CENG3420 Lab3 Report

Ian Ha Jin Quan (1155138078)

Lab 3-1

For state 1, the 8th bit should be 1 so that the next state would go back to itself until READY becomes 1. LD.PC and LD.MAR should be disabled and LD.MDR should be enabled so that MDR will load the new value. Lastly MIO_EN should be enabled so that the memory can be read.

For state 32, j6-j0 should be 0100001 so that it will transition to state 33 next. LD.MDR should be enabled to load the new value. GateRS2 should be enabled to write the value of rs2 to bus. MDRMUX should be 1 to allow MDR to read from the bus.

For state 33, j6-j0 should be 0100001 so that it will loop to itself until READY becomes 1. MIO_EN and WE should be enabled to allow writing to memory. DATASIZE is 1 for selecting a bit width.

For state 51, j6-j0 should be 0000000 so that it will transition back to state 0 next. LD.REG should be enabled to allow reading from register. RS1En and RS2En should be enabled to allow the ALU to read the values for both registers. GateALUSHF should be enabled to allow the result of ALU to be written to the bus.

For state 102, j6-j0 should be 0000000 so that it will transition back to state 0 next. LD.PC should be enabled to allow PC to load value from PCMUX. PCMUX should be 1 so that it will PC will be loaded with the value of rd that is in the BUS. ADDR1MUX and ADDR2MUX should be 10 and 01 respectively so that rs1 and imm value will be selected and added together. RS1En should be enabled to allow rs1 to be read by ADDR1MUX. Finally, GateMAR should be open to write the value rd to the BUS.

In riscv-lc.c, CURRENT_LATCHES.REGS[0] = 0 is added to hardwire the register x0 to 0.

Lab3-2

For reading and writing memory, I followed the hints in the slides for writing and reading 8, 16 and 32 bit memory. I used switch-case to select the appropriate bit width based on the datasize_mux.

For LD.REG, the register specified in IR 11:7 should be loaded with the value from BUS.

For LD_MAR, the MAR in next latches should be loaded with value from the BUS.

For LD_IR, the IR in next latches should be loaded with value from the BUS.

For LD_PC, the PC in next latches should be loaded with the value from PCMUX, which will select the value from BUS or the value of PC+4.

Lab 3-3

For the value of GateMAR, it will get its value form the MARMUX, which will select between IR[31.12] or value_of_MARMUX.

For the value of shift function unit, it will need the value of funct3 and fucnt7, which can be get from IR[14:12] and IR[31:25] respectively. Then it will need the input from rs1 and RS2MUX.

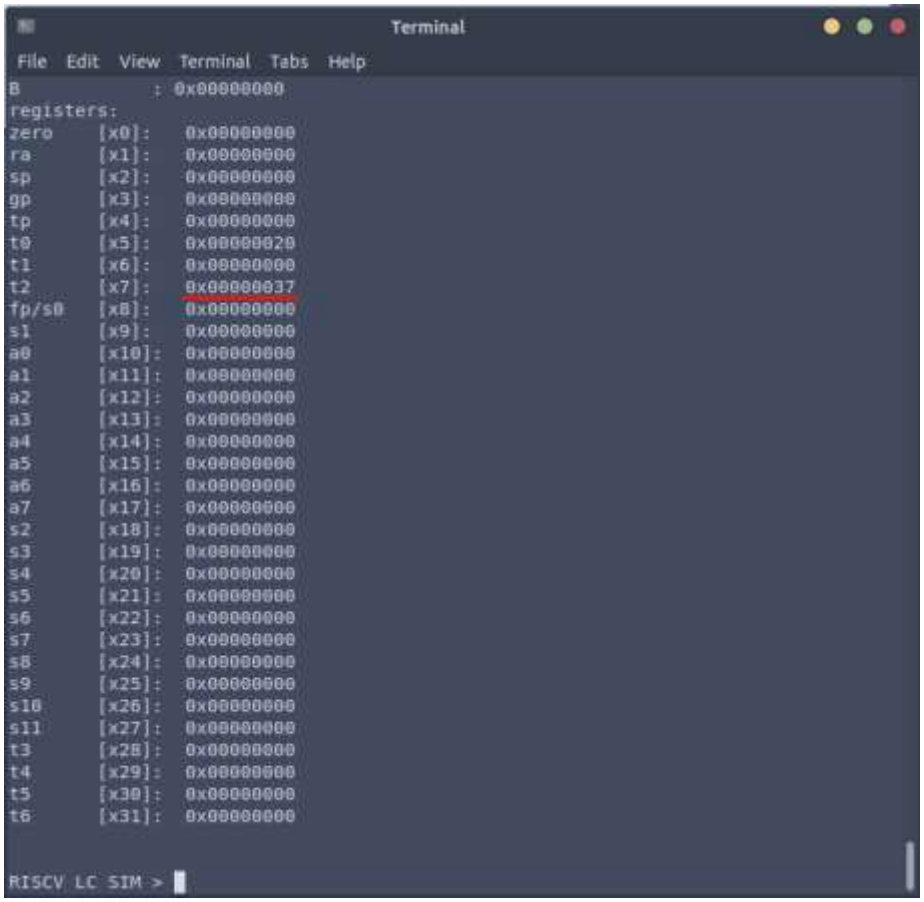
For GateALUSHF, it will choose between the output value of ALU or Shift based on the value of funct3, which can be get from IR[14:12].

For GateRS2, it will get its value directly from rs2.

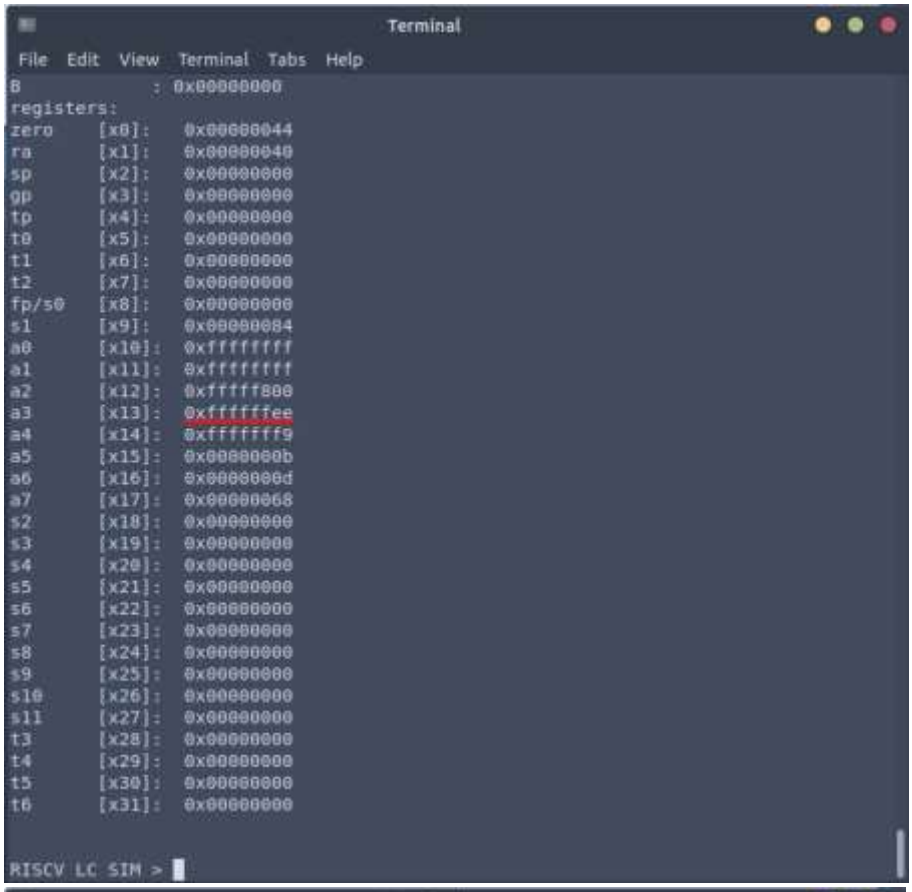
For function drive_bus(), in the case-switch statement, case 1 will be GateMAR, case 2 will be GateALUSHF, case 4 will be GatePC, case 8 will be GateRS2 and case 16 will be GateMDR.

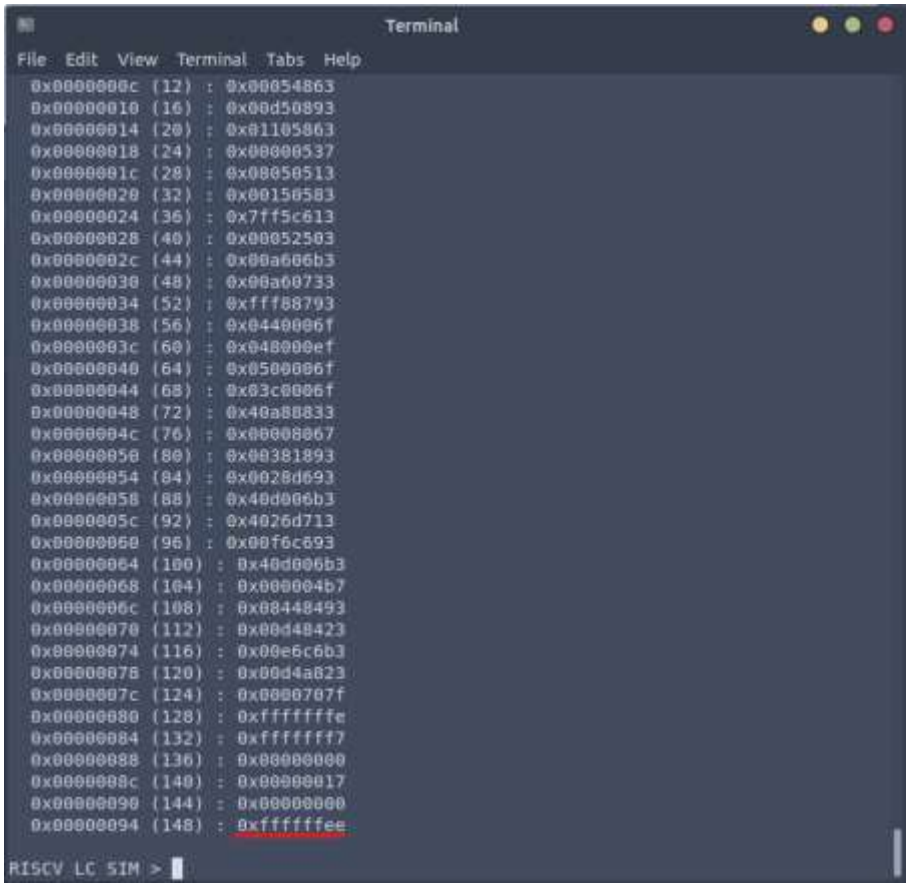
Console result:

Count10



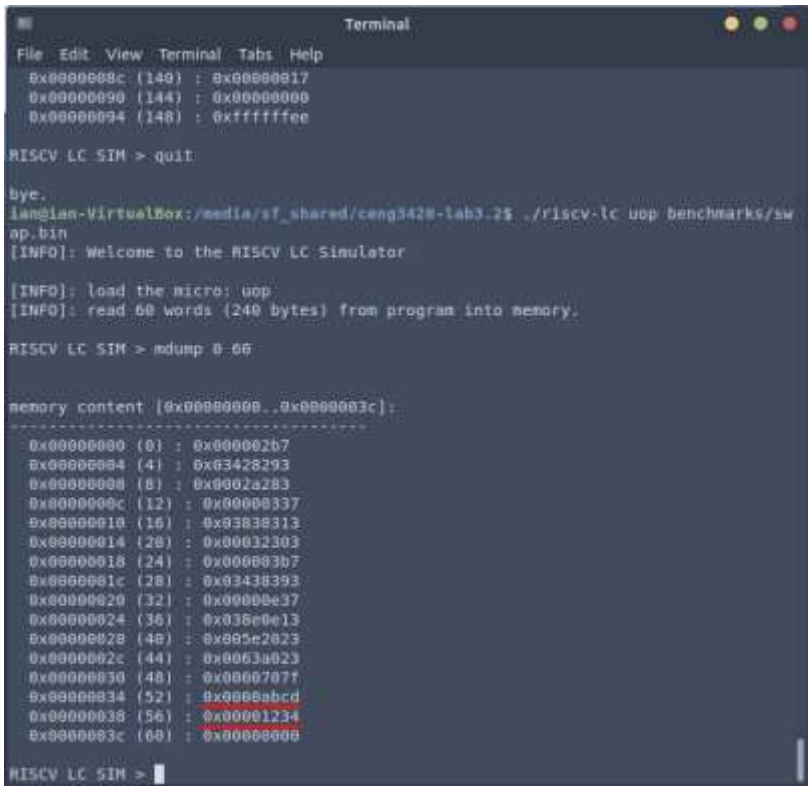
isa


Terminal
File Edit View Terminal Tabs Help
B : 0x00000000
registers:
zero [x0]: 0x00000044
ra [x1]: 0x00000040
sp [x2]: 0x00000000
gp [x3]: 0x00000000
tp [x4]: 0x00000000
t0 [x5]: 0x00000000
t1 [x6]: 0x00000000
t2 [x7]: 0x00000000
fp/s0 [x8]: 0x00000000
s1 [x9]: 0x00000084
a0 [x10]: 0xffffffff
a1 [x11]: 0xffffffff
a2 [x12]: 0xfffff800
a3 [x13]: 0xffffffee
a4 [x14]: 0xfffffff9
a5 [x15]: 0x0000000b
a6 [x16]: 0x0000000d
a7 [x17]: 0x00000068
s2 [x18]: 0x00000000
s3 [x19]: 0x00000000
s4 [x20]: 0x00000000
s5 [x21]: 0x00000000
s6 [x22]: 0x00000000
s7 [x23]: 0x00000000
s8 [x24]: 0x00000000
s9 [x25]: 0x00000000
s10 [x26]: 0x00000000
s11 [x27]: 0x00000000
t3 [x28]: 0x00000000
t4 [x29]: 0x00000000
t5 [x30]: 0x00000000
t6 [x31]: 0x00000000
RISCV LC SIM >


Terminal
File Edit View Terminal Tabs Help
0x0000000c (12) : 0x00054863
0x00000010 (16) : 0x00d50893
0x00000014 (20) : 0x01105863
0x00000018 (24) : 0x00000537
0x0000001c (28) : 0x00050513
0x00000020 (32) : 0x00150583
0x00000024 (36) : 0x7ff5c613
0x00000028 (40) : 0x00052503
0x0000002c (44) : 0x00a606b3
0x00000030 (48) : 0x00a00733
0x00000034 (52) : 0xfff88793
0x00000038 (56) : 0x0440006f
0x0000003c (60) : 0x048000ef
0x00000040 (64) : 0x0500006f
0x00000044 (68) : 0x03c0006f
0x00000048 (72) : 0x40a88833
0x0000004c (76) : 0x00008067
0x00000050 (80) : 0x00381893
0x00000054 (84) : 0x0028d693
0x00000058 (88) : 0x40d006b3
0x0000005c (92) : 0x4026d713
0x00000060 (96) : 0x00f6c693
0x00000064 (100) : 0x40d006b3
0x00000068 (104) : 0x000004b7
0x0000006c (108) : 0x00848493
0x00000070 (112) : 0x00d48423
0x00000074 (116) : 0x00e6c6b3
0x00000078 (120) : 0x00d4a823
0x0000007c (124) : 0x0000707f
0x00000080 (128) : 0xfffffffe
0x00000084 (132) : 0xfffffff7
0x00000088 (136) : 0x00000000
0x0000008c (140) : 0x00000017
0x00000090 (144) : 0x00000000
0x00000094 (148) : 0xffffffee
RISCV LC SIM >

Swap

Before:



After: