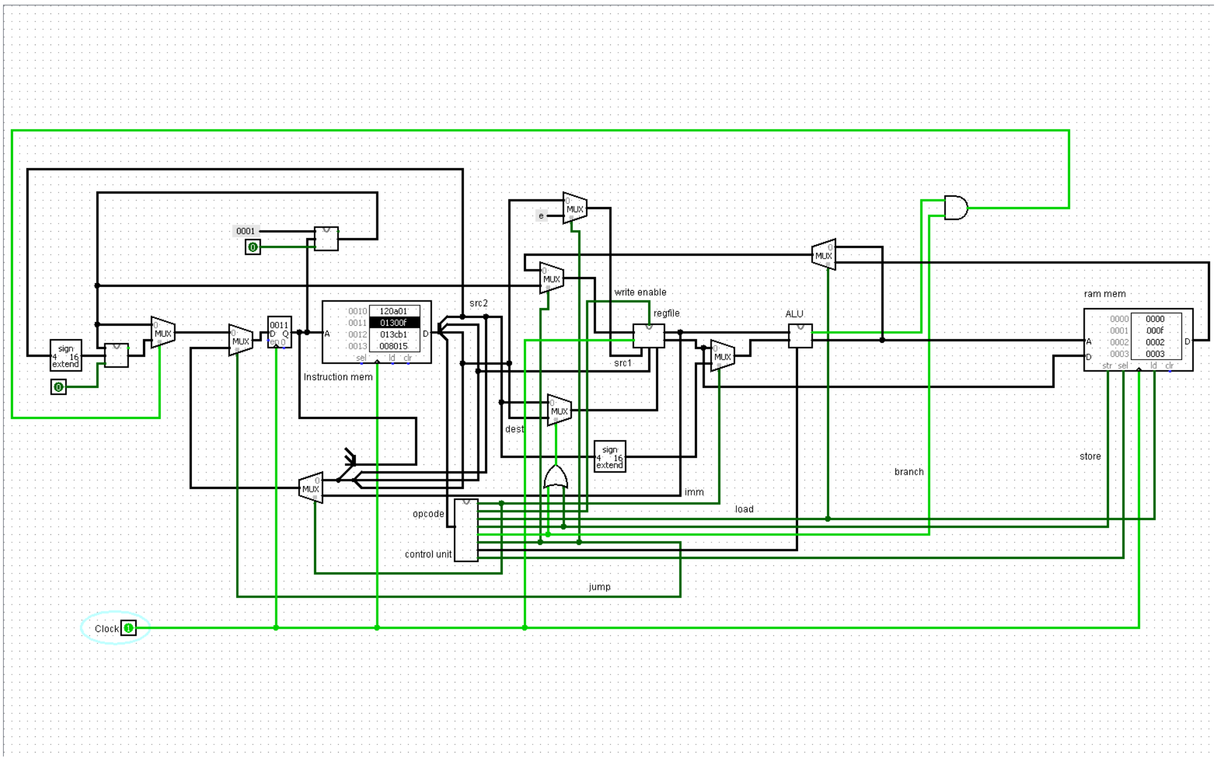


Figure 21: The CPU in an infinite loop at the end of the program (was stated in the program). Can see the correct results loaded in address 1.

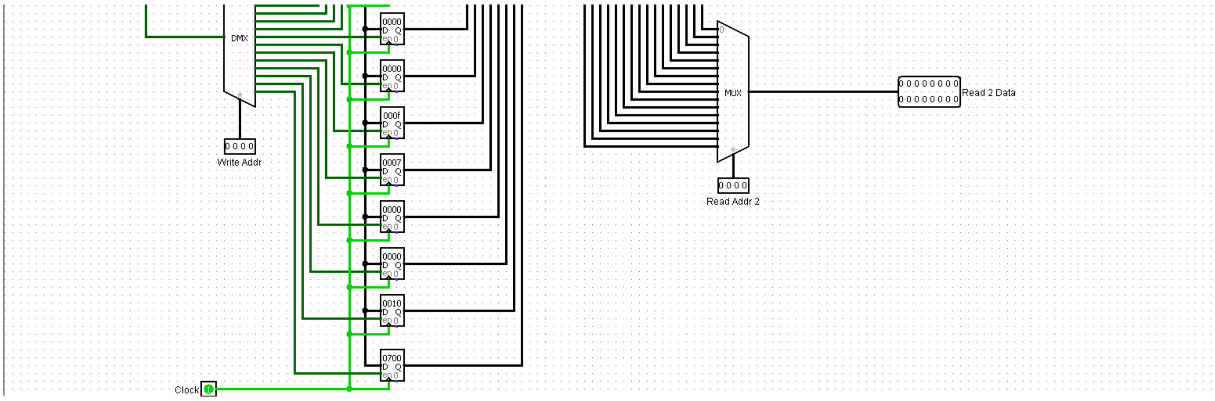
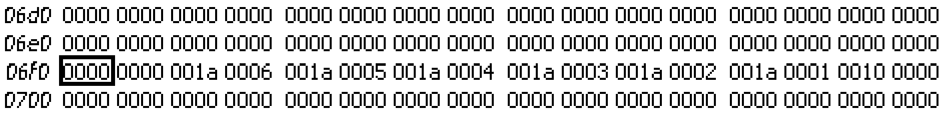


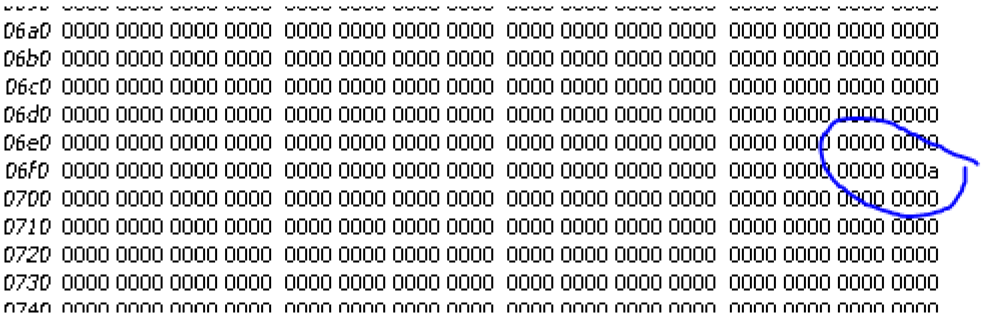
Figure 22: Stack Pointer was initialized at the begining of the program and reset at the end of the program.



```
06d0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06e0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06f0 0000 0000 001a 0006  001a 0005 001a 0004  001a 0003 001a 0002  001a 0001 0010 0000
0700 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
```

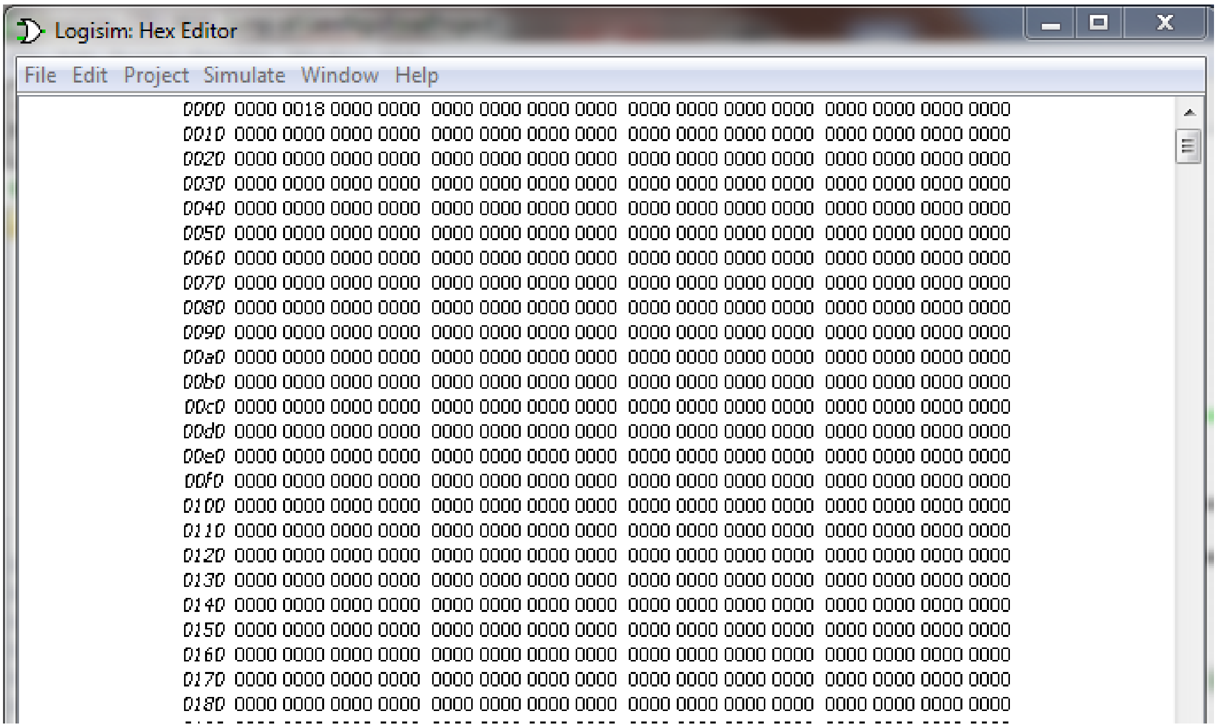
Figure 23: How the stack was used during the program.

2. Lam Ngo’s test: Calling a function Add, and then calls a function Double (which calls another function Triple). The program should store the value 18 at address 1 at the end.



```
06a0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06b0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06c0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06d0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06e0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
06f0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 000a
0700 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0710 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0720 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0730 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
```

Figure 24: The stack used to save the return address.



```
0000 0000 0018 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0010 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0020 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0030 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0040 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0050 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0060 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0070 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0080 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0090 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00a0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00b0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00c0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00d0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00e0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
00f0 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0100 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0110 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0120 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0130 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0140 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0150 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0160 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0170 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
0180 0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000  0000 0000 0000 0000
```

Figure 25: The result 18 at address 1.

VII. Test Pseudoinstructions:

Test BLT and BGT: BLT should be taken and BGT not taken. $4 and $5 should contain 4 and 5 at the end of the program.

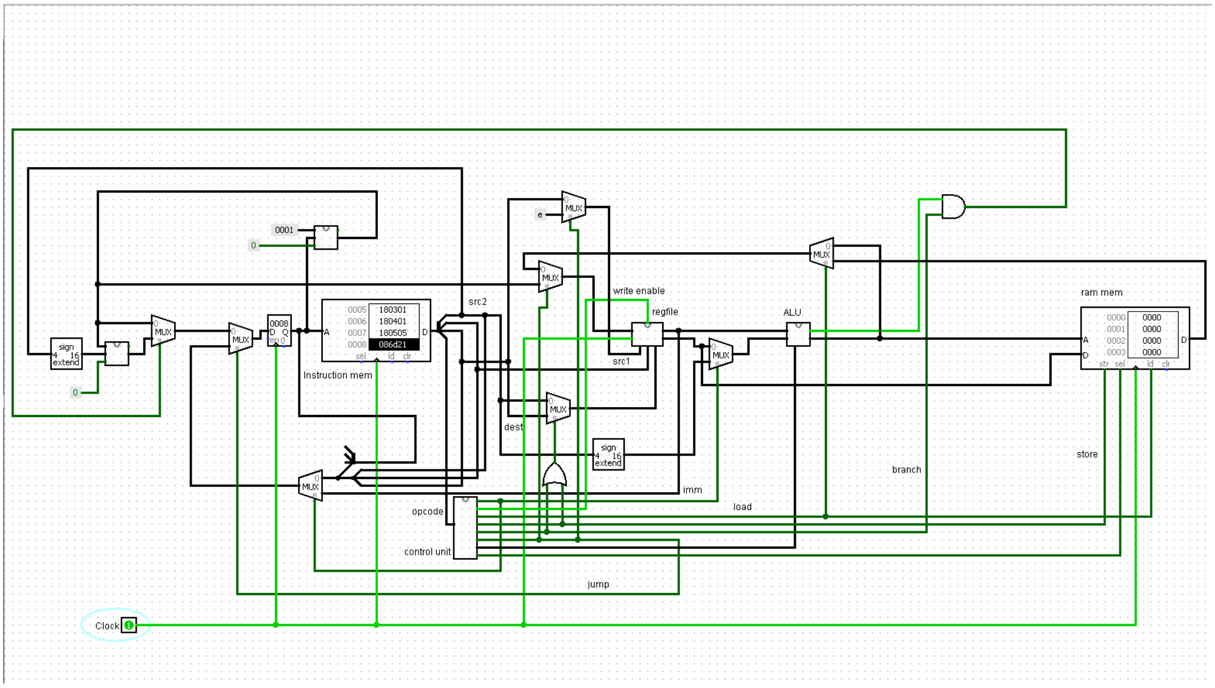


Figure 26: BGT not taken control signal.

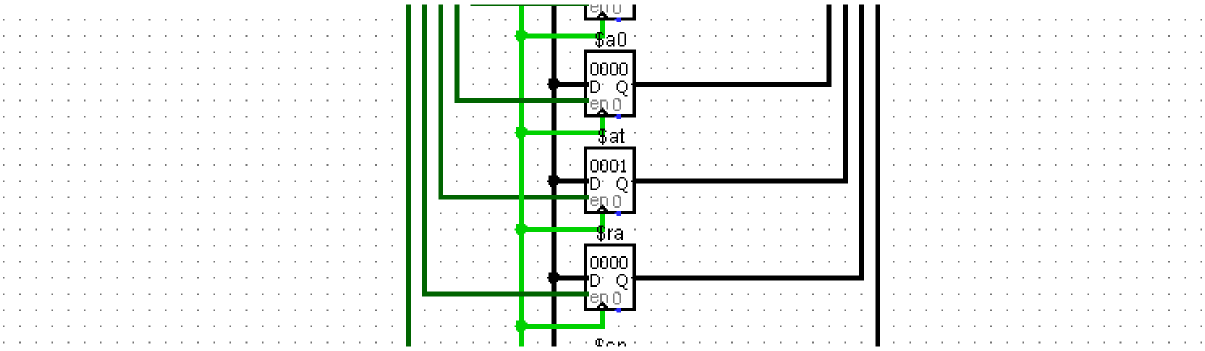


Figure 27: $at changed to 1 for Blt, so BLT should be taken.

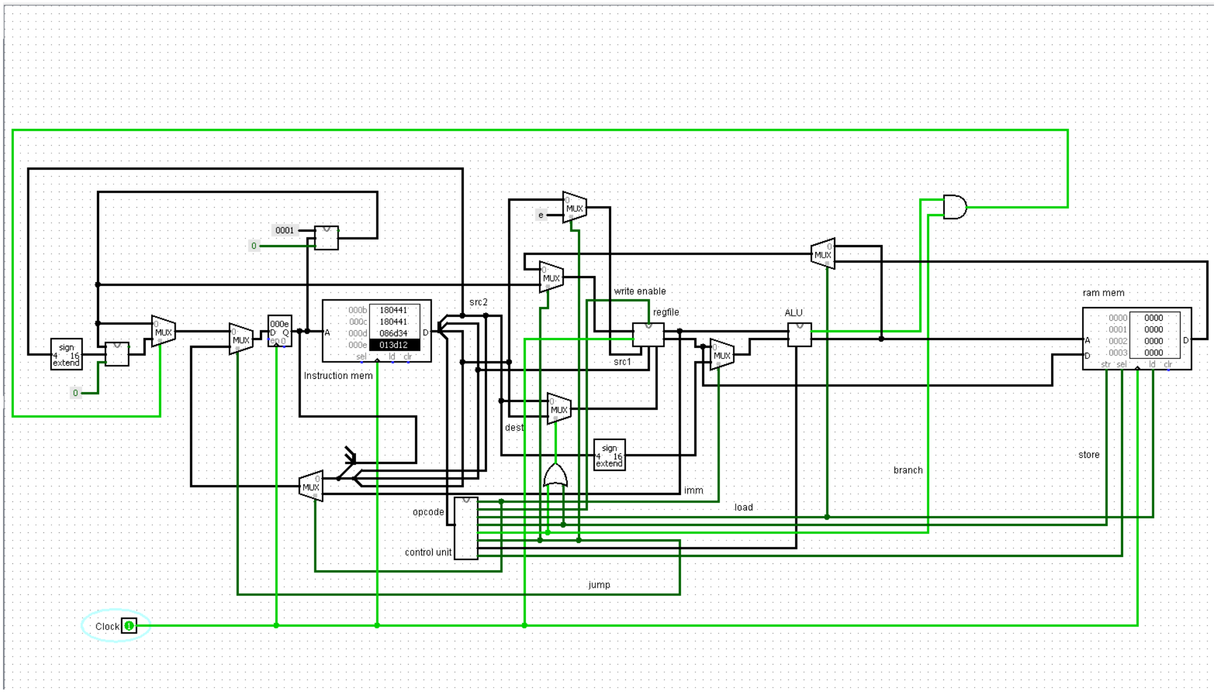


Figure 28: Branch is taken after $at is changed to 1 for BLT.

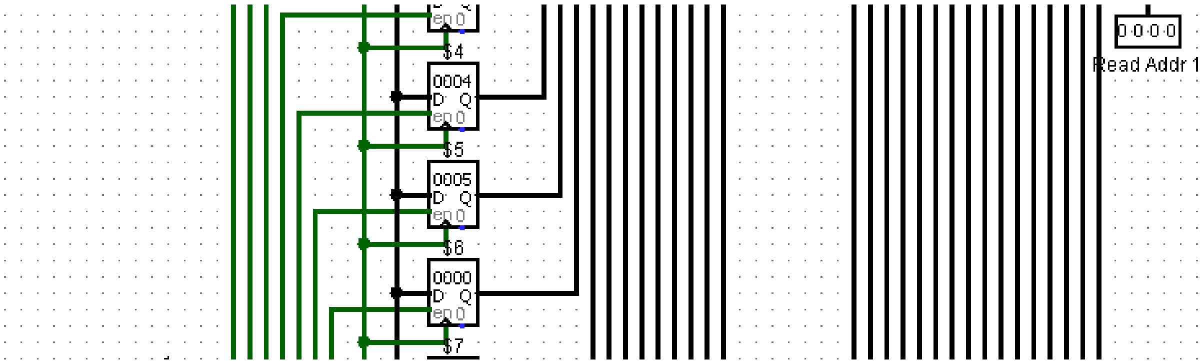


Figure 29: $4 and $5 contain 4 and 5.