1 Activity 1 - Forwarding, Revisited

Simulate the following sequence of instructions. Do NOT insert any stalls. Each time that a register is used, determine:

• Were the values read in ID correct?
• If not, what pipeline register currently contains the not-yet-written values?

Draw a pipeline diagram, showing the dependencies:

```
lui $t0, 0xdead # la $t0, LABEL (part 1)
ori $t0, $t0, 0xbee4 # la $t0, LABEL (part 2)

lw $t1, 0($t0)
addi $t1, $t1,1
sw $t1, 0($t0)

lw $t1, 4($t0)
sw $t1, 8($t0)

lw $t1, 12($t0)
nop
sw $t1, 16($t0)
```
2 Activity 2 - Simulating a Stall

Consider a pipeline, which is full (that is, each phase holds an ongoing instruction). Let’s call these instructions

- $x+4$ - The instruction in IF
- $x$ - The instruction in ID
- $x-4$ - The instruction in EX
- $x-8$ - The instruction in MEM
- $x-12$ - The instruction in WB

Suppose that you discover that the instruction in the ID phase needs to stall (that is, it must pause). The instruction IF must also pause.

First, describe the contents of all 4 pipeline registers as they are at the beginning of this clock cycle. Also describe what the PC currently contains during this clock cycle.

Next, describe what each pipeline phase will be doing - pay attention not only to what they do (“decode the instruction”) but also to which instruction they are working on.

Next, describe what you should store in the pipeline registers, at the end of this clock cycle.

Finally, describe what each pipeline phase will be doing during the next clock cycle. Again, pay close attention to which instruction is in each phase.