PIPELINE VERIFICATION

We have to make sure that our Pipeline implementation correctly resolves all the hazards. There are 3 types of hazards that occur

- Data Hazard
- Load Hazard
- Control Hazard

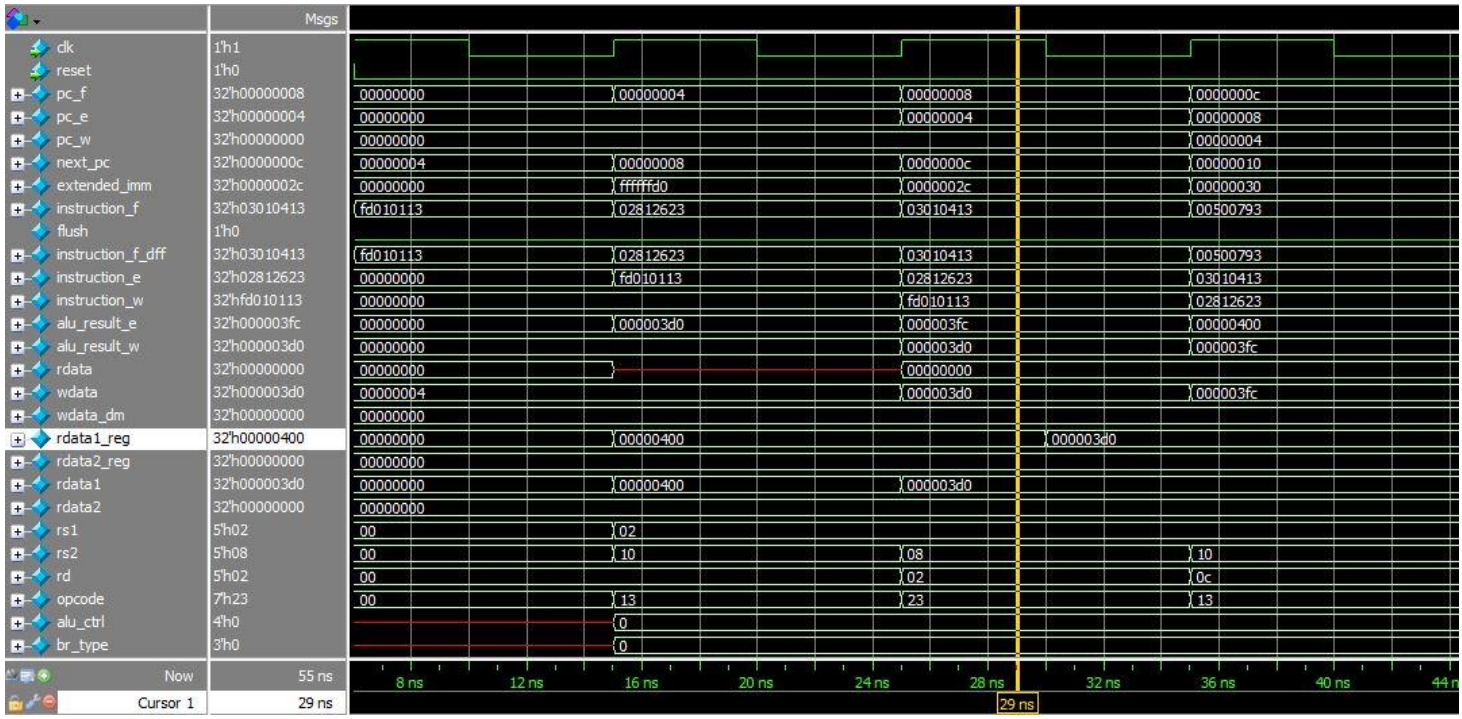
Data Hazard

Data Hazard occurs when one instruction uses the destination register of the previous instruction as operand. The instructions demonstrating Data Hazard in this example are

0: fd010113 addi sp,sp,-48

4: 02812623 sw s0,44(sp)

Simulation



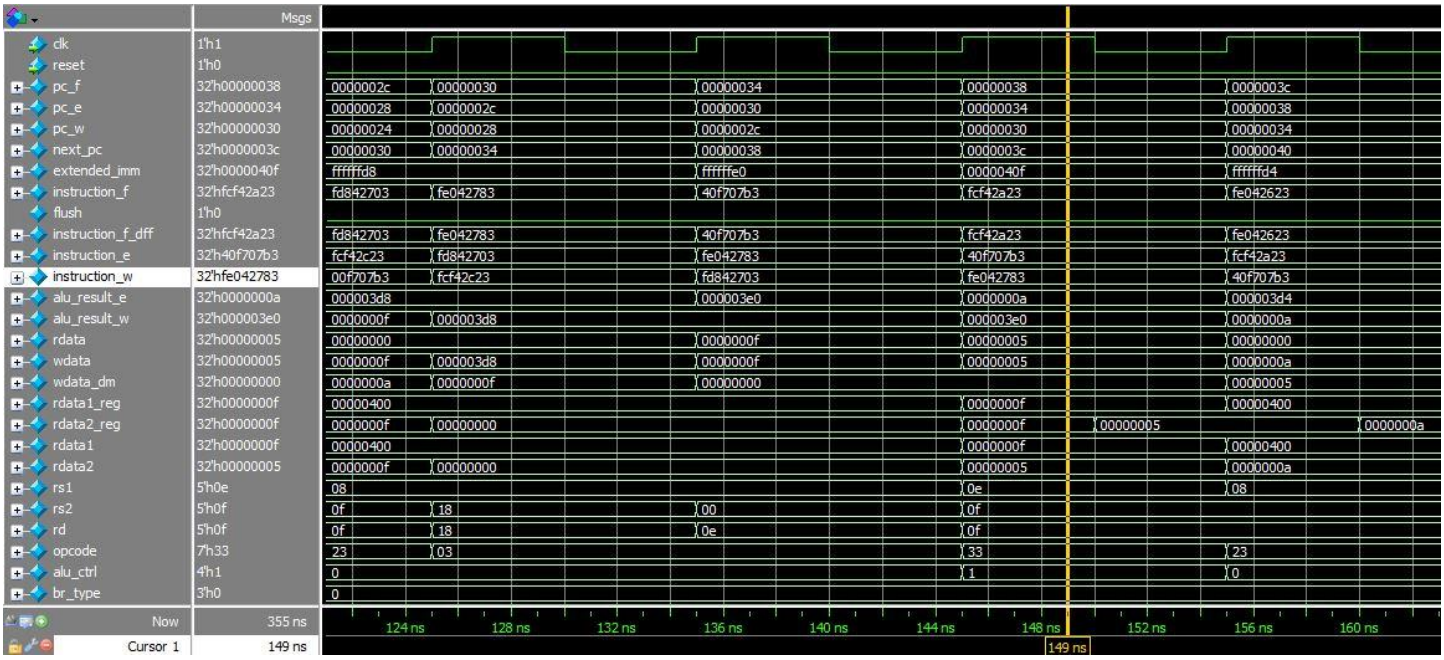
We can observe that sw instruction uses sp as an operand, but previous addi has sp as destination register. Register values **rdata1_reg** and **rdata2_reg** are from the register file and **rdata1** and **rdata2** are the ones being fed into the ALU. As this is a data hazard, rdata1 and rdata1_reg do not have the same value. Register file (rdata1_reg) has the wrong value, therefore the forwarding unit select value from the writeback stage and that value is fed into the ALU. We can observe that rdata1 has a different value from rdata1_reg. This correctly resolves the hazard.

Load Hazard

Load Hazard occurs when one instruction uses the destination register of the previous instruction as operand, but the previous instruction was a load instruction. The instructions demonstrating Load Hazard in this example are

34:	**40f707b3**	**sub**	**a5,a4,a5**
38:	**fcf42a23**	**sw**	**a5,-44(s0)**

Simulation



In this case, the forwarding stall unit again provides us with the correct value. It provides the data at the corresponding address of the data memory, instead of just providing us with the result of the ALU (which is just the address).

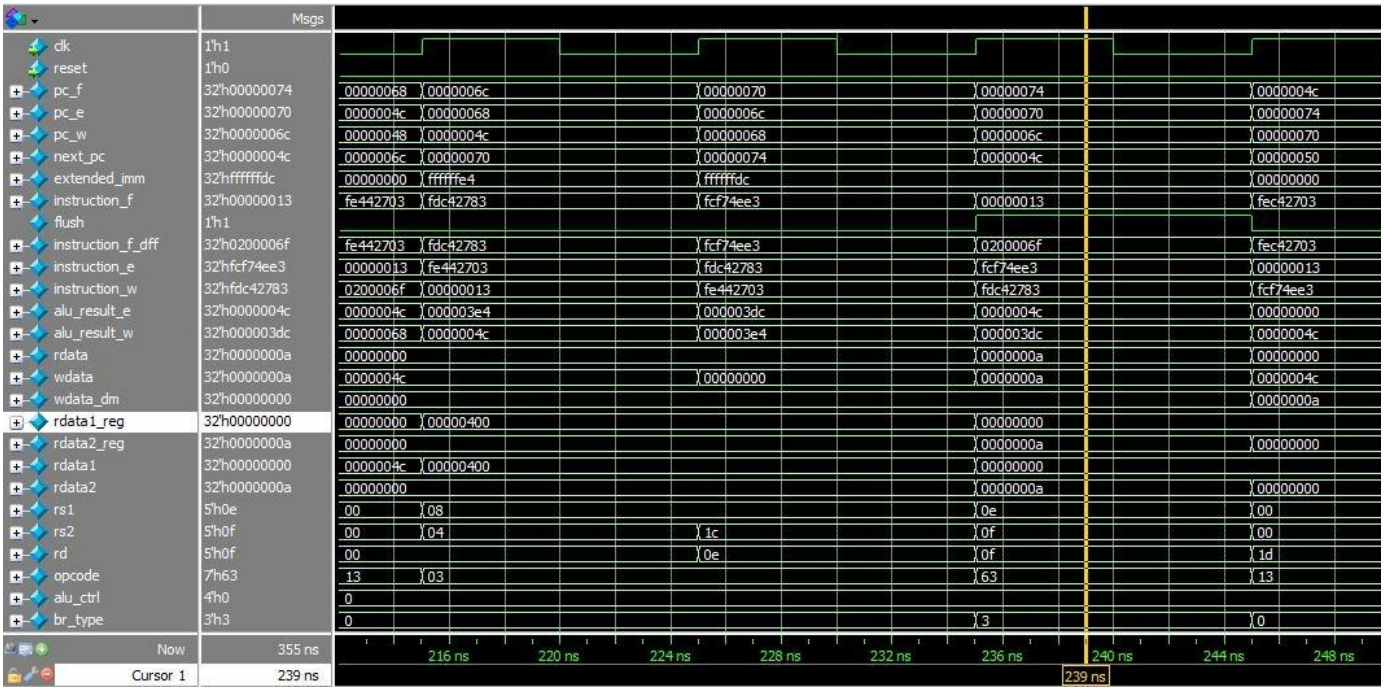
Control Hazard

Control Hazard occurs whenever we jump or take a branch. As the branch/jump instruction is in the execute phase, wrong instructions get loaded into the fetch stage. We don't want to execute those instructions as we are jumping or branching to some other point in code. So we flush the fetch stage to resolve this hazard. Let us execute the following instructions.

70: fcf74ee3 blt a4,a5,4c <main+0x4c>

74: 0200006f j 94 <main+0x94>

Simulation



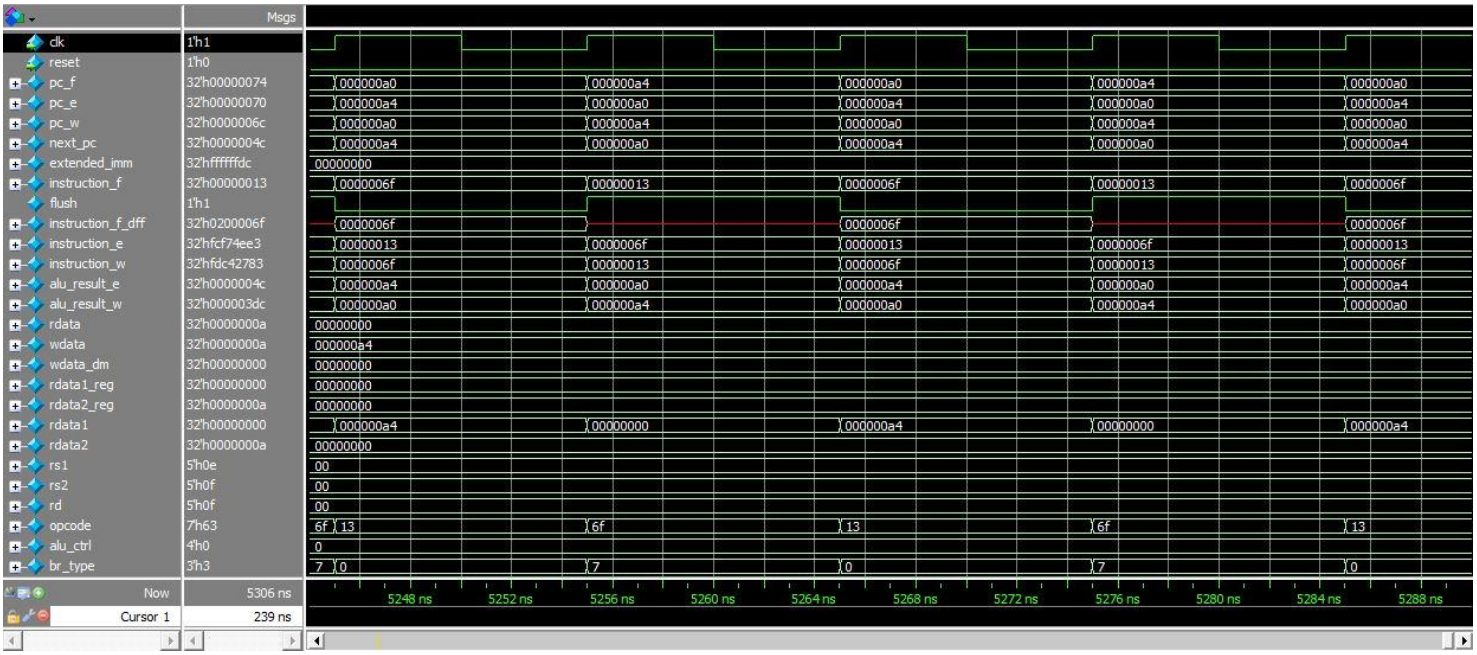
Here, we can observe that as the branch is taken, the fetch stage has been flushed. **Instruction**_f has value of **32'h00000013**, which is a **nop** instruction. This resolves the control hazard.

Let us observe the final output by executing this code.

```
00000000 <main>:
   0: fd010113              addi  sp,sp,-48
   4: 02812623              sw s0,44(sp)
   8: 03010413              addi  s0,sp,48
   c: 00500793              li a5,5
  10: fef42023              sw a5,-32(s0)
  14: 00a00793              li a5,10
  18: fcf42e23              sw a5,-36(s0)
  1c: fe042703              lw a4,-32(s0)
  20: fdc42783              lw a5,-36(s0)
  24: 00f707b3              add   a5,a4,a5
  28: fcf42c23              sw a5,-40(s0)
  2c: fd842703              lw a4,-40(s0)
  30: fe042783              lw a5,-32(s0)
  34: 40f707b3              sub   a5,a4,a5
  38: fcf42a23              sw a5,-44(s0)
  3c: fe042623              sw zero,-20(s0)
  40: fe042423              sw zero,-24(s0)
  44: fe042223              sw zero,-28(s0)
  48: 0200006f              j  68 <main+0x68>
  4c: fec42703              lw a4,-20(s0)
  50: fd442783              lw a5,-44(s0)
  54: 00f707b3              add   a5,a4,a5
  58: fef42623              sw a5,-20(s0)
  5c: fe442783              lw a5,-28(s0)
  60: 00178793              addi  a5,a5,1
  64: fef42223              sw a5,-28(s0)
  68: fe442703              lw a4,-28(s0)
  6c: fdc42783              lw a5,-36(s0)
  70: fcf74ee3              blt   a4,a5,4c <main+0x4c>
  74: 0200006f              j  94 <main+0x94>
  78: fec42703              lw a4,-20(s0)
  7c: fe042783              lw a5,-32(s0)
  80: 40f707b3              sub   a5,a4,a5
  84: fef42623              sw a5,-20(s0)
  88: fe842783              lw a5,-24(s0)
  8c: 00178793              addi  a5,a5,1
  90: fef42423              sw a5,-24(s0)
  94: fec42703              lw a4,-20(s0)
  98: fe042783              lw a5,-32(s0)
  9c: fcf75ee3              bge   a4,a5,78 <main+0x78>
  a0: 0000006f              j  a0 <main+0xa0>

```

Final Waveform



We can observe that the processor has entered the final instruction, and jumps back to it.

Memory Values