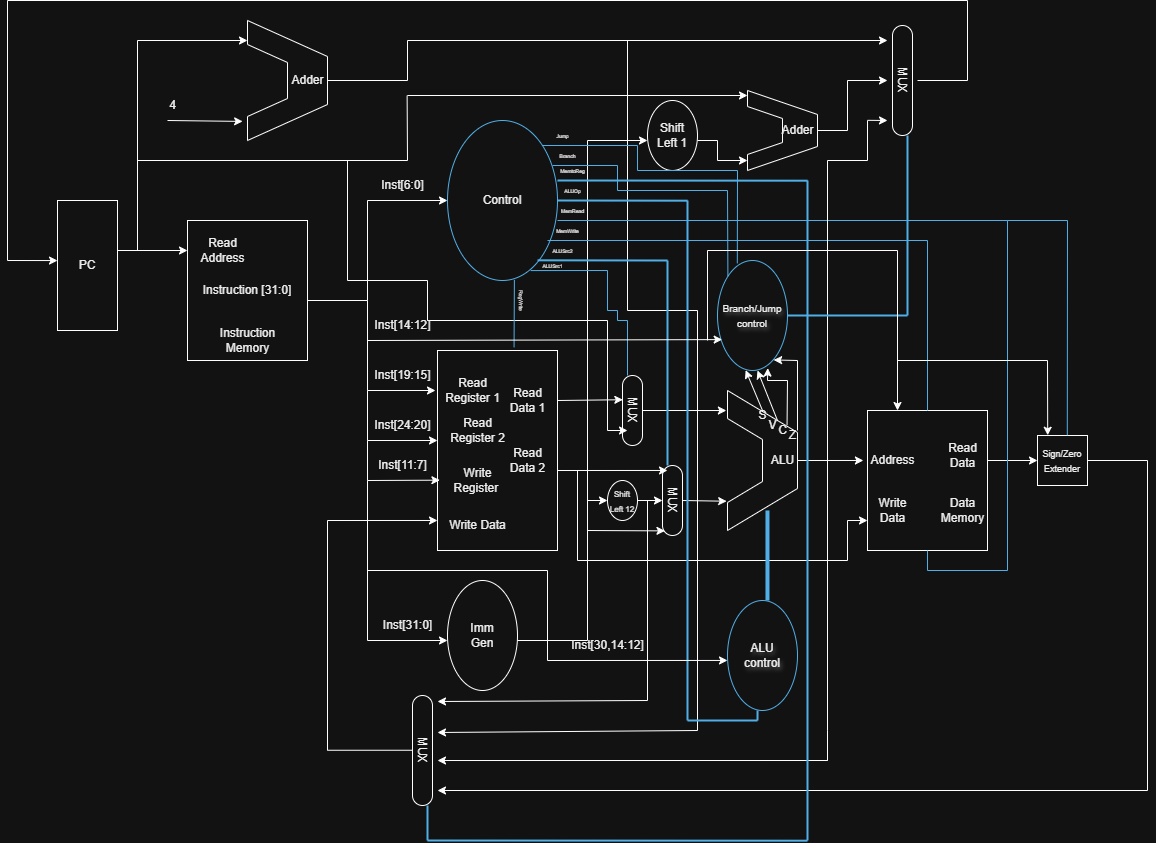
**Single Cycle CPU Report**

**Seif ElAnsary**

**Nadine ElGarem**

**Schematic**

The schematic that I created for the full RISV32I single cycle CPU is:



As you can see, we chose to abstract the branching control in a separate branching/jumping control unit. This simplifies the schematic a bit. An extra AluSrc mux was added near to Read data 1 in order to accommodate AUIPC. The data memory now also handles zero/sign extending and the MemtoReg mux was increased in size in order to accommodate, PC + 4, ALU result, LUI and data memory output. ALU control unit was enhanced to support all the new instructions and the Branch mux was expanded to accommodate JALR.

**Implementation**

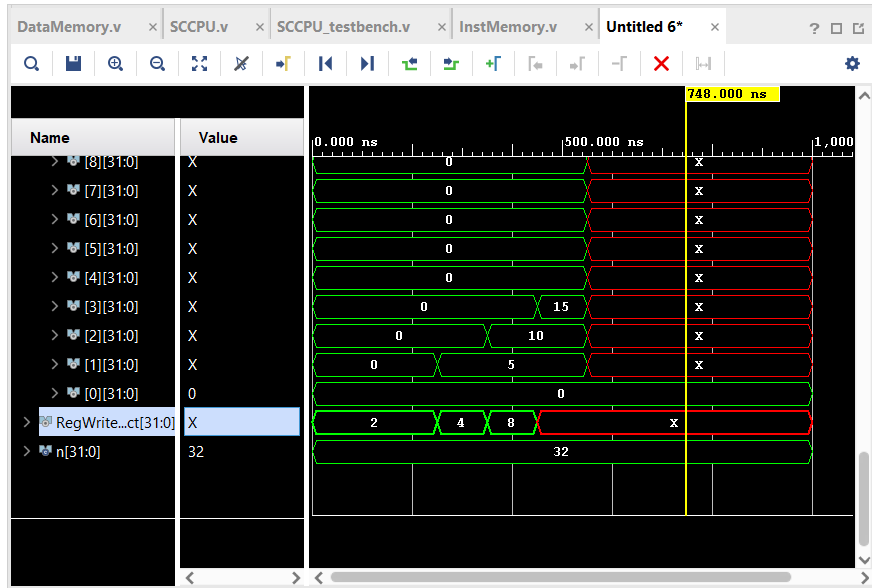
We successfully implanted all 37 instructions successfully as you will see in the testing below and there exists no assumptions about any instructions. Each and every bug was swiftly found and dealt with during the exhaustive training we did.

**Test Waveforms**

1. **Add**0x00500093 #addi x1, x0, 5 x1 = 5

0x00A00113 #addi x2, x0, 10 # x2 = 10

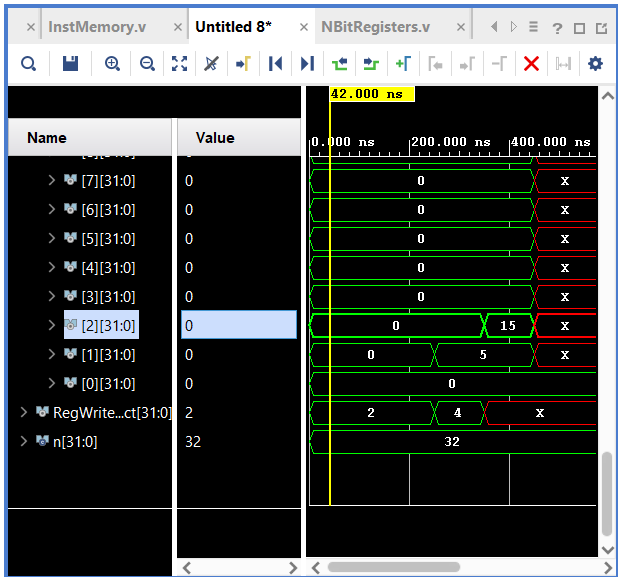
0x002081B3 #add x3, x1, x2 # x3 = 15



1. **Addi**

0x00500093 #addi x1, x0, 5 # x1 = 5

0x00A08113 #addi x2, x1, 10 # x2 = 15

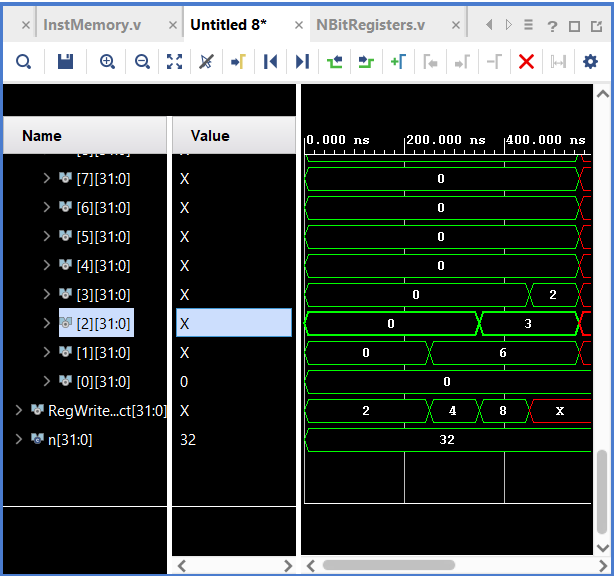


1. **And**

0x00600093 #addi x1, x0, 6 # 0110

0x00300113 #addi x2, x0, 3 # 0011

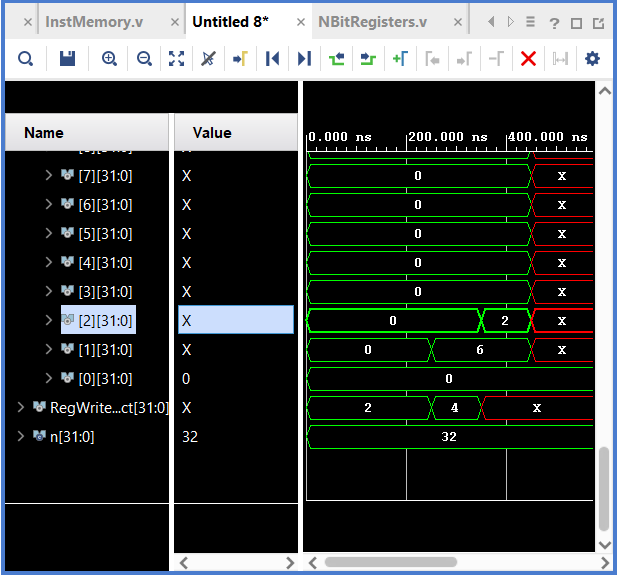
0x0020F1B3 #and x3, x1, x2 # 0010 = 2



1. **Andi**

0x00600093 #addi x1, x0, 6 # 0110

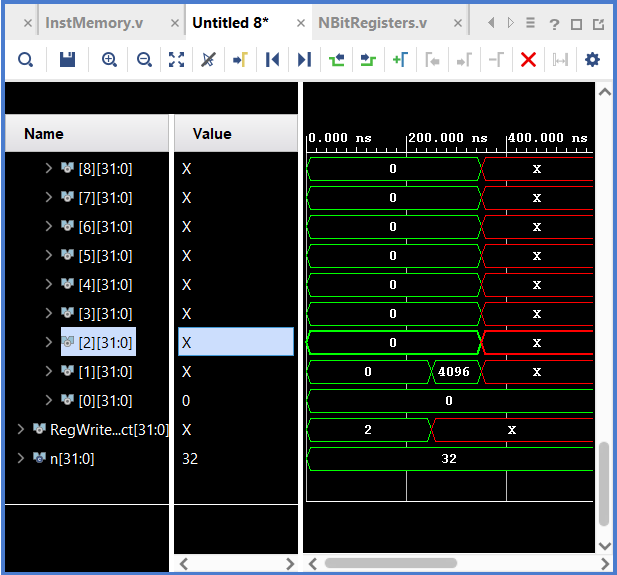
0x0030F113 #andi x2, x1, 3 # x2 = 2



1. **AUIPC**

0x00001097 #auipc x1, 0x00001 # x1 = PC + 0x1000(4096)

00001097 # auipc x1, 0x00001



1. **BEQ**

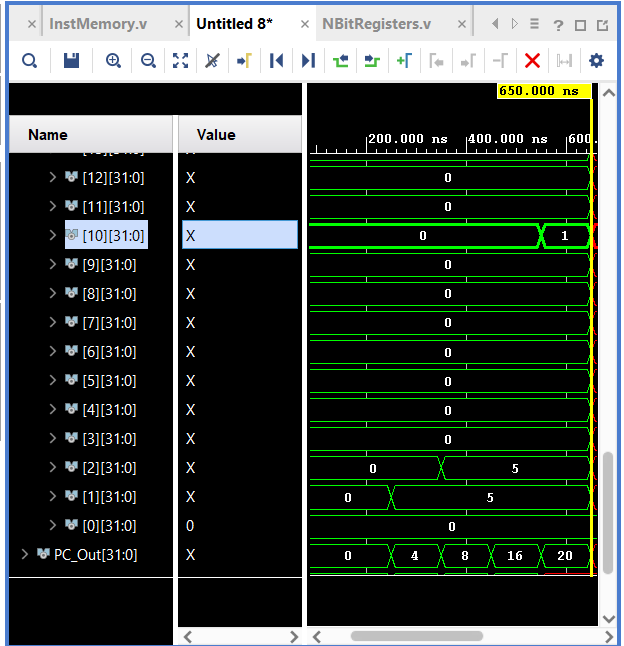
0x00500093 # addi x1, x0, 5

0x00500113 # addi x2, x0, 5

0x00208463 # beq x1, x2, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1



1. **BGE**

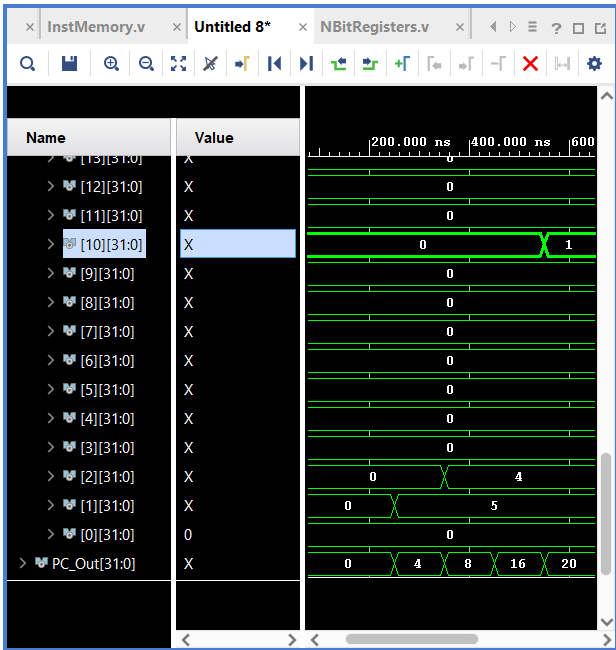
0x00500093 # addi x1, x0, 5

0x00400113 # addi x2, x0, 4

0x0020D463 # bge x1, x2, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1



1. **BGEU**

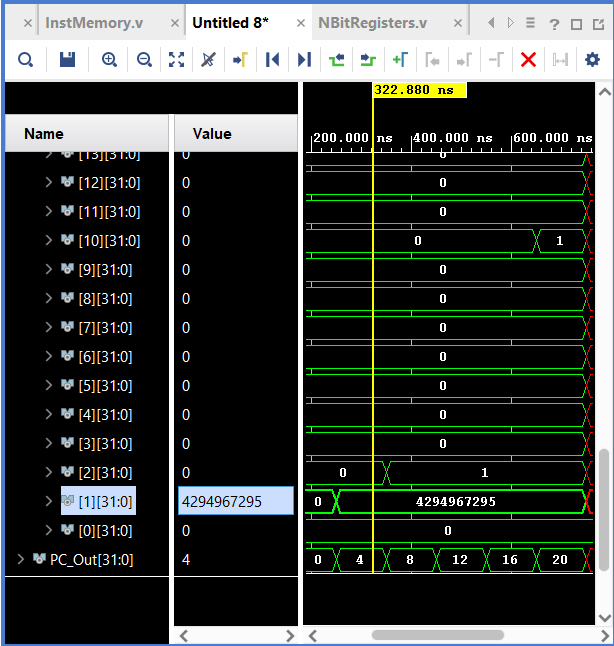
0xFFF00093 # addi x1, x0, -1

0x00100113 # addi x2, x0, 1

0x0020F463 # bgeu x1, x2, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1



1. **BLT**

# test\_blt.s

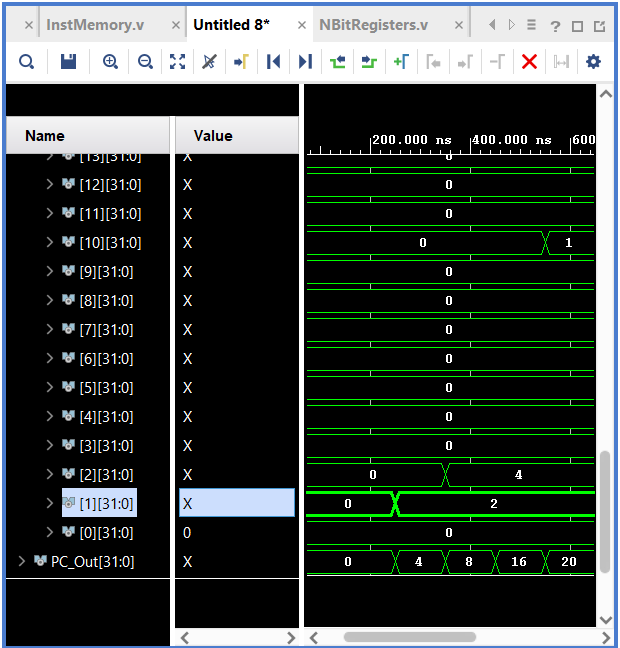
0x00200093 # addi x1, x0, 2

0x00400113 # addi x2, x0, 4

0x0020C463 # blt x1, x2, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1



1. **BLTU**

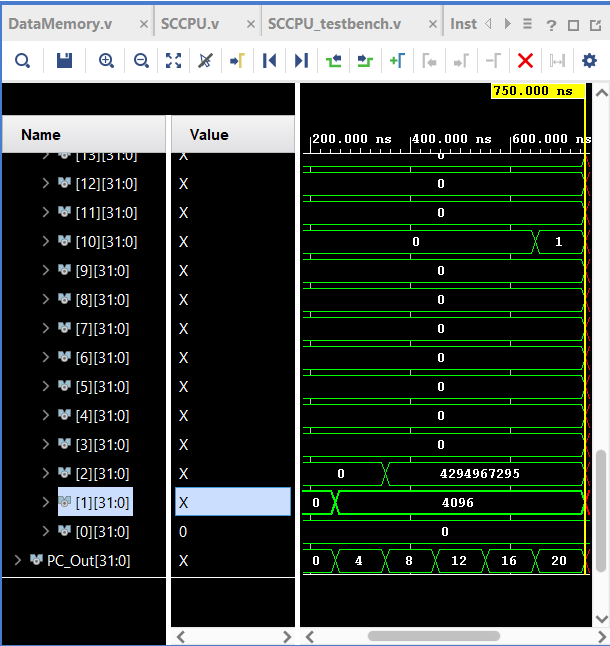
0x000010B7 # lui x1, 0x00001

0xFFF00113 # addi x2, x0, -1

0x0020E463 # bltu x1, x2, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1



1. **BNE**

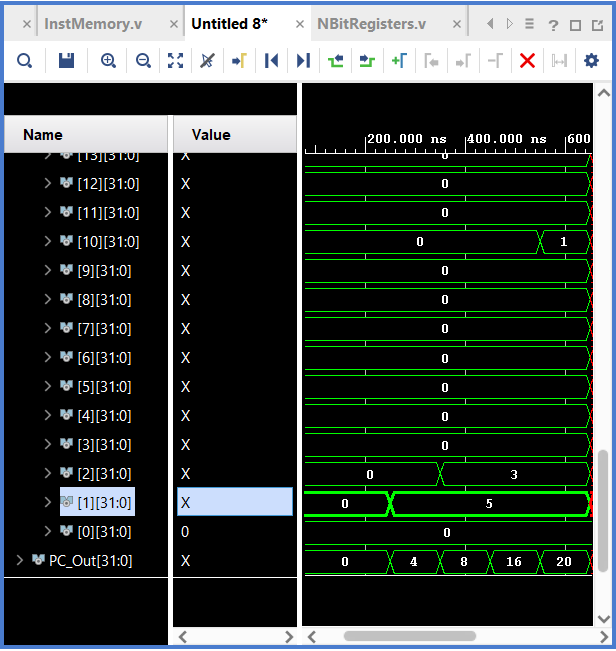
0x00500093 # addi x1, x0, 5

0x00300113 # addi x2, x0, 3

0x00209463 # bne x1, x2, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1

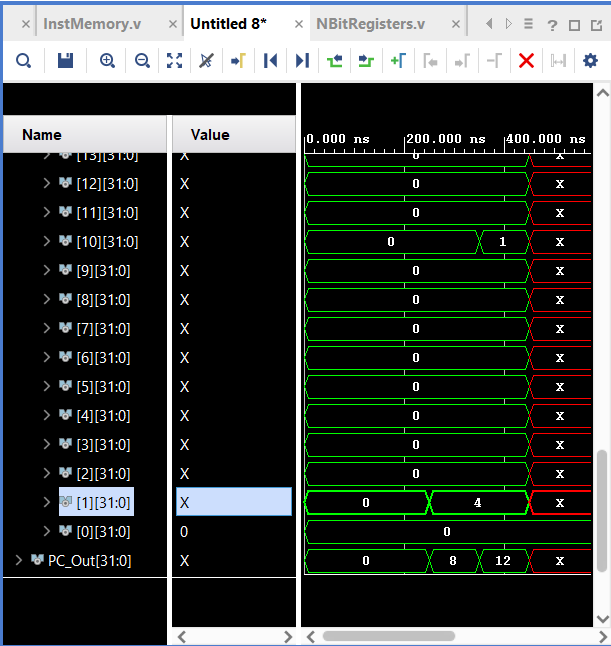


1. **JAL**

0x008000EF # jal x1, 8

0x00000513 # addi x10, x0, 0

0x00100513 # addi x10, x0, 1



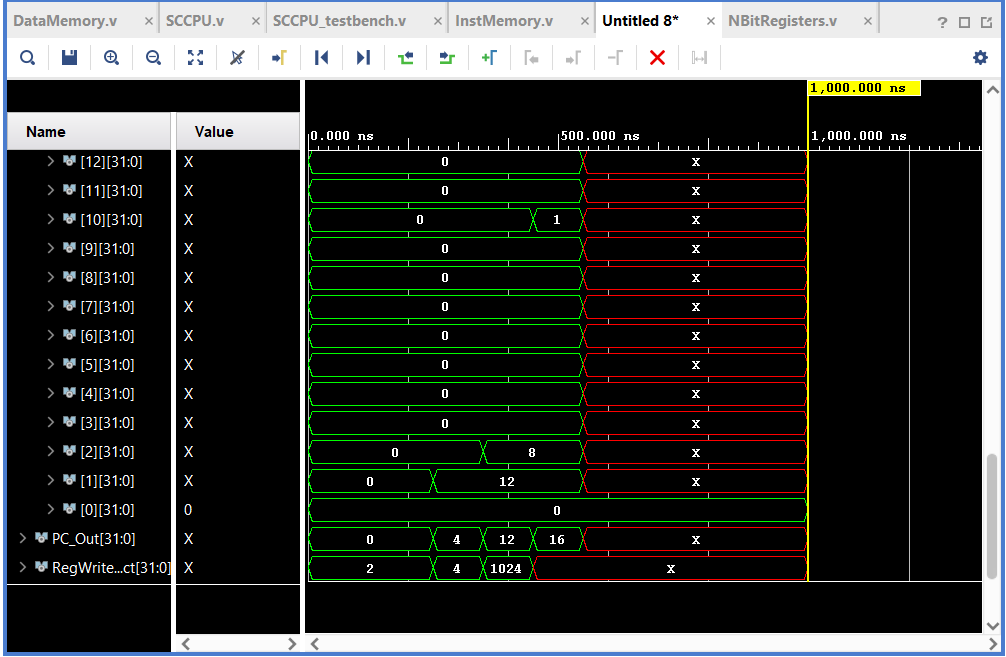
1. **JALR**

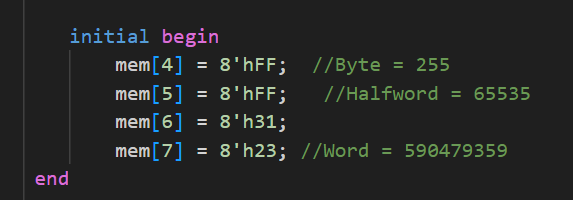
0x00C00093 # addi x1, x0, 12

0x00008167 # jalr x2, x1, 0

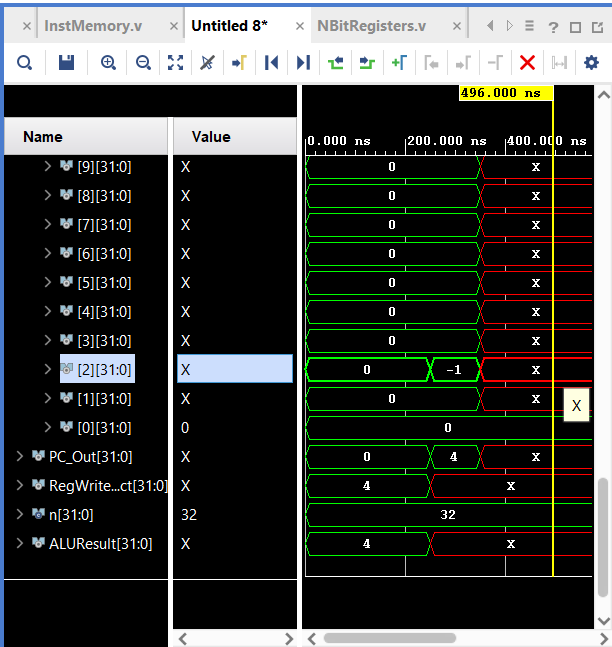
0x00000513 # addi x10, x0, 0 (skipped)

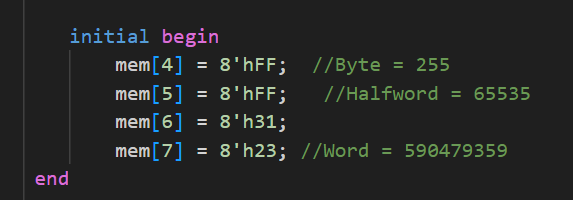
0x00100513 # addi x10, x0, 1 (target)



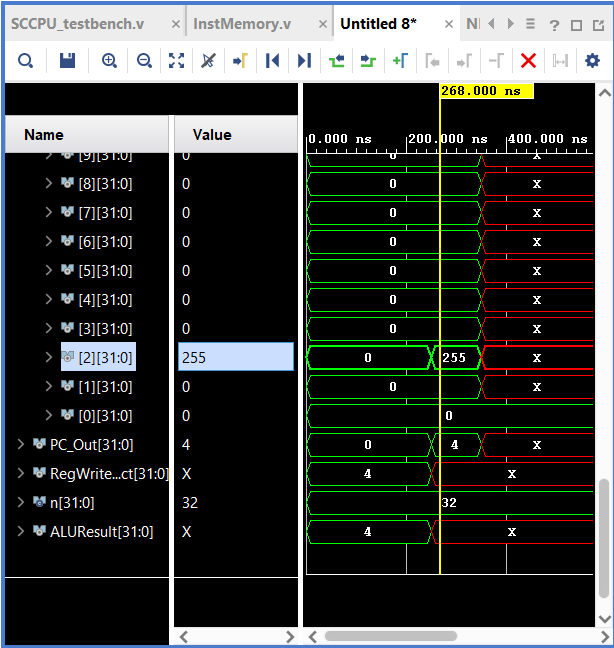
1.  **LB**

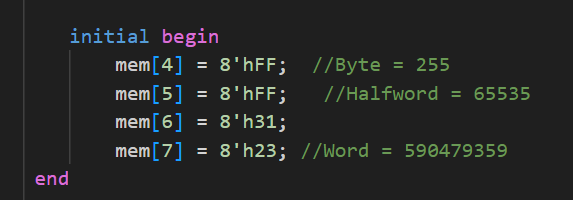
0x00400103 # lb x2, 4(x0)



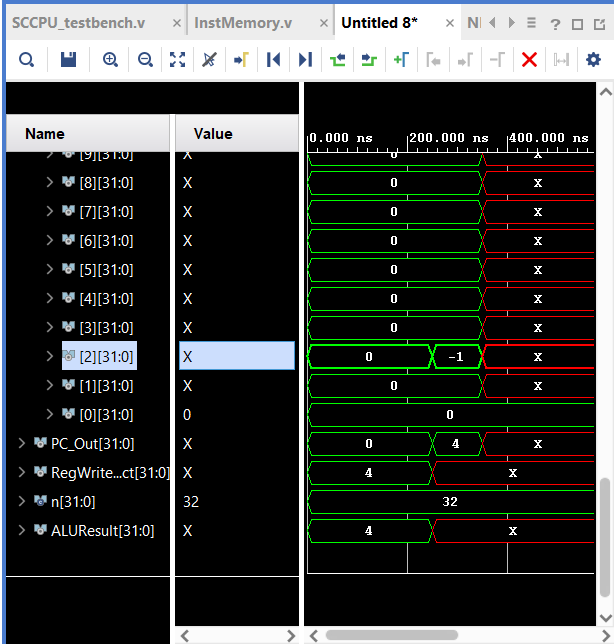
1. **LBU**

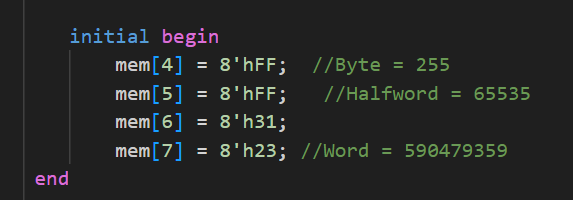
0x00404103 # lbu x2, 4(x0)



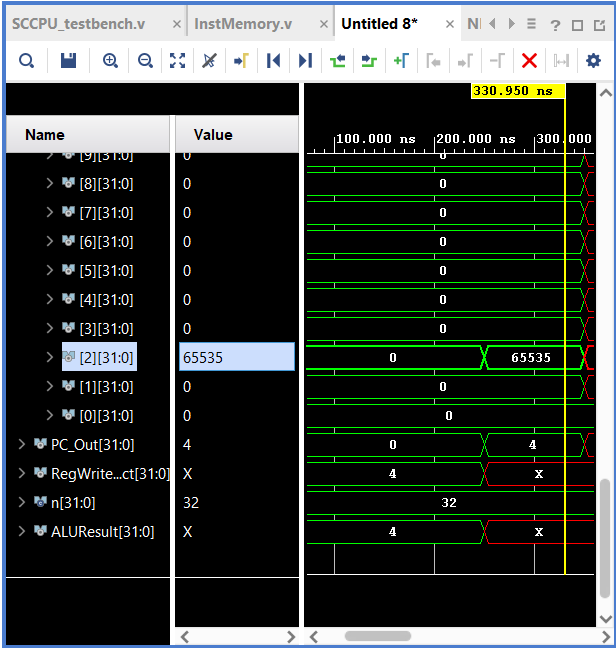
1. **LH**

0x00401103 # lh x2, 4(x0)



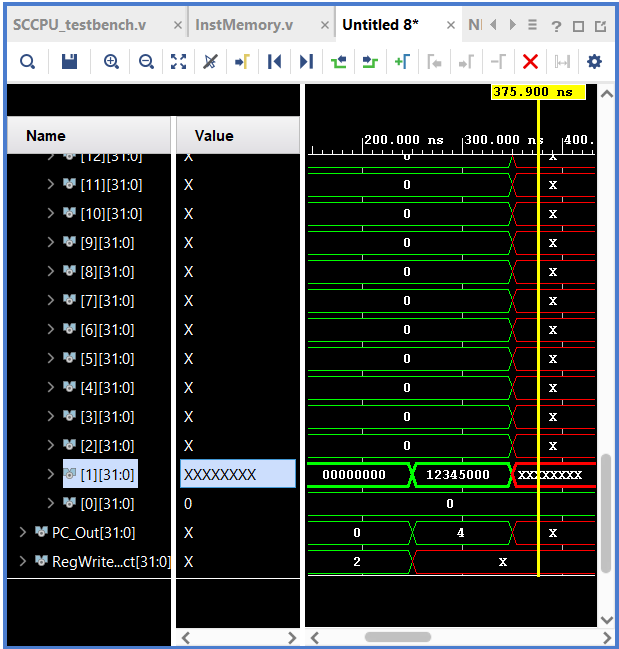
1. **LHU**

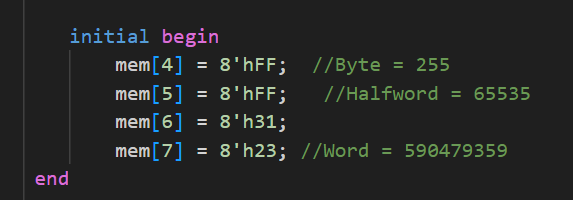
0x00405103 # lhu x2, 4(x0)

****

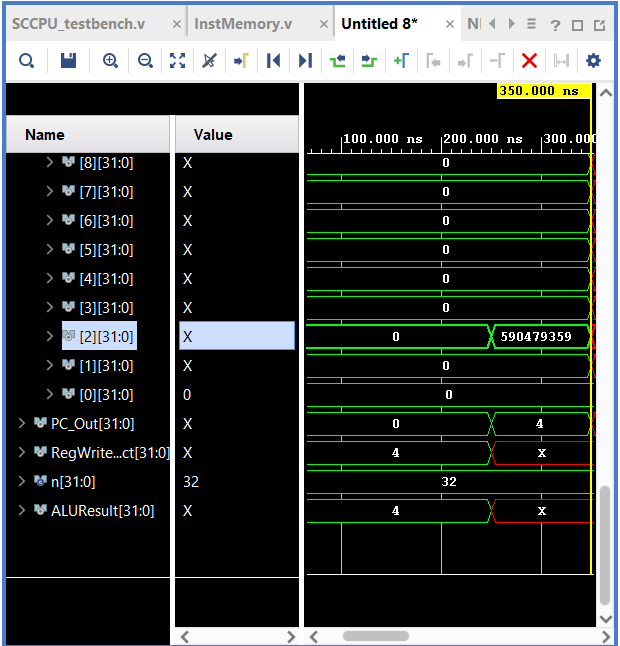
1. **LUI**

0x123450B7 # lui x1, 0x12345



1. **LW**

0x00402103 # lw x2, 4(x0)

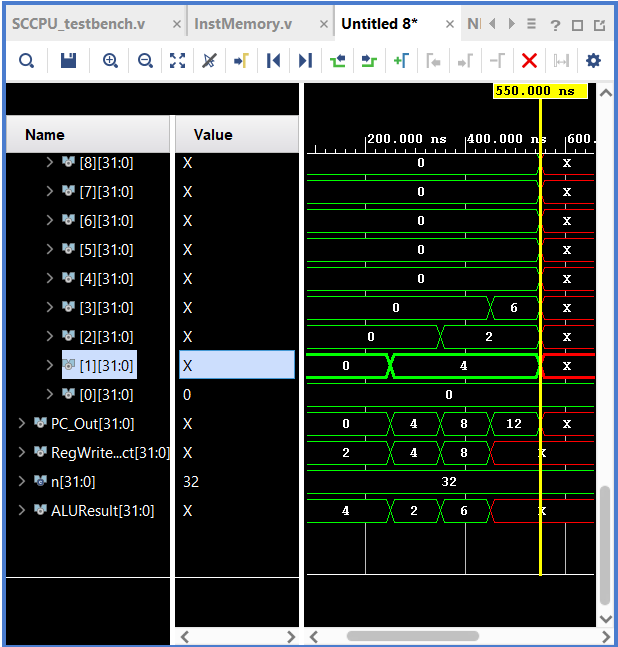


1. **OR**

0x00400093 # addi x1, x0, 4

0x00200113 # addi x2, x0, 2

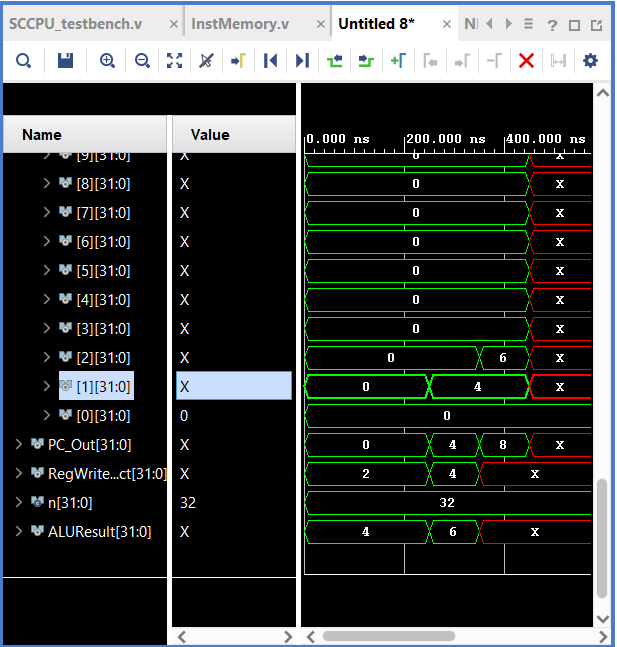
0x0020E1B3 # or x3, x1, x2



1. **ORI**

0x00400093 # addi x1, x0, 4

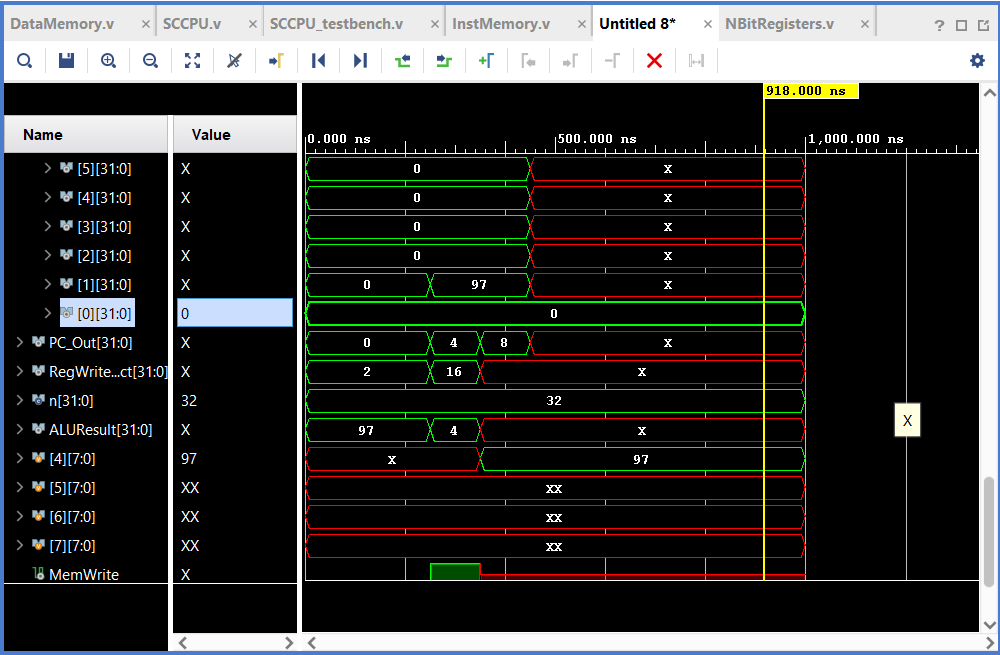
0x0020E113 # ori x2, x1, 2



1. **SB**

0x06100093 # addi x1, x0, 97

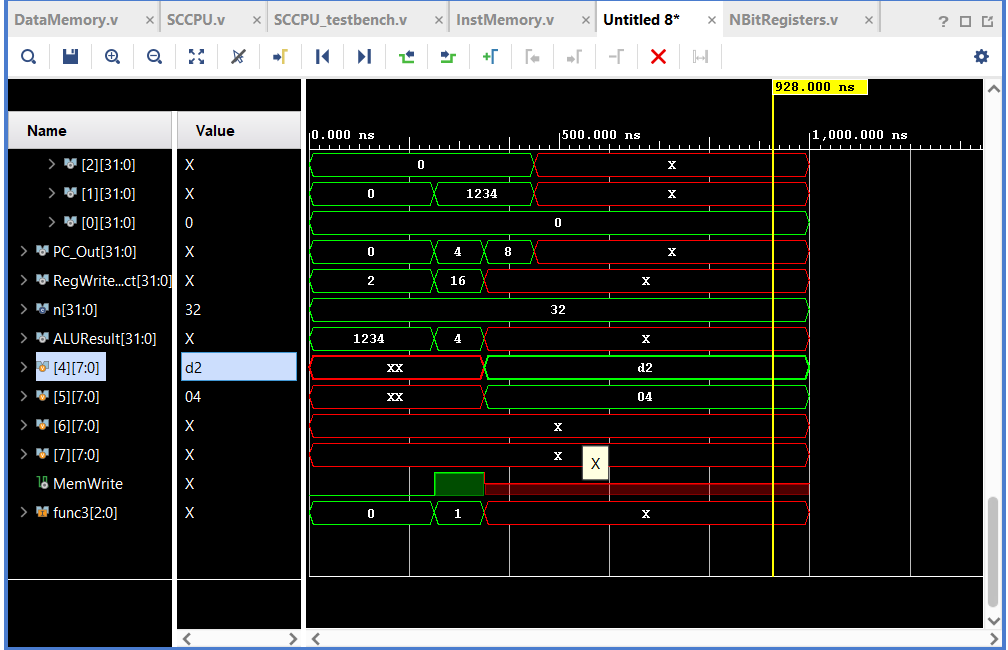
0x00100223 # sb x1, 4(x0)



1. **SH**

0x4D200093 # addi x1, x0, 1234 (04D2 in hex)

0x00101223 # sh x1, 4(x0)

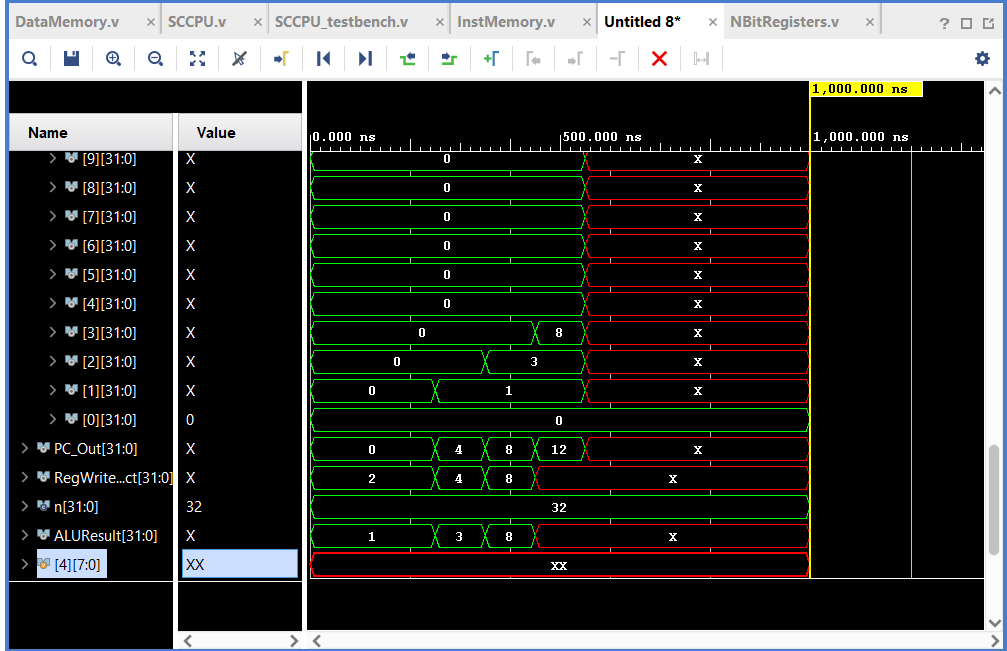


1. **SLL**

0x00100093 # addi x1, x0, 1

0x00300113 # addi x2, x0, 3

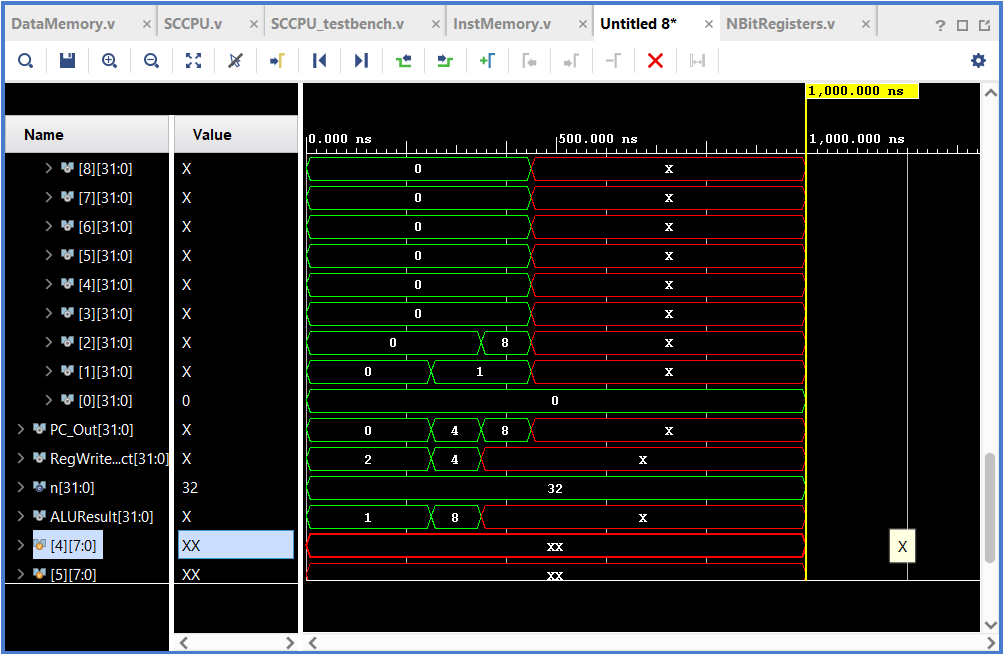
0x002091B3 # sll x3, x1, x2



1. **SLLI**

0x00100093 # addi x1, x0, 1

0x00309113 # slli x2, x1, 3

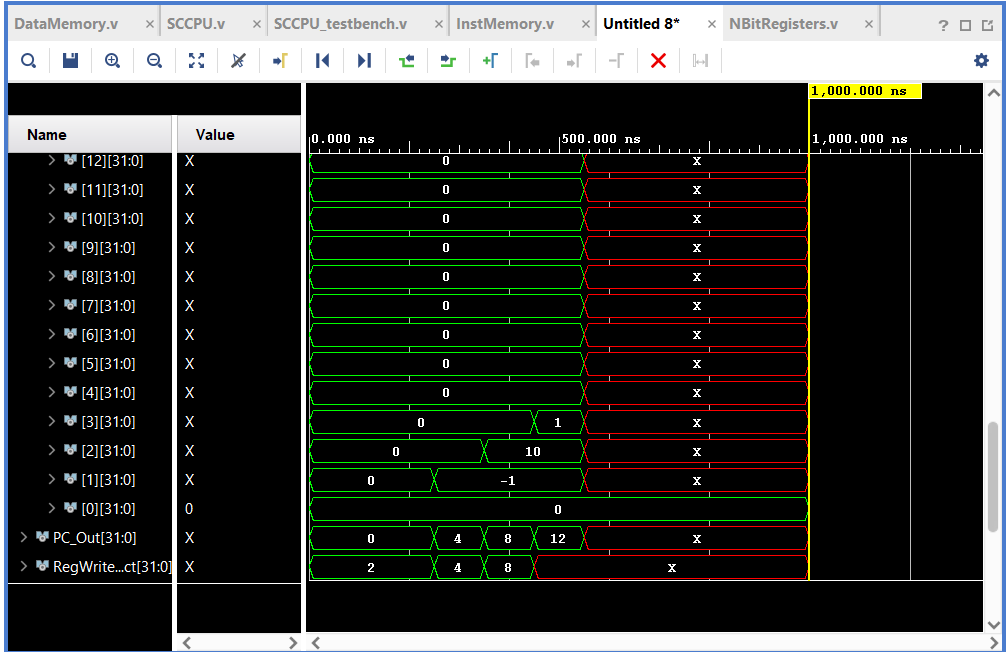


1. **SLT**

0xFFF00093 # addi x1, x0, -1

0x00A00113 # addi x2, x0, 10

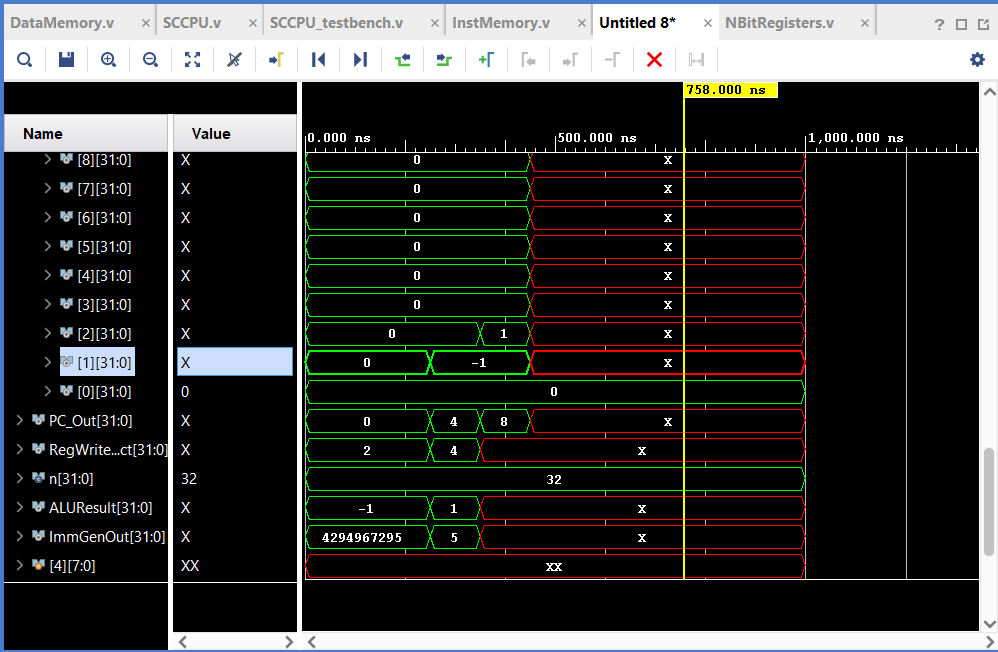
0x0020A1B3 # slt x3, x1, x2



1. **SLTI**

0xFFF00093 # addi x1, x0, -1

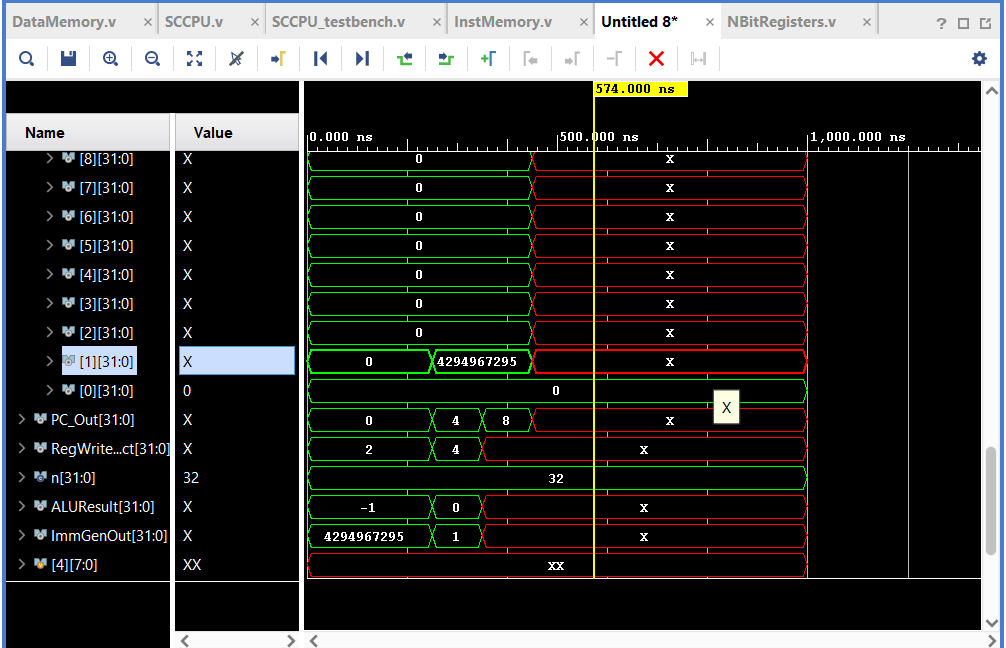
0x0050A113 # slti x2, x1, 5



1. **SLTIU**

0xFFF00093 # addi x1, x0, -1

0x0010b113 # sltiu x2, x1, 1

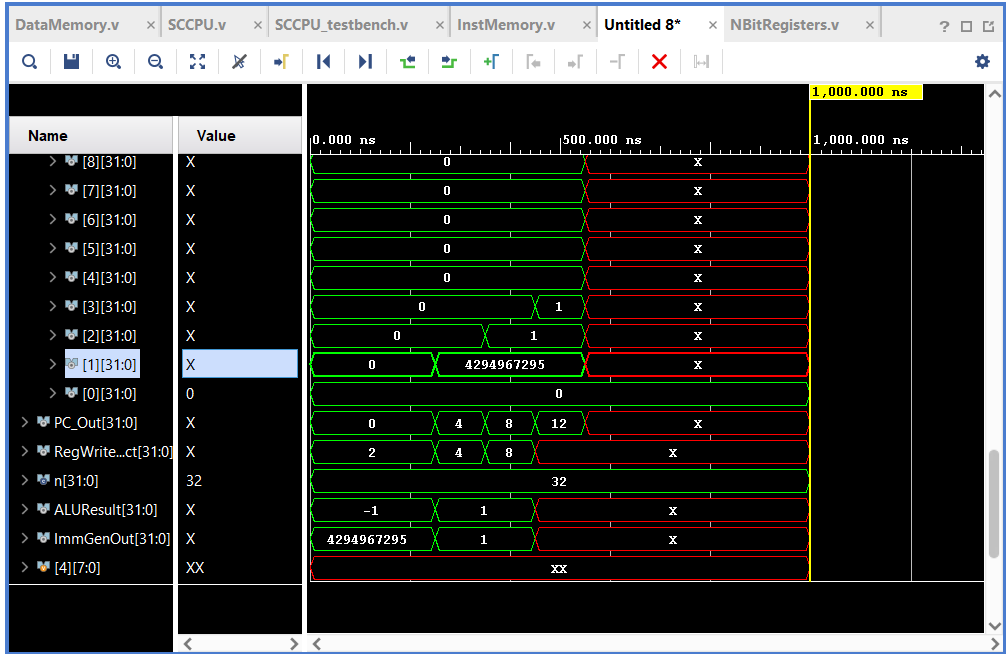


1. **SLTU**

0xFFF00093 # addi x1, x0, -1

0x00100113 # addi x2, x0, 1

0x001131b3 # sltu x3, x2, x1

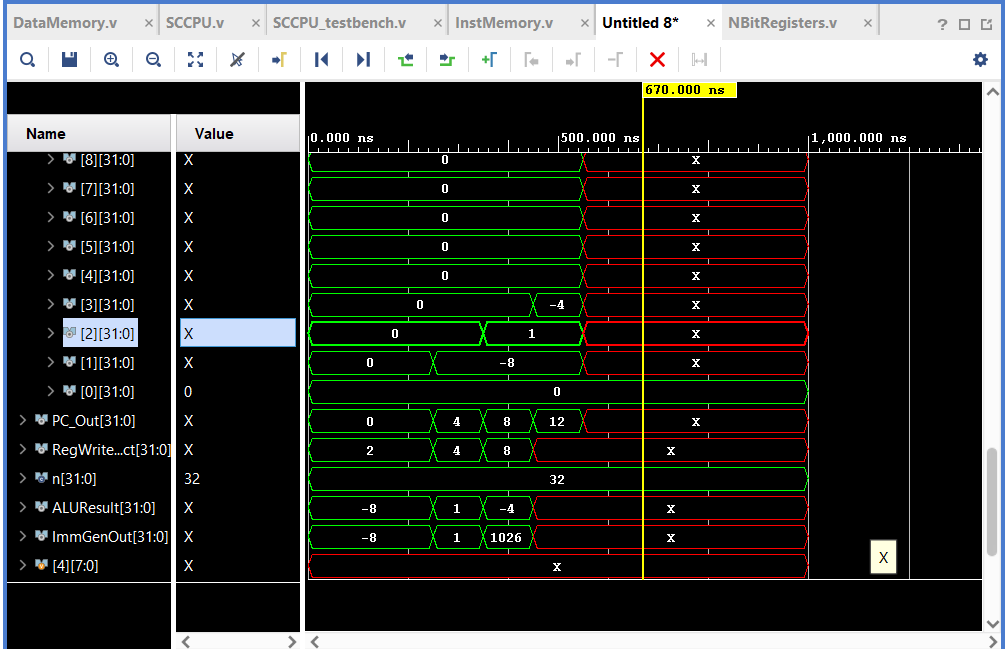


1. **SRA**

0xFF800093 # addi x1, x0, -8

0x00100113 # addi x2, x0, 1

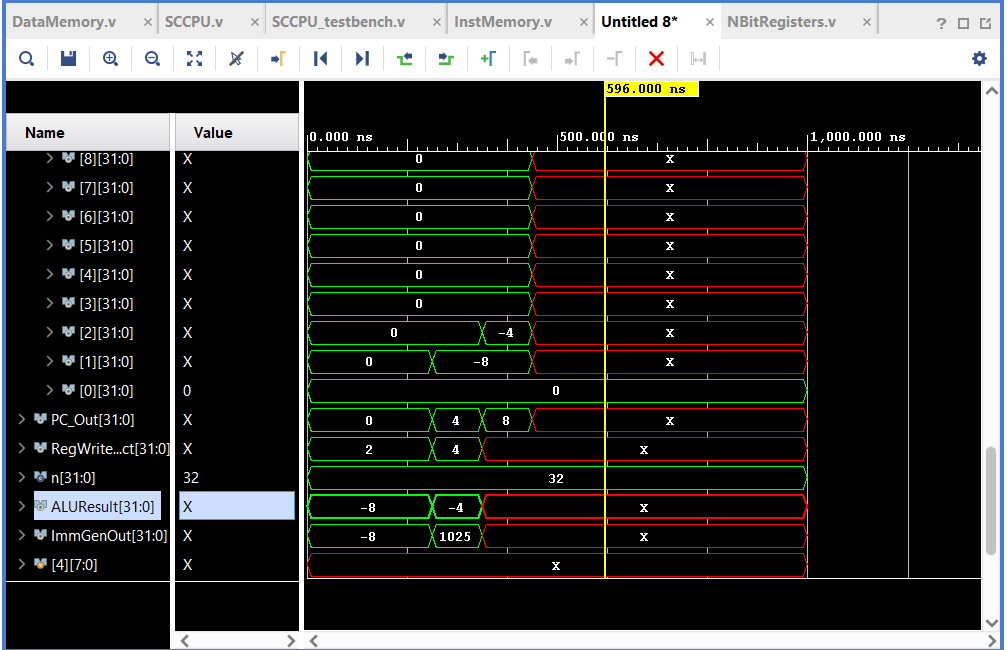
0x4020D1B3 # sra x3, x1, x2



1. **SRAI**

0xFF800093 # addi x1, x0, -8

0x4010D113 # srai x2, x1, 1

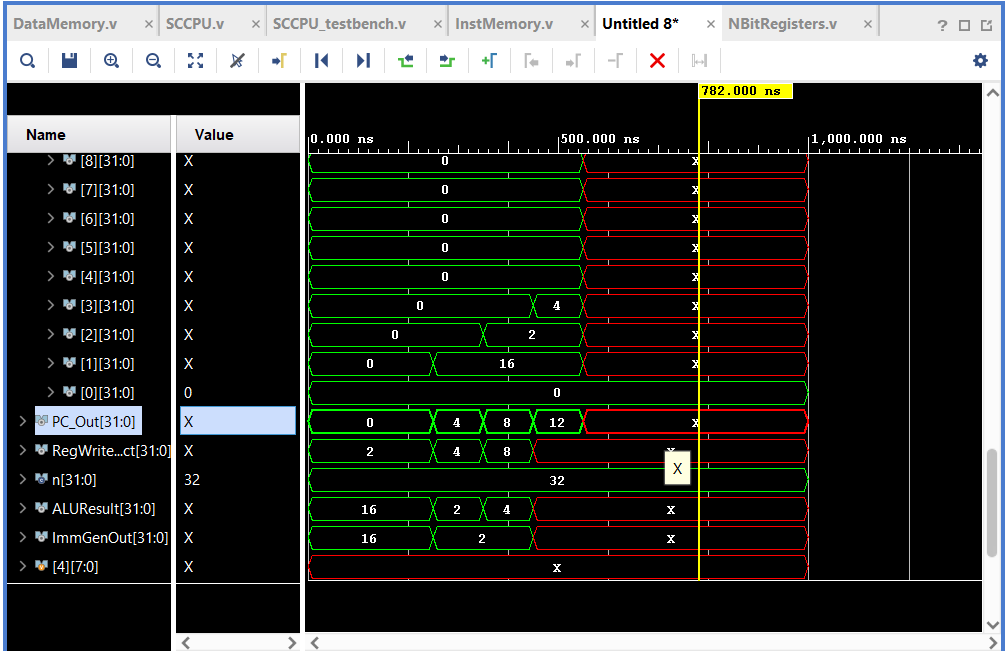


1. **SRL**

0x01000093 # addi x1, x0, 16

0x00200113 # addi x2, x0, 2

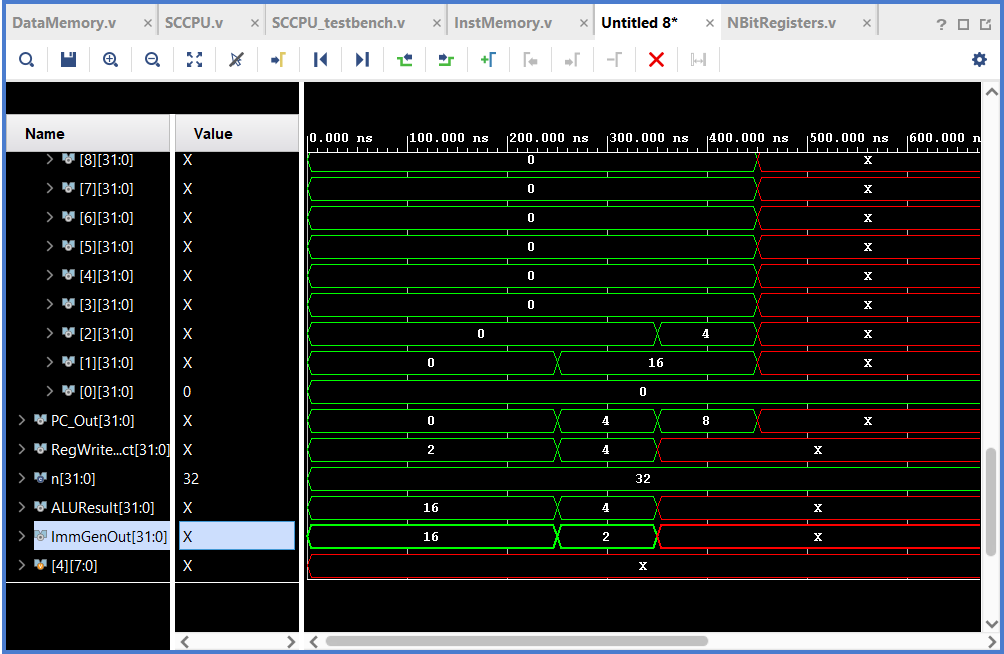
0x0020D1B3 # srl x3, x1, x2



1. **SRLI**

0x01000093 # addi x1, x0, 16

0x0020D113 # srli x2, x1, 2

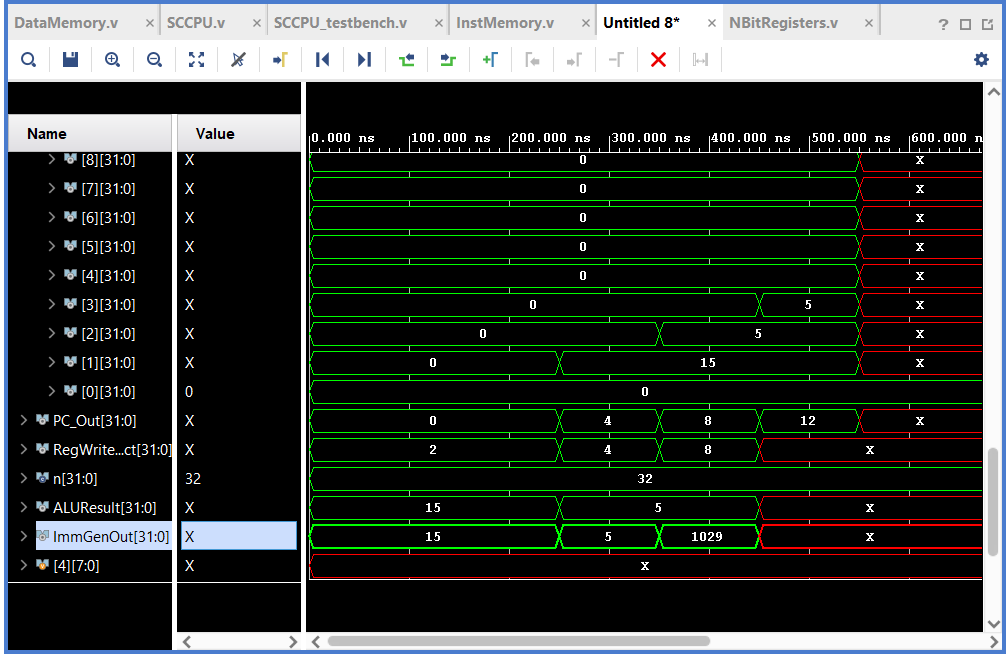


1. **SUB**

0x00F00093 # addi x1, x0, 15

0x00500113 # addi x2, x0, 5

0x405101B3 # sub x3, x1, x2

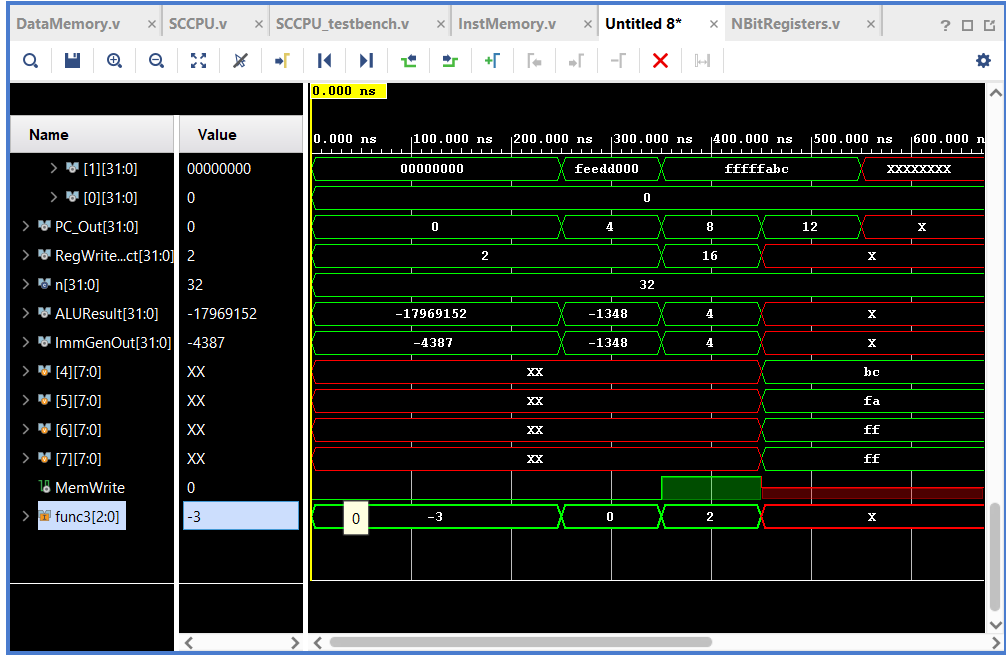


1. **SW**

0xfeedd0b7 #lui x1, 0xFEEDD

0xABCD0093 # addi x1, x0, 0xABC

0x00102223 # sw x1, 4(x0)

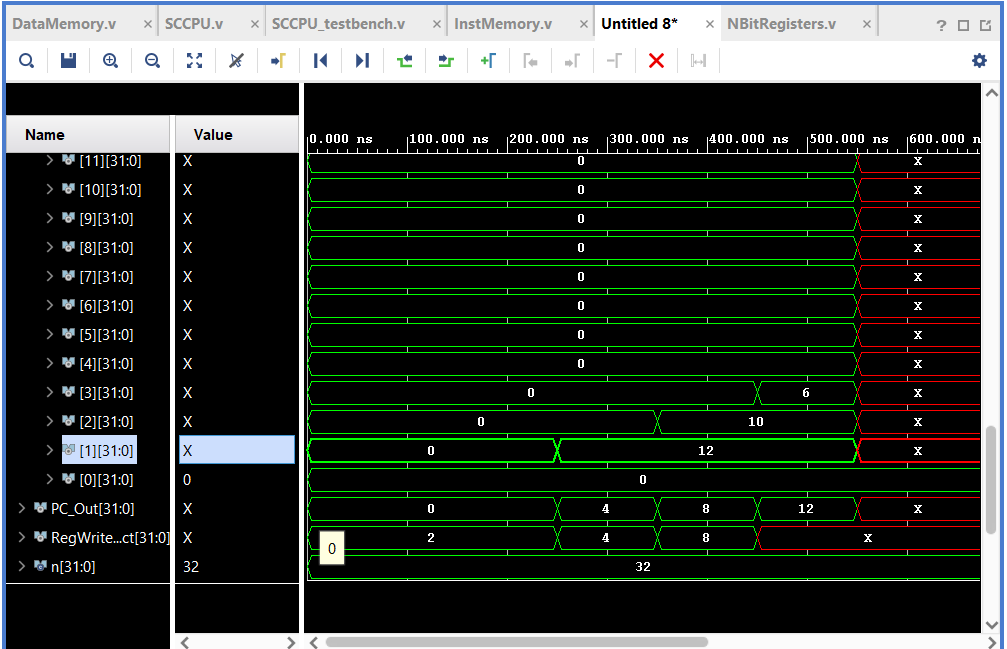


1. **XOR**

0x00C00093 # addi x1, x0, 12

0x00A00113 # addi x2, x0, 10

0x0020C1B3 # xor x3, x1, x2 (1100 ^ 1010) = 0110 = 6



1. **XORI**

0x00C00093 # addi x1, x0, 12

0x00A0C113 # xori x2, x1, 10 = 6

