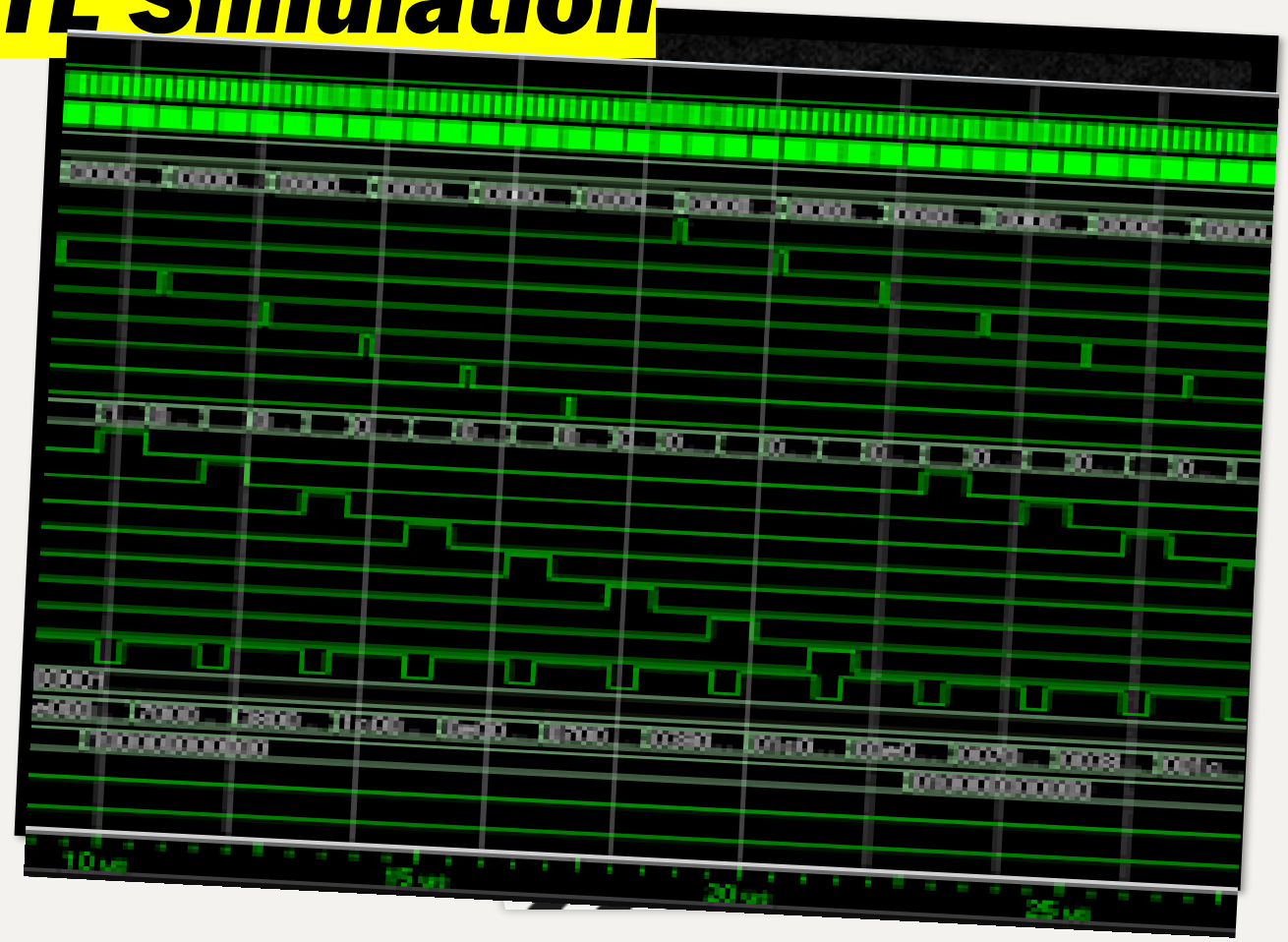
EE-309 Project

-Harshraj Chaudhari (210040060)
-Ishaan Manhaar (210070033)
-Jainesh Mehta (210071001)
-Khushagra Gehlot (21D070041)

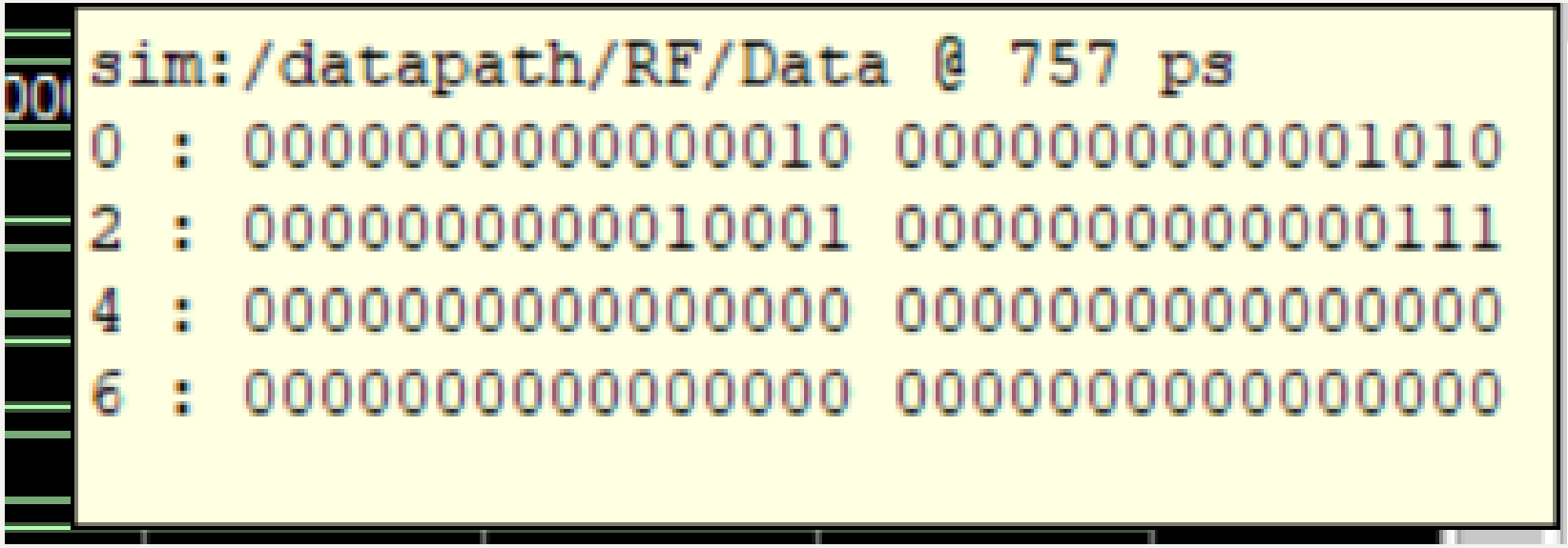
RTL Simulation



Instructions used

- ADA => "00010010","10011000" // R3=R1+R2 (works)
- ACC => "00010010","10011110" // R3=R1+cpl(R2) if C=1 (works)
- ACW => "00010010","10011111" // R3=R1+cpl(R2)+Carry (works)
- ADI => "00000010","10001010" // R2=R1+Imm(works)
- LLI => "00110100","10101010" // R2<=Imm(works)
- LW => "01000010","10000010" // R1<=RAM(R2+2)(works)
- SW => "01010010","10000010" // R1=>RAM(R2+2)(works)
- LM => "01100010","00010010" // R6<=M(R1+1),R3<=M(R1+4)(works)
- SM => "01110010","00010010" // R6=>M(R1+1),R3=>M(R1+4)(works)
- BEQ => "10000010","01000011" // branches to PC+6(works)
- JLR => "11010010","10000011" // branches to R2, PC+2 in R1(works)

ADA followed by ADA with immediate dependency



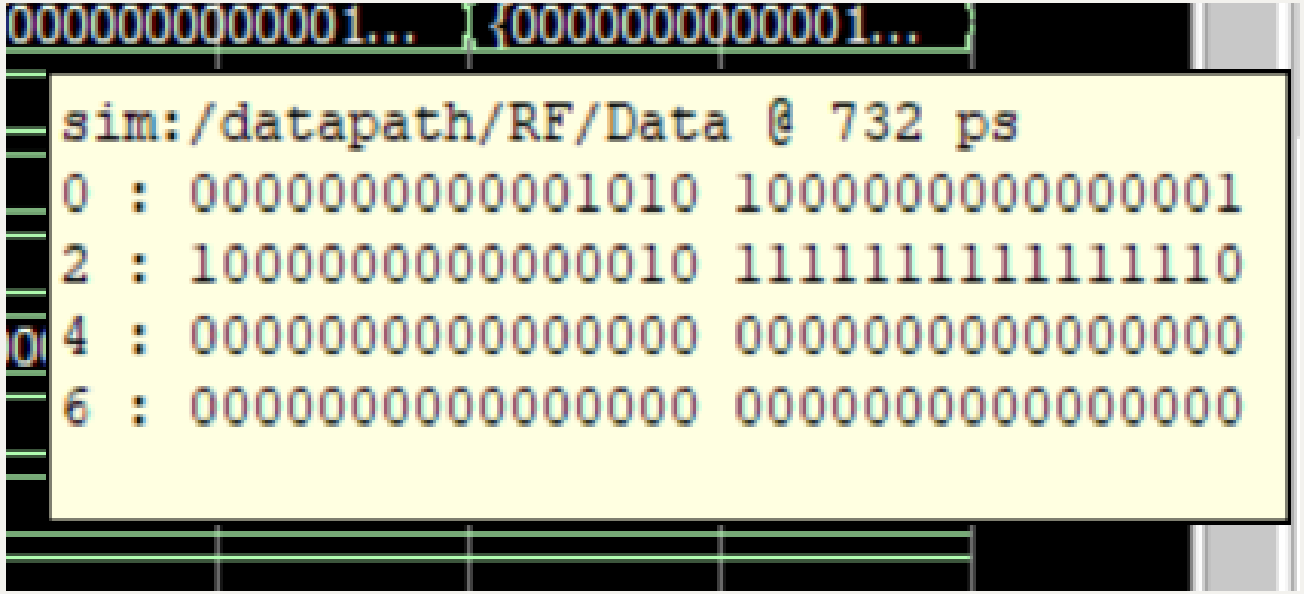
Adds content of R1 and R2 and Stores in R3. Here it is Followed by another ADA instruction with immediate Dependency.

ADA Followed by ADI

```
sim:/datapath/RF/Data @ 743 ps
0 : 0000000000001010 1000000000000001
2 : 1000000000001011 0000000000000011
4 : 0000000000000000 0000000000000000
6 : 0000000000000000 0000000000000000
```

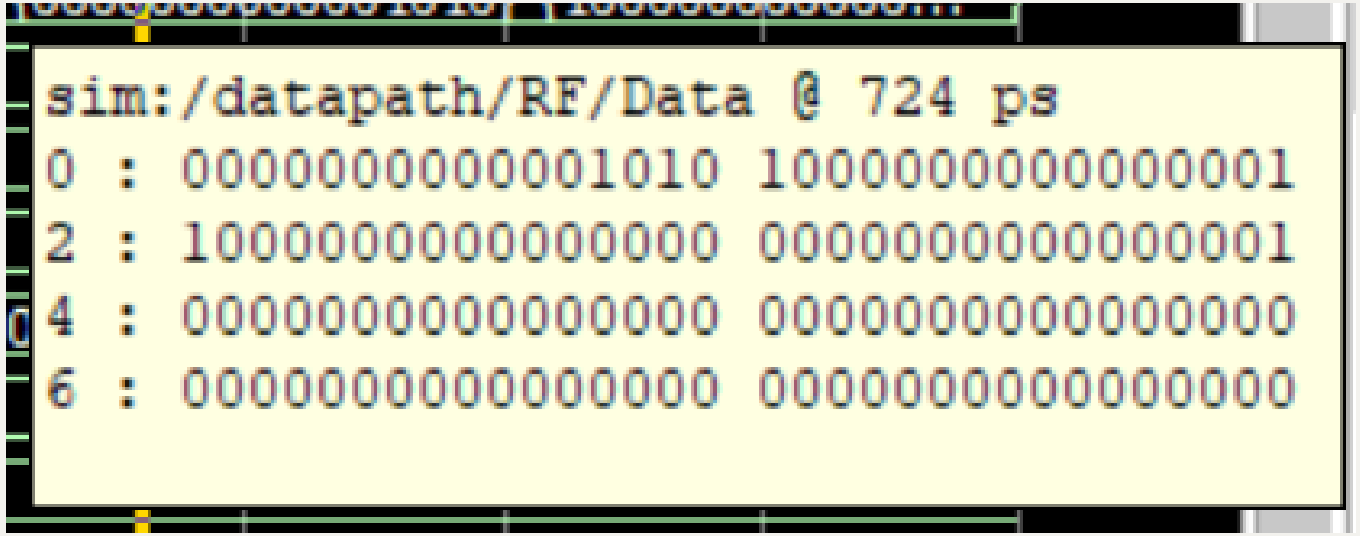
ADA Followed by ADI. No dependency.
Immediate was added with R1 stored at R2.

ADA followed by ACC



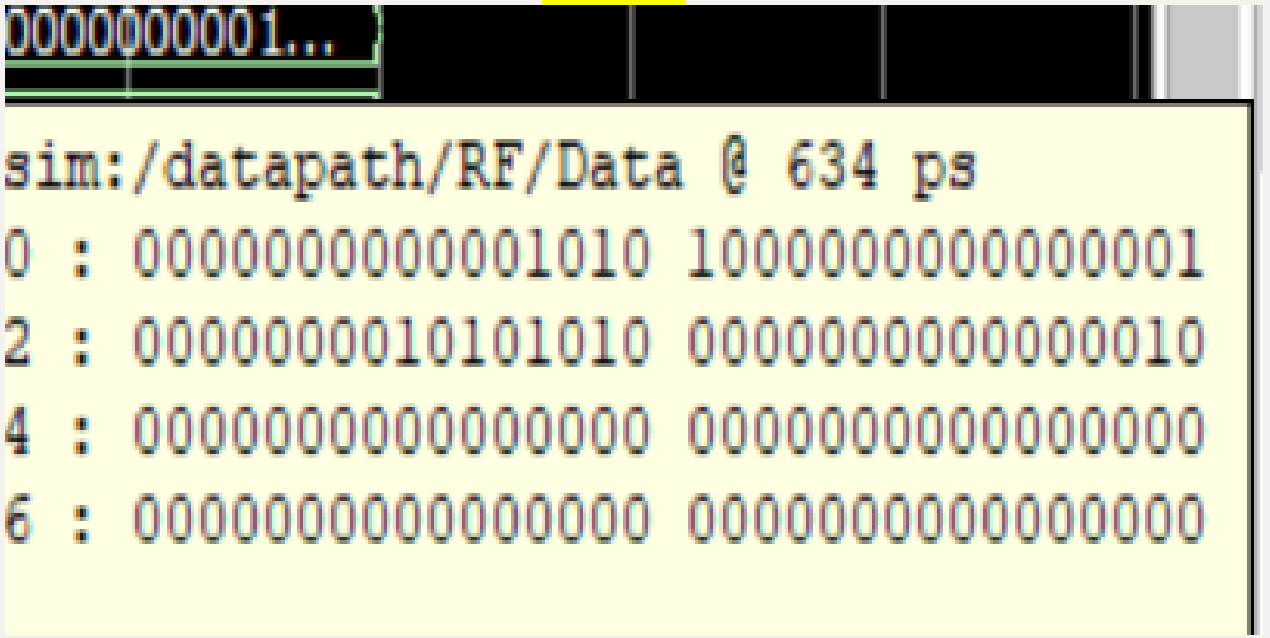
ADA Generates a carry by adding R1 and R2 and storing in R3, then ACC adds R1 with Complement of R2 and stores it in R3.(This happens as carry bit was set in previous instruction).

ADA followed by ACW



Adds R1 with Complement of R2 and carry(Generated due to ADA Instruction) and stores it in R3

LLI



Loads the immediate value into Registor R2

LW and LM

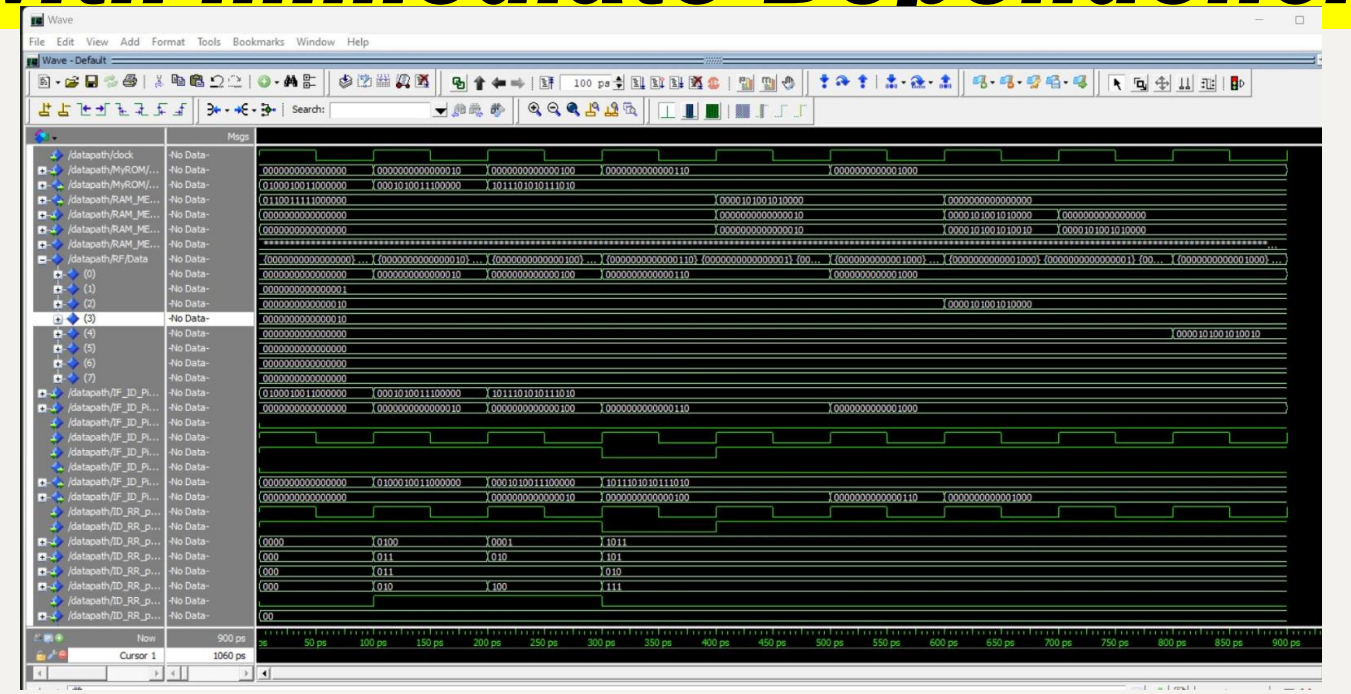
```
sim:/datapath/RF/Data @ 1037 ps
0 : 0000000000000100 0000000000000001
2 : 0000000000000010 0000000000000000
4 : 0000000000000000 0000000000000000
6 : 0101000001010100 0000000000000000
```

Loads the data from R1+1 address in R6 and from R1+4 address in R3

```
sim:/datapath/RF/Data @ 656 ps
0 : 0000000000001010 0101010011000010
2 : 0000000000000010 0000000000000010
4 : 0000000000000000 0000000000000000
6 : 0000000000000000 0000000000000000
```

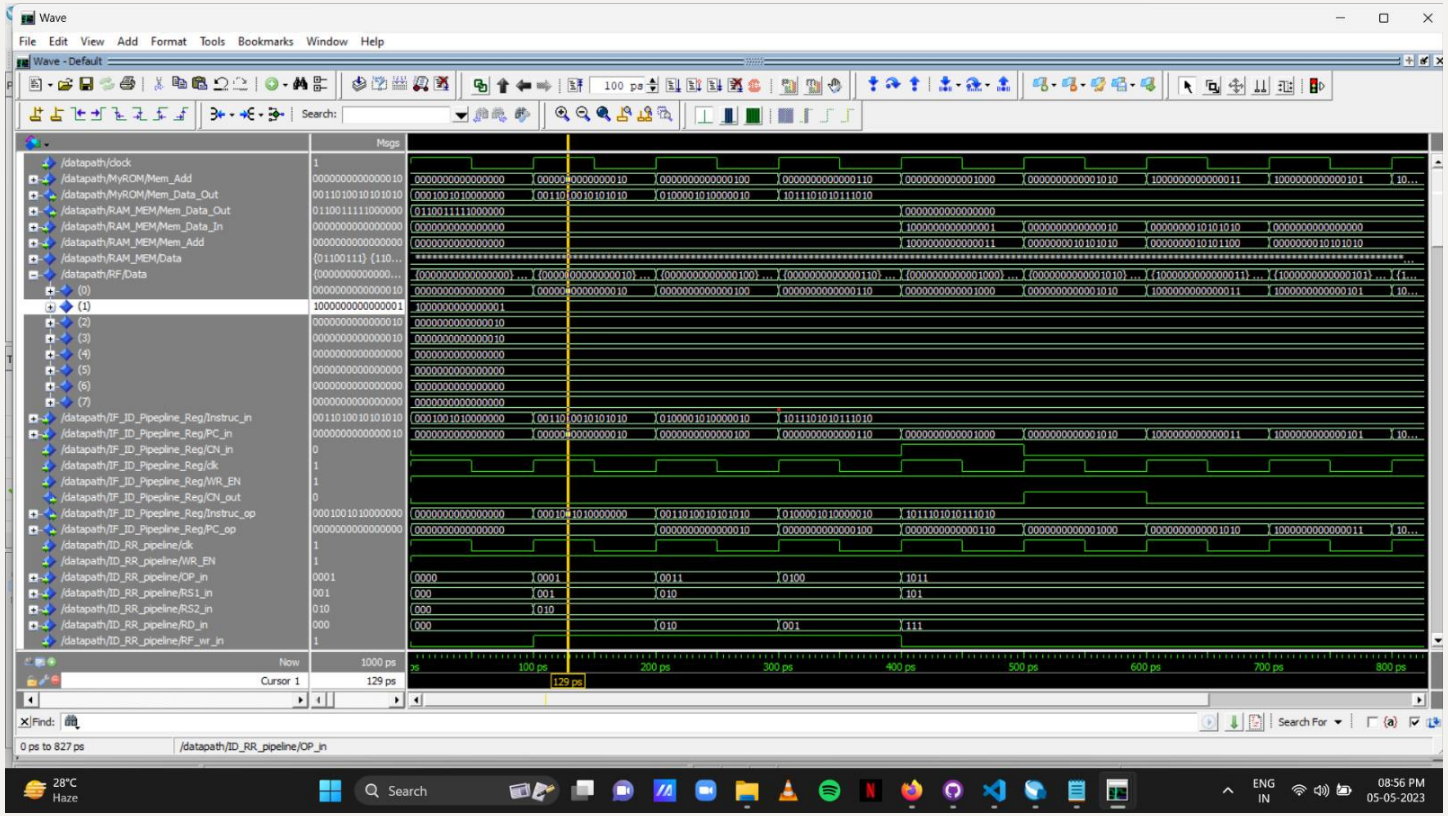
Loads the data from R2+2 address to R1

LW with Immediate Dependencies

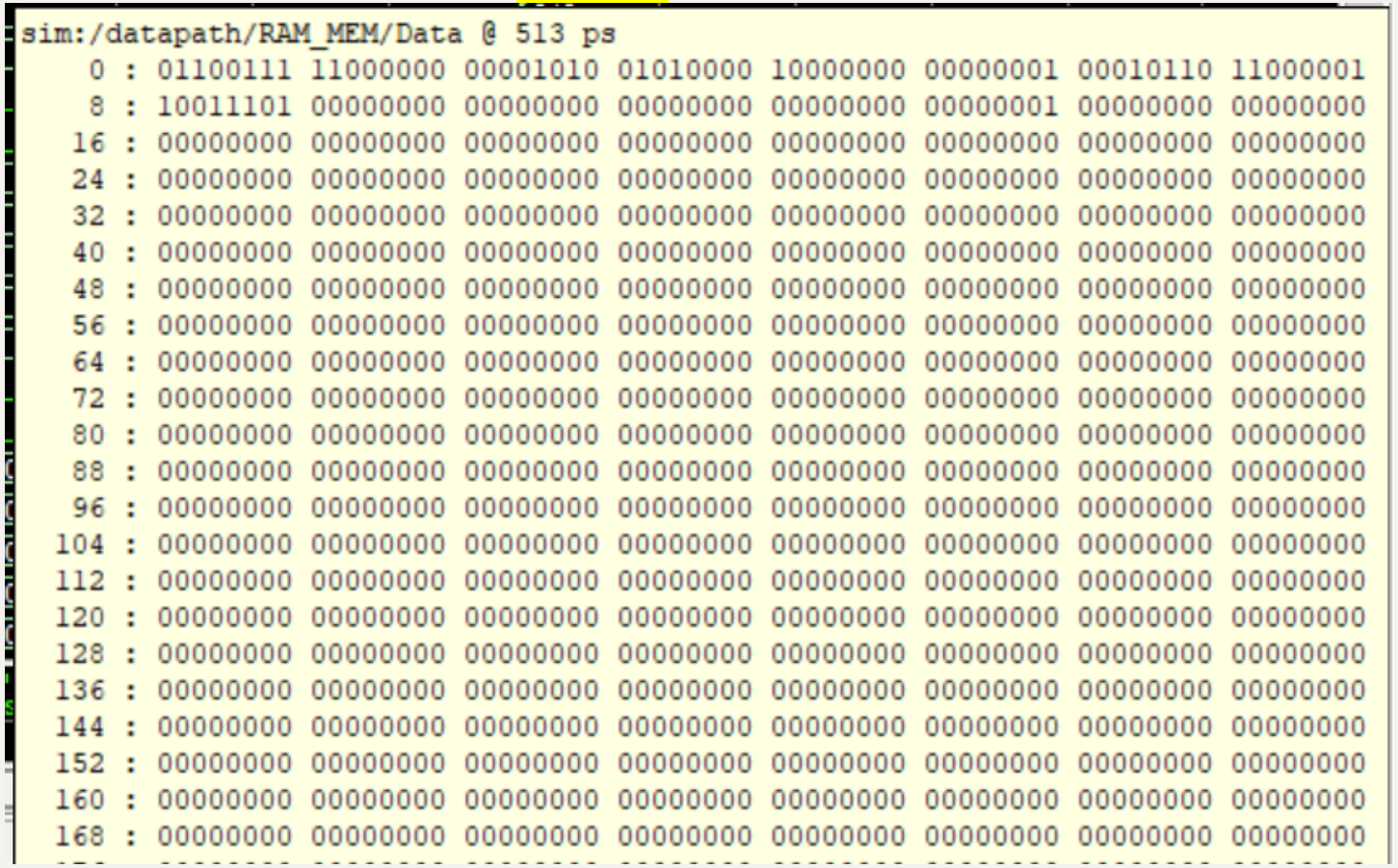


Load with immediate dependency, Hazard is detected and !
Cycle stall is implemented.

LLI with Destination in R0



SW



```
sim:/datapath/RAM_MEM/Data @ 513 ps
    0 : 01100111 11000000 00001010 01010000 10000000 00000001 00010110 11000001
    8 : 10011101 00000000 00000000 00000000 00000000 00000001 00000000 00000000
   16 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   24 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   32 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   40 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   48 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   56 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   64 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   72 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   80 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   88 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   96 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  104 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  112 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  120 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  128 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  136 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  144 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  152 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  160 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  168 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

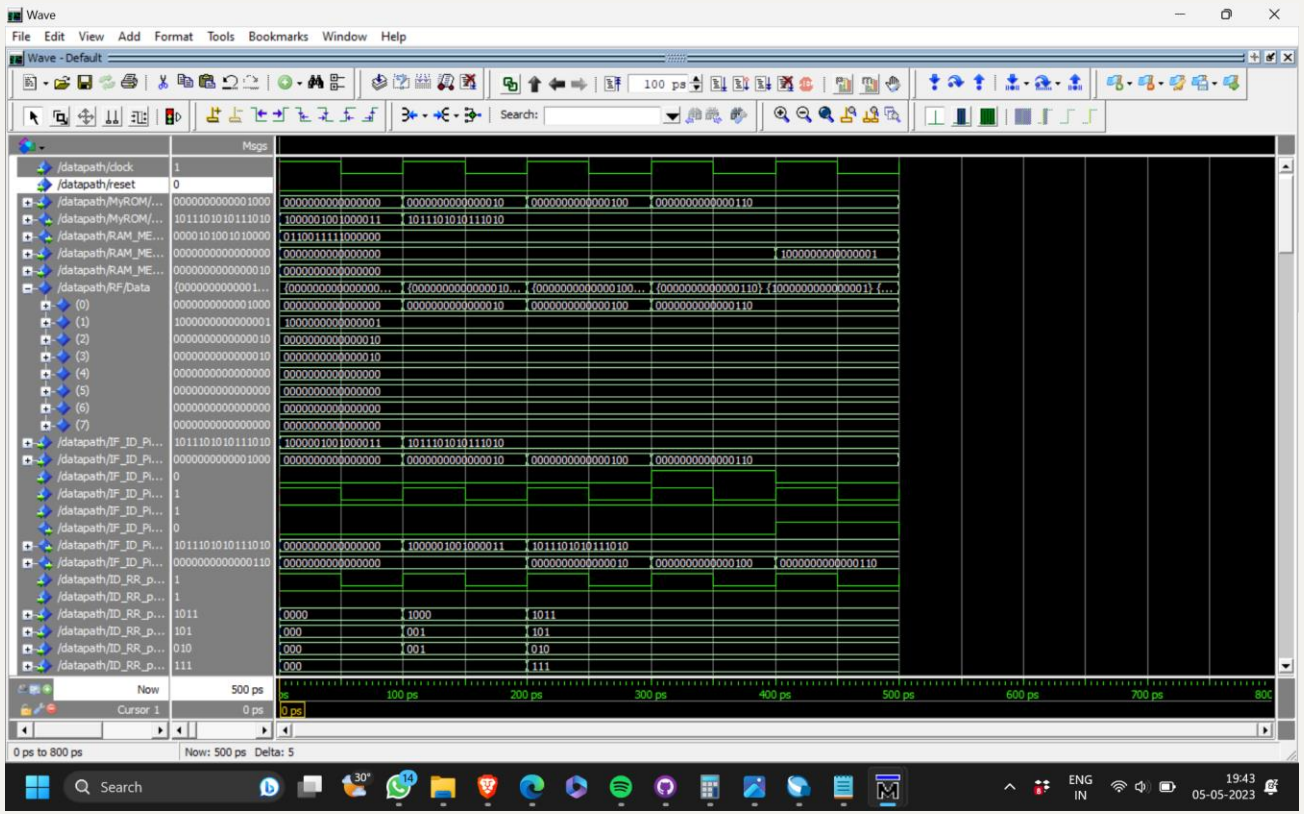
Stores data from R1 into RAM at address R2+2

SM

```
sim:/datapath/RAM_MEM/Data @ 1020 ps
    0 : 01100111 11000000 00001010 11111111 11111111 11000010 00010110 11000001
    8 : 10011101 00000000 00000010 00000000 00000000 00000001 00000000 00000000
   16 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   24 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   32 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   40 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   48 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   56 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   64 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   72 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   80 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   88 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   96 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  104 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  112 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  120 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  128 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  136 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

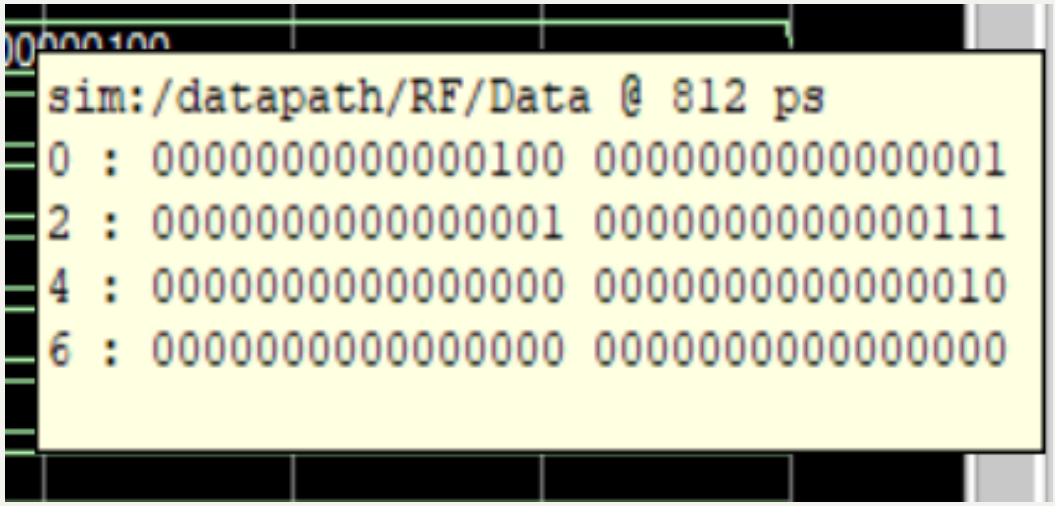
Stores data from R6 and R3 into RAM at address R1+1 and R1+4

BEQ



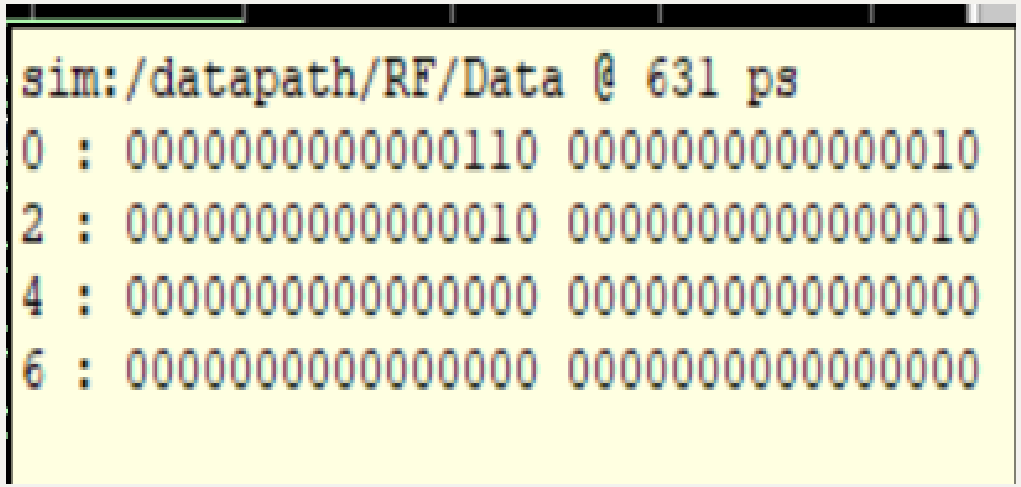
Since this instruction compares the same Register, it branches to PC+6

JAL followed by ADI



Jumps to PC+6 and stores PC+2 in R5. ADI which followed this was Flushed. But the branched load operation occurs

JLR



branches PC to data in R2, and PC+2 is stored in R1. 2 instructions following were Flushed.

Fibonacci Sequence Generator

Instructions Used:
"01101000","00000011": LM := RAM(R4) => R7, RAM(R4+1) => R6
"00001001","00000001": ADI := R4 = R4 + 2
"00011111","10101000": ADA := R5 = R6 + R7
"01011011","00000001": SW := R5 => RAM(R4+2)
"11010010","10000000": JLR := Jumps to first instruction.

RAM should be initialized as RAM(0,1) = 1 and RAM(2,3) = 1
No RF initialization is required.

As can be seen this the Image attached, RAM(0,1) stores 1, RAM(2,3) stores 1, RAM(4,5) stores 2, RAM(6,7) stores 3 and so on. This is the Fibonacci Sequence