

RTL Simulation

EE-309 Project

- Harshraj Chaudhari (210040060)
- Ishaan Manhaar (210070033)
- Jainesh Mehta (210071001)
- Khushagra Gehlot (210070041)



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 ADI

```
sim:/datapath/RF/Data @ 743 ps
0 : 000000000000001010 100000000000000001
2 : 100000000000001011 000000000000000011
4 : 000000000000000000 000000000000000000
6 : 000000000000000000 000000000000000000
```

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



ADA followed by ACC

```
00000000000001... {00000000000001...}
sim:/datapath/RF/Data @ 732 ps
0 : 00000000000001010 10000000000000001
2 : 10000000000000010 11111111111111110
4 : 00000000000000000 00000000000000000
6 : 00000000000000000 00000000000000000
```

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

```
sim:/datapath/RF/Data @ 724 ps
0 : 00000000000001010 10000000000000001
2 : 10000000000000000 00000000000000001
4 : 00000000000000000 00000000000000000
6 : 00000000000000000 00000000000000000
```

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



LLI

0000000001...

sim:/datapath/RF/Data @ 634 ps

0 : 00000000000001010 100000000000000001

2 : 00000000010101010 000000000000000010

4 : 00000000000000000 000000000000000000

6 : 00000000000000000 000000000000000000

Loads the immediate value into Register R2



LW and LM

```
sim:/datapath/RF/Data @ 1037 ps
0 : 000000000000000100 000000000000000001
2 : 000000000000000010 000000000000000000
4 : 000000000000000000 000000000000000000
6 : 0101000001010100 000000000000000000
```

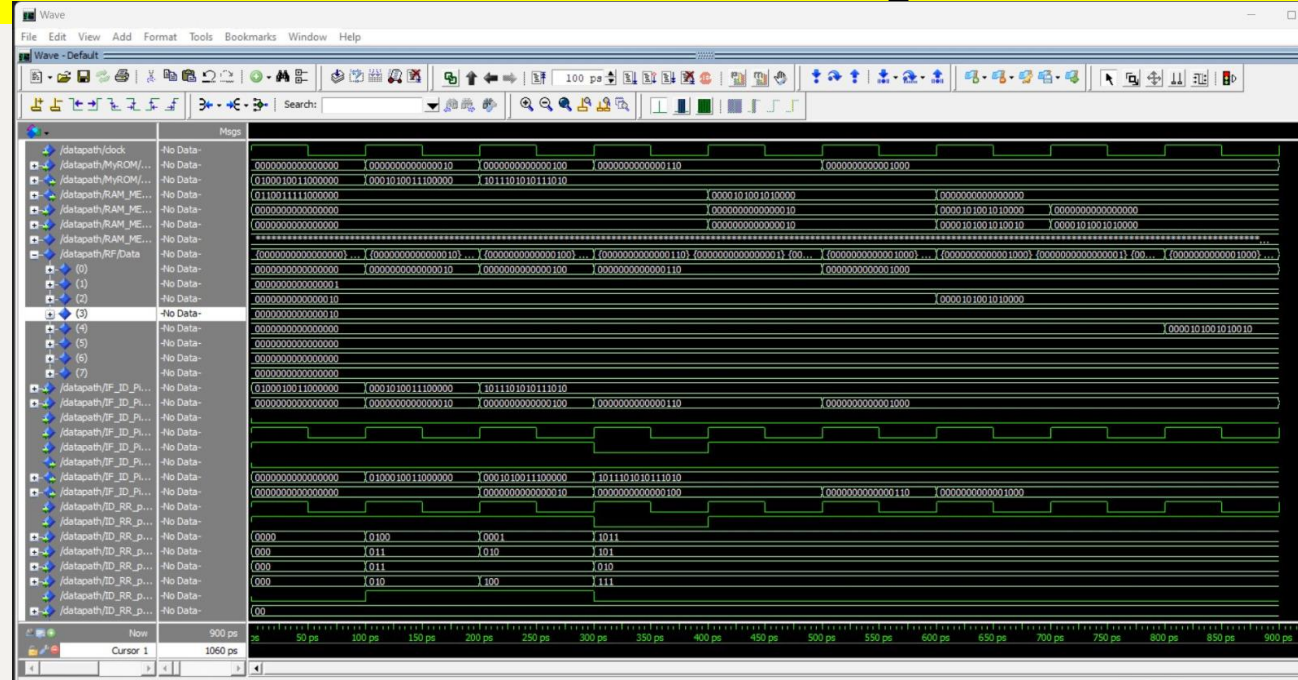
Loads the data from $R1+1$ address
in $R6$ and from $R1+4$ address in
 $R3$

```
sim:/datapath/RF/Data @ 656 ps
0 : 00000000000001010 01010100110000010
2 : 0000000000000010 00000000000000010
4 : 0000000000000000 00000000000000000
6 : 0000000000000000 00000000000000000
```

Loads the data from
 $R2+2$ address to $R1$



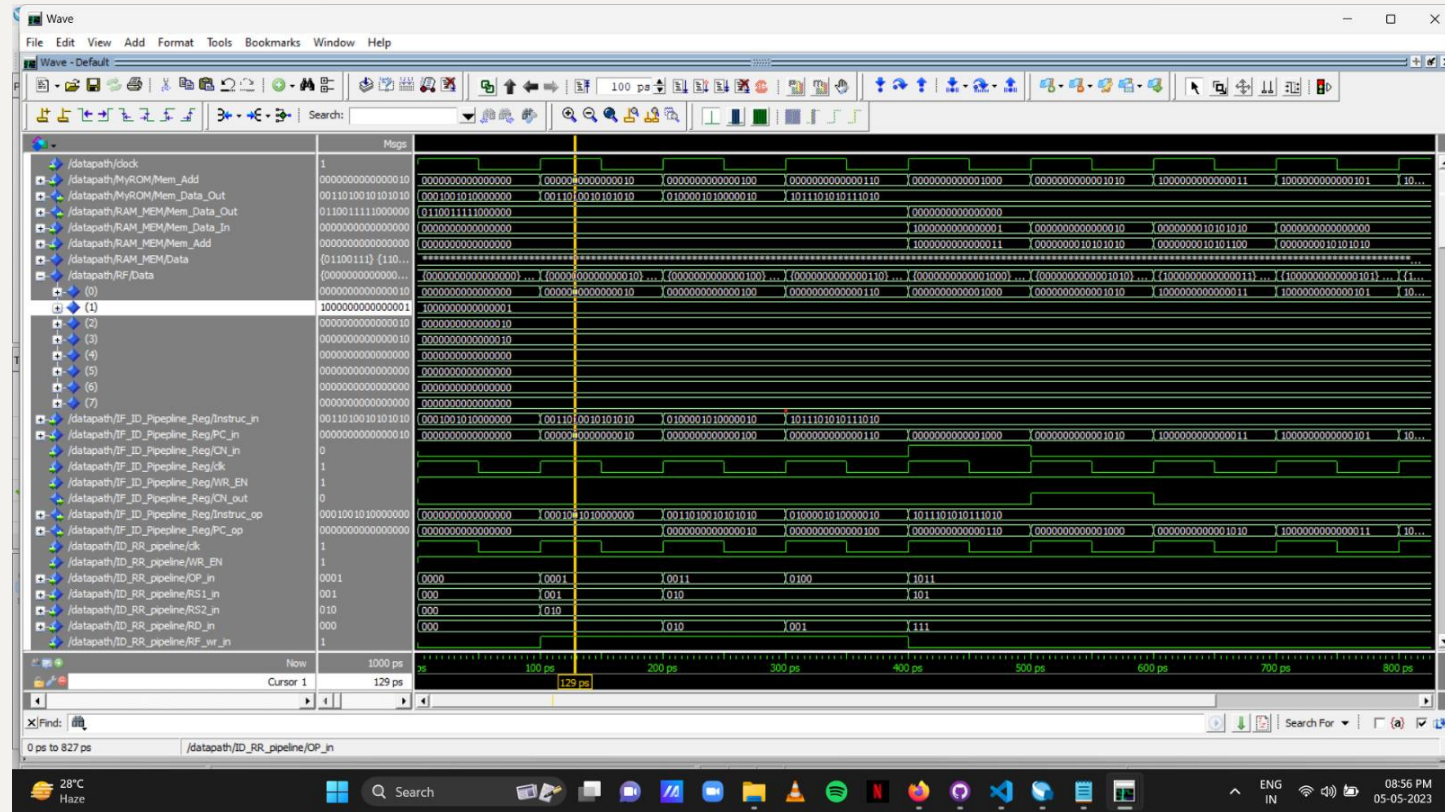
LW with Immediate Dependencies



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



JAL followed by ADI

```
sim:/datapath/RF/Data @ 812 ps
0 : 0000000000000100 0000000000000001
2 : 0000000000000001 0000000000000111
4 : 0000000000000000 0000000000000010
6 : 0000000000000000 0000000000000000
```

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



JLR

```
sim:/datapath/RF/Data @ 631 ps
0 : 00000000000000110 00000000000000010
2 : 00000000000000010 00000000000000010
4 : 00000000000000000 00000000000000000
6 : 00000000000000000 00000000000000000
```

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

```
sim:/datapath/RAM_MEM/Data @ 981579 ps
0 : 00000000 00000001 00000000 00000001 00000000 00000010 00000000 00000011
8 : 00000000 00000101 00000000 00001000 00000000 00001101 00000000 00010101
16 : 00000000 00100010 00000000 00110111 00000000 01011001 00000000 10010000
24 : 00000000 11101001 00000001 01111001 00000010 01100010 00000011 11011011
32 : 00000110 00111101 00001010 00011000 00010000 01010101 00011010 01101101
40 : 00101010 11000010 01000101 00101111 01101111 11110001 10110101 00100000
48 : 00100101 00010001 11011010 00110001 11111111 01000010 11011001 01110011
56 : 11011000 10110101 10110010 00101000 10001010 11011101 00111101 00000101
64 : 11000111 11100010 00000100 11100111 11001100 11001001 11010001 10110000
72 : 10011110 01111001 01110000 00101001 00001110 10100010 01111110 11001011
80 : 10001101 01101101 00001100 00111000 10011001 10100101 10100101 11011101
88 : 00111111 10000010 11100101 01011111 00100100 11100001 00001010 01000000
96 : 00101111 00100001 00111001 01100001 01101000 10000010 10100001 11100011
104 : 00001010 01100101 10101100 01001000 10110110 10101101 01100010 11110101
112 : 00011001 10100010 01111100 10010111 10010110 00111001 00010010 11010000
120 : 10101001 00001001 10111011 11011001 01100100 11100010 00100000 10111011
128 : 10000101 10011101 10100110 01011000 00101011 11110101 11010010 01001101
136 : 11111110 01000010 11010000 10001111 11001110 11010001 10011111 01100000
144 : 01101110 00110001 00001101 10010001 01111011 11000010 10001001 01010011
152 : 00000101 00010101 10001110 01101000 10010011 01111101 00100001 11100101
160 : 10110101 01100010 11010111 01000111 10001100 10101001 01100011 11110000
168 : 11110000 10011001 01010100 10001001 01000101 00100010 10011001 10101011
176 : 11011110 11001101 01111000 01111000 01010111 01000101 11001111 10111101
184 : 00100111 00000010 11110110 10111111 00011101 11000001 00010100 10000000
192 : 00110010 01000001 01000110 11000001 01111001 00000010 10111111 11000011
200 : 00111000 11000101 11111000 10001000 00110001 01001101 00101001 11010101
208 : ...
.
.
.
```

