**Problem 1:**  The code below executes on the implementation illustrated.

(a) Draw a pipeline execution diagram up until the first fetch of the third iteration.

(b) What is the CPI for a large number of iterations?

```assembly
addi r3, $0, 123
LOOP:
lw r1, 0(r2)
bne r1, r3, LOOP
lw r2, 4(r1)
```

**Problem 2:**  Is there any way to add bypass paths to the implementation above so that the code executes with fewer stalls:

(a) Suggest bypass paths that might have critical path impact but which probably won’t halve the clock frequency.

(b) Explain why it is impossible to remove all stalls by adding bypass paths.
Problem 3: The `beqir` instruction from the midterm exam solution compares the contents of the `rs` register to the immediate, if the two are equal the branch is taken, the address of the branch target is in the `rt` register. In the code example below `beqir` compares the contents of `r3` to the constant 123, if they are equal the branch is taken with register `r1` holding the target address, in this case to TARG. The delay slot, `nop`, is also executed.

(a) Show the changes needed to implement this instruction on the implementation above.

(b) Include bypass paths so that the code below executes as fast as possible:

```assembly
lui r1, hi(TARG)
ori r1, r1, lo(TARG)
beqir r3, 123, r1
nop

# Lots more code.
TARG:
xor r9, r10, r11
```