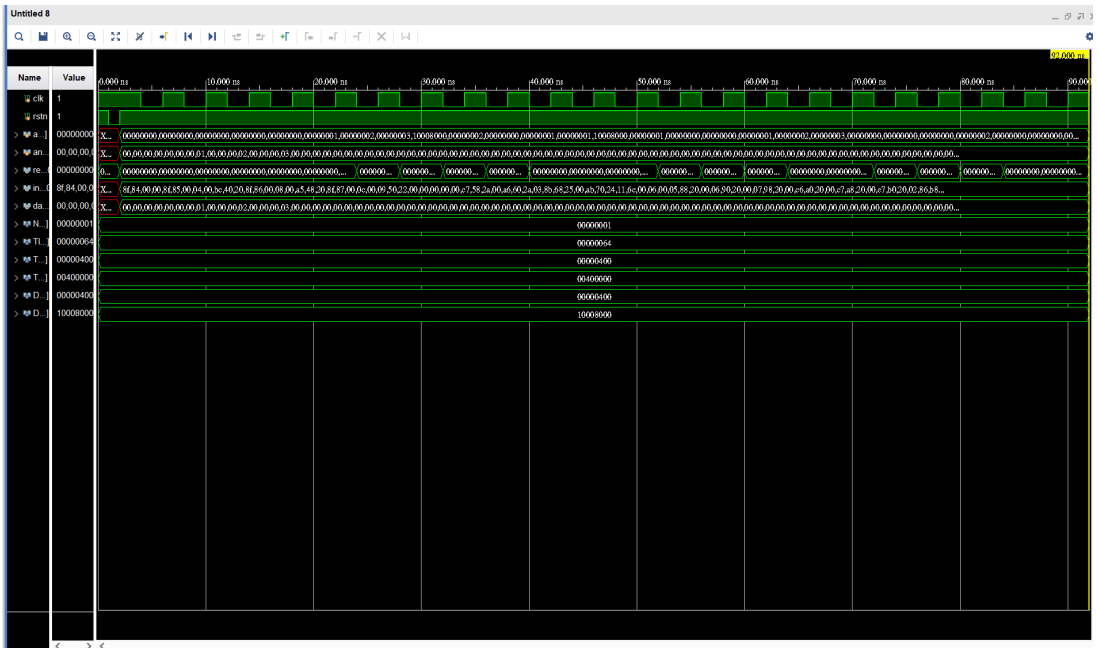
# Lab3: Pipelined Processor - Part 1

1. Experimental Result
    a. Show the waveform screen shot of the test we provided.
        - Here is the waveform of the 0.s file that TA provided.

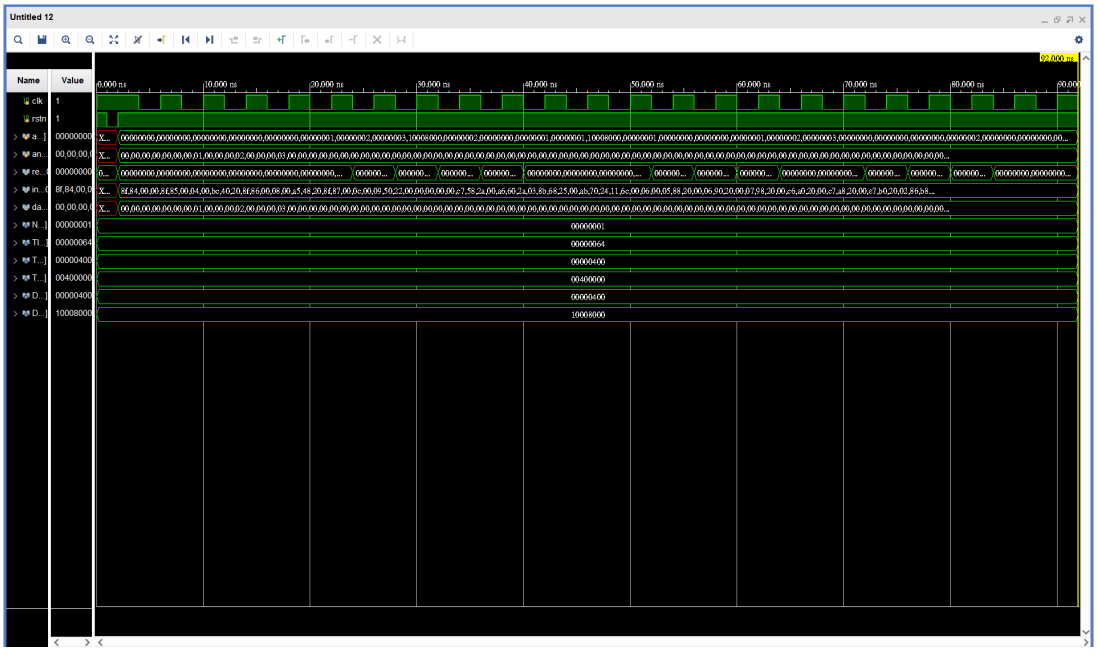



    b. What other cases you've tested? Why you choose them?
        - Here is the 1.s file that we create. We choose them to test the function of the design of my pipelined.v .

testbench > ASM 1.s

```
1           .data   0x10008000          # Start of Dynamic Data (pointed by $gp)
2           .word   0x0                 # 0($gp) - Changed initial value to 4
3           .word   0x1                 # 4($gp) - Changed initial value to 3
4           .word   0x2                 # 8($gp) - Unchanged
5           .word   0x3                 # 12($gp) - Changed initial value to 1
6
7           .text   0x00400000          # Start of Text (pointed by PC)
8   main:   lw      $a0, 0($gp)         # Load word from address pointed by $gp into $a0
9           lw      $a1, 4($gp)         # Load word from address $gp+4 into $a1
10          sub     $t0, $a0, $a1       # $t0 = $a0 - $a1
11          lw      $a2, 8($gp)         # Load word from address $gp+8 into $a2
12          add     $t1, $a1, $a2       # $t1 = $a1 + $a2
13          lw      $a3, 12($gp)        # Load word from address $gp+12 into $a3
14          mul     $t2, $a2, $a3       # $t2 = $a2 * $a3
15          nop                         # No operation (pipeline delay slot)
16          slt     $t3, $a0, $a1       # Set $t3 = 1 if $a0 < $a1
17          slt     $t4, $a2, $a3       # Set $t4 = 1 if $a2 < $a3
18          xor     $t5, $t3, $t4       # $t5 = $t3 XOR $t4
19          or      $t6, $a0, $a1       # $t6 = $a0 OR $a1
20          beq     $t5, $zero, end     # Branch to end if $t5 equals 0
21          add     $s1, $a1, $a3       # Set $s1 = $a1 + $a3
22          add     $s2, $a0, $a2       # Set $s2 = $a0 + $a2
23          add     $s3, $t0, $t1       # Set $s3 = $t0 + $t1
24          add     $s4, $t1, $t2       # Set $s4 = $t1 + $t2
25          add     $s5, $t2, $t3       # Set $s5 = $t2 + $t3
26          add     $s6, $t4, $t5       # Set $s6 = $t4 + $t5
27  end:    add     $s7, $s4, $s5       # Set $s7 = $s4 + $s5
```

- Here is the waveform of the 1.s testbench.




2. Answer the following Questions

   a. For each code sequence below, state whether it must stall, can avoid stalls using only forwarding, or can execute without stalling or forwarding.

| Sequence 1 | Sequence 2 | Sequence 3 |
|---|---|---|
| lw $t0,0($t0)<br>add $t1,$t0,$t0 | add $t1,$t0,$t0<br>addi $t2,$t0,#5<br>addi $t4,$t1,#5 | addi $t1,$t0,#1<br>addi $t2,$t0,#2<br>addi $t3,$t0,#2<br>addi $t3,$t0,#4<br>addi $t5,$t0,#5 |

In Sequence 1: Must stall.

In Sequence 2: Can avoid stalls using only forwarding.

In Sequence 3: Can execute without stalling or forwarding.

b. Explain the difference between throughput and latency.

- **Focus**: Latency focuses on the delay for a single task, while throughput focuses on the volume of tasks or data handled in a timeframe.
- **Optimization**: Reducing latency means decreasing the time to complete a task, often focusing on speed per task. Increasing throughput aims at enhancing the system's capacity to handle more tasks or data simultaneously.
- **Trade-off**: Improving throughput might increase latency as systems deal with more tasks at once, potentially slowing individual task completion. Reducing latency might decrease throughput if resources are dedicated to speeding up individual tasks rather than handling multiple tasks efficiently.

c. A group of students were debating the efficiency of the five-stage pipeline when one student pointed out that not all instructions are active in every stage of the pipeline. After deciding to ignore the effects of hazards, they made the following five statements. Which ones are correct？Explain why or why not.

a. Allowing jumps, branches, and ALU instructions to take fewer stages than the

b. Trying to allow some instructions to take fewer cycles does not help, since the throughput is determined by the clock cycle; the number of pipe stages per instruction affects latency, not throughput.

c. You cannot make ALU instructions take fewer cycles because of the writeback of the result, but branches and jumps can take fewer cycles, so there is some opportunity for improvement.

d. Instead of trying to make instructions take fewer cycles, we should explore making the pipeline longer, so that instructions take more cycles, but the cycles are shorter. This could improve performance.

A: The forth statement. Since making the pipeline longer (increasing the number of stages) can allow the clock cycle to be shortened (higher clock frequency), which may improve performance in terms of throughput. However, this also introduces more complexity in managing hazards and may increase the latency per instruction.

3. Problem Encountered & Solution

   A: The problem i encountered in this lab is the data transport from one state to the other state. In the beginnig, i was confused about the pipelined register and i spent lot of time to figure out and practice the diagram of the figure 4.51. But it is a treasured experience to understand the step and the verilog code in pipelined CPU.

4. Feedback

   However, I think the hints and the spec can express more clearly in this lab, the concept to do this lab is good.