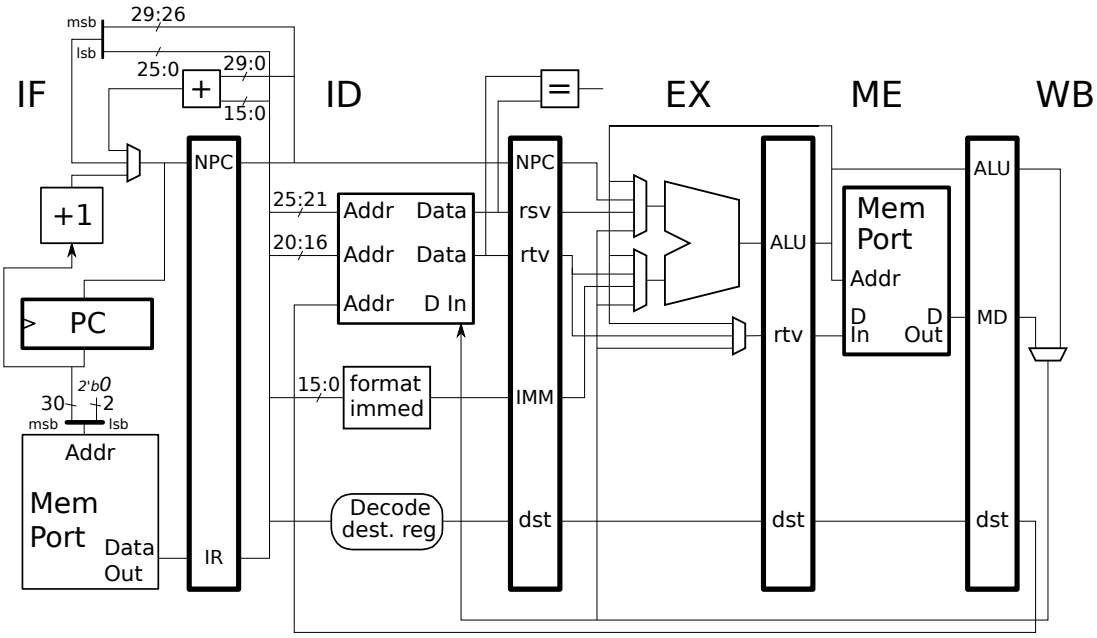
Problem 1: Appearing below are **incorrect** executions on the illustrated implementation. For each execution explain why it is wrong and show the correct execution. *Note: This problem was assigned in 2020, 2021, and 2022, and their solutions are available. DO NOT look at the solutions unless you are lost and can't get help elsewhere. Even in that case just glimpse.*

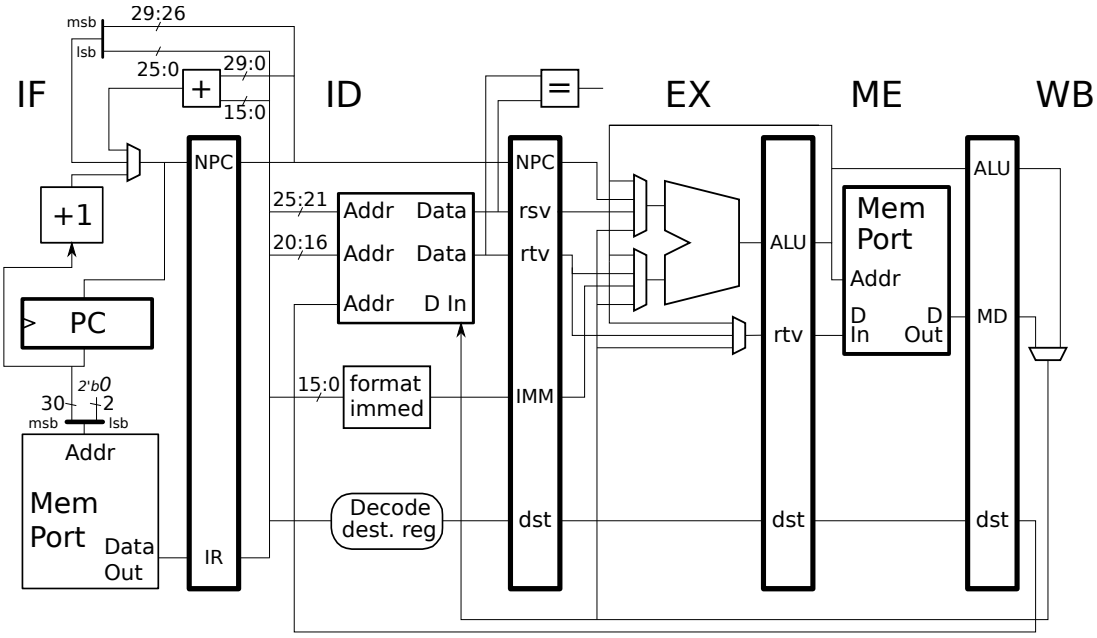


(*a*) Explain error and show correct execution.

```
# Cycle           0  1  2  3  4  5  6  7
add r1, r2, r3    IF ID EX ME WB
xor r4, r1, r5       IF ID -> EX ME WB
```

(*b*) The execution of the branch below has **two** errors. One error is due to improper handling of the `andi` instruction. (That is, if the `andi` were replaced with a `nop` there would be no problem in the execution below.) The other is due to the way the `beq` executes. As in all code fragments in this problem, the program is correct, the only problem is with the illustrated execution timing.

```
# Cycle:          0  1  2  3  4  5  6  7  8
andi r2, r2, 0xff IF ID EX ME WB
beq r1, r2, TARG     IF ID EX ME WB
add r3, r4, r5          IF ID EX ME WB
xor                        IFx
TARG:
sw r6, 7(r8)                  IF ID EX ME WB
# Cycle:          0  1  2  3  4  5  6  7  8
```



(*c*) Explain error and show correct execution.

```
# Cycle          0  1  2  3  4  5  6  7
lw r2, 0(r4)     IF ID EX ME WB
add r1, r2, r7      IF ID EX ME WB
```

(*d*) Explain error and show correct execution.

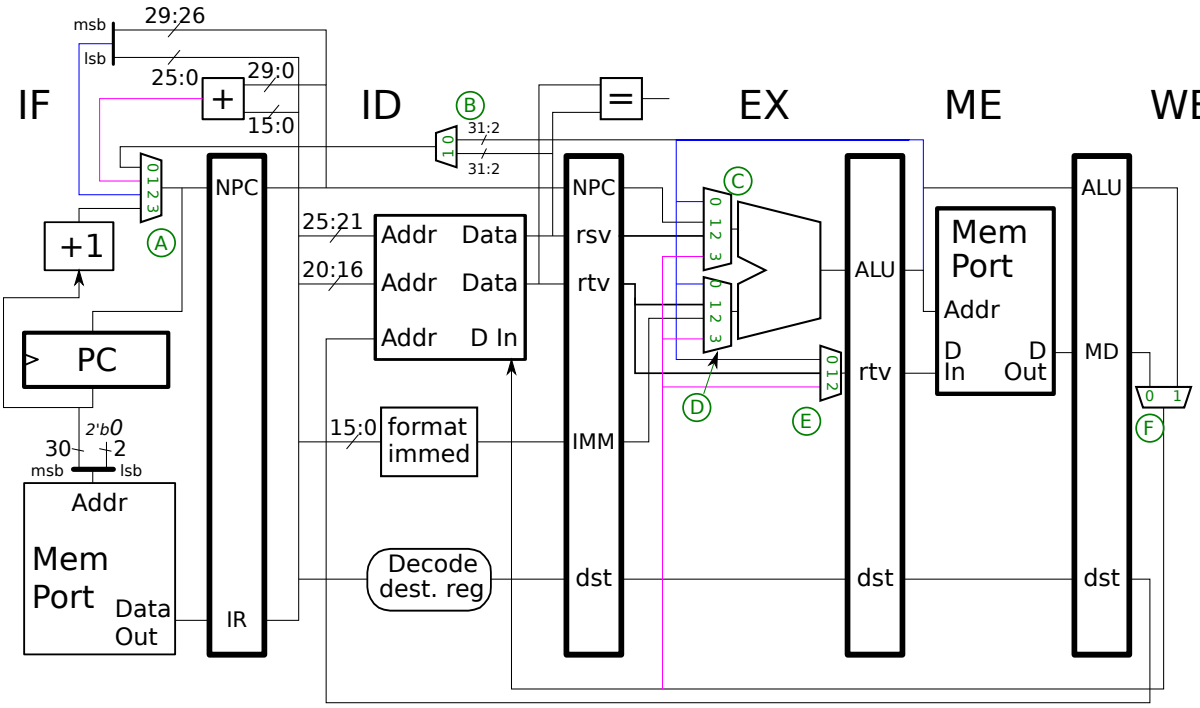
```
# Cycle          0  1  2  3  4  5  6  7
add r1, r2, r3   IF ID EX ME WB
lw r1, 0(r4)        IF ID -> EX ME WB
```

(*e*) Explain error and show correct execution.

```
# Cycle          0  1  2  3  4  5  6  7
add r1, r2, r3   IF ID EX ME WB
sw r1, 0(r4)        IF ID -> EX ME WB
```

Problem 2: Illustrated below is a MIPS implementation in which each multiplexor has a label, such as a circled A at the multiplexor providing a value for the PC. (The implementation debuted on the 2018 midterm exam.) The multiplexor inputs are also numbered. Below the illustration an execution of the program on the implementation is shown for two iterations of a loop. Below the execution is a table with one row for each labeled multiplexor. Complete the table so that it shows the values on the multiplexors' select signals at each cycle based on the execution. Leave an entry blank if its value does not make a difference.

Wire thicknesses and colors have been varied to make it easier to trace them through the diagram. *Before attempting this problem, solve 2018 Midterm Exam Problem 2b, which also appeared as 2022 Homework 3 Problem 2. Also see the 2014 Midterm Exam Problem 1 for a similar problem.*



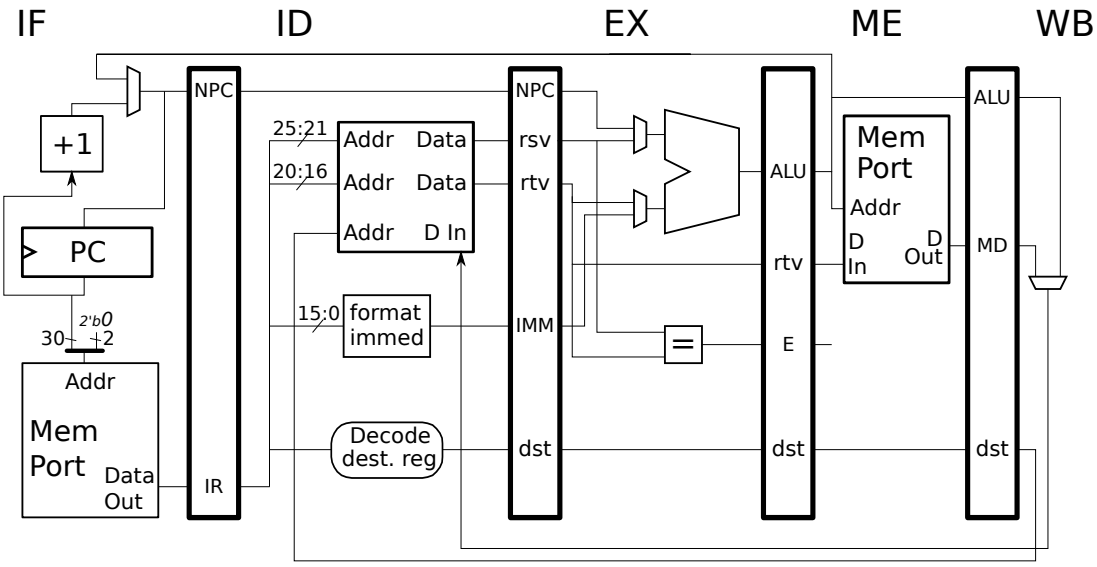
Continued on the next page.

- [ ] Complete the table (the rows starting with `A:`, `B:`, etc.) based on the execution below.
- [ ] Omit select signal values if they do not matter. For example, omit values for `E` for cycles in which there is not a store instruction in `EX`.
- [ ] Assume that the branch is taken the second time it appears. (No assumption needed for its first appearance.)

```
 addi r1, r1, -4   IF    ID    EX    ME    WB
LOOP: # Cycle      0     1     2     3     4     5     6     7     8     9
 sw r2, 4(r1)            IF    ID    EX    ME    WB
 lw r1, 8(r2)                  IF    ID    EX    ME    WB
 bne r2, r3, LOOP                    IF    ID    EX    ME    WB
 add r2, r2, r6                            IF    ID    EX    ME    WB
LOOP: # Cycle      0     1     2     3     4     5     6     7     8     9
 sw r2, 4(r1)                              IF    ID    EX    ME    WB
 lw r1, 8(r2)                                    IF    ID    EX    ME    WB
 bne r2, r3, LOOP                                      IF    ID    EX    ME    WB
 add r2, r2, r6                                              IF    ID    EX    ME    WB
     # Cycle      0     1     2     3     4     5     6     7     8     9     10    11    12

 A:

 B:

     # Cycle      0     1     2     3     4     5     6     7     8     9     10    11    12

 C:

 D:

     # Cycle      0     1     2     3     4     5     6     7     8     9     10    11    12

 E:

 F:

     # Cycle      0     1     2     3     4     5     6     7     8     9     10    11    12
```

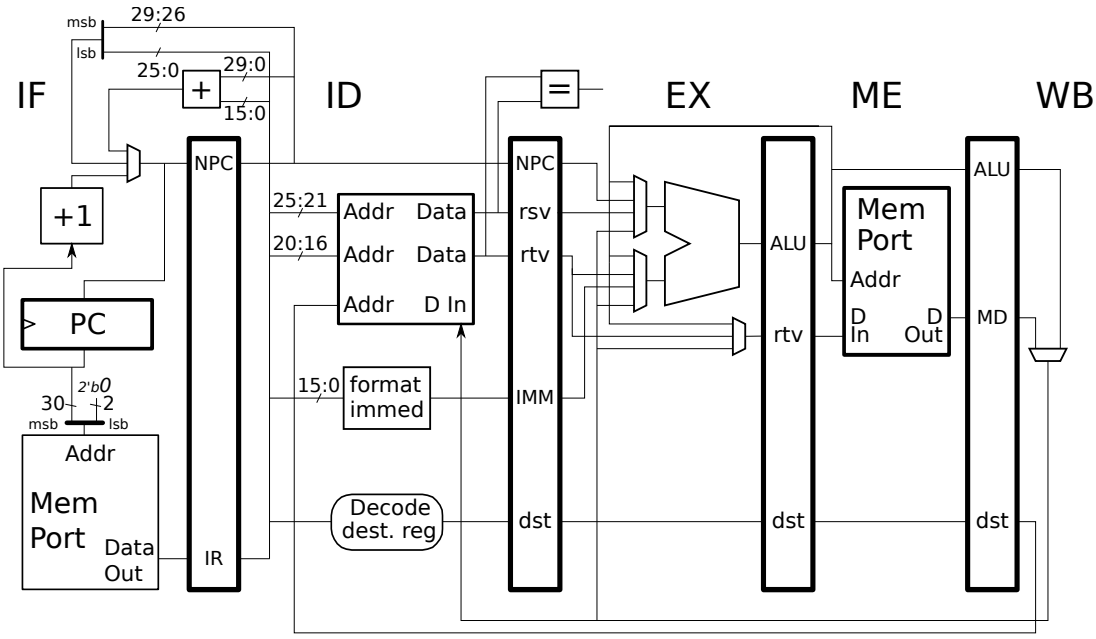
Problem 3: Show the execution of the code fragments on the following implementations for enough iterations to determine the instruction throughput (IPC). **As always, base the behavior of branches and the availability of bypasses on the implementations. Also, don't forget that MIPS branches have a delay slot.** Sorry for yelling, but I hate it when students miss things.

This problem appeared as most of Problem 1 on the 2022 Final Exam. A solution is not yet available.



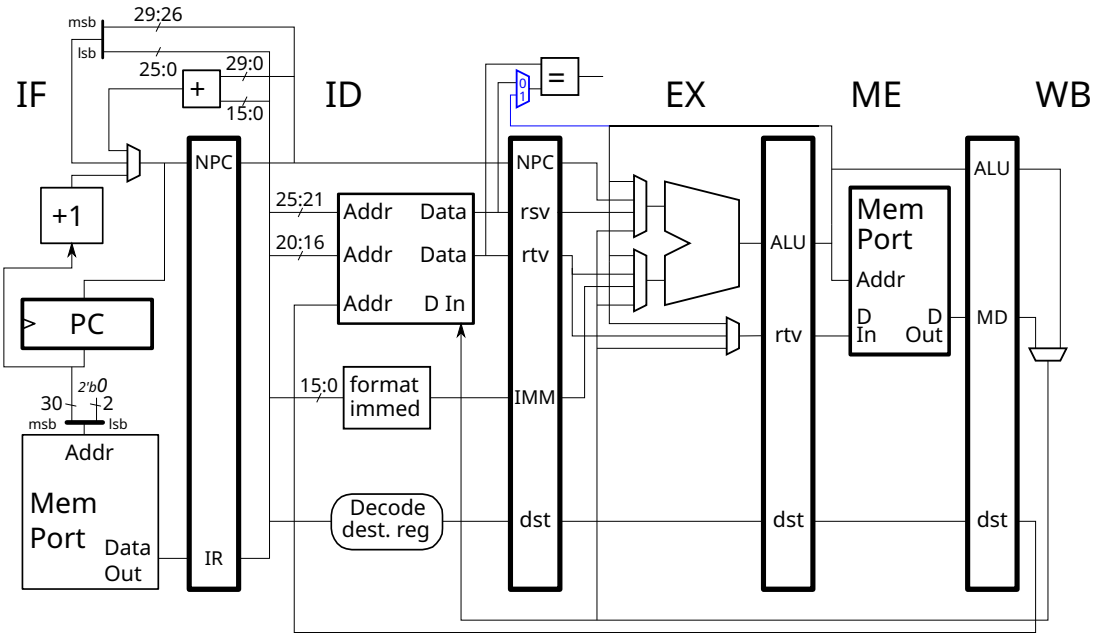
☐ Show execution and ☐ determine instruction throughput (IPC) based on a large number of iterations.

```
LOOP:
 bne r1, r2, LOOP

 addi r1, r1, 4

 xor r5, r6, r7

 sub r8, r9, r10
```



☐ Show execution and ☐ determine instruction throughput (IPC) based on a large number of iterations.

```
LOOP:
 bne r1, r2, LOOP

 addi r1, r1, 4

 xor r5, r6, r7

 sub r8, r9, r10
```



☐ Show execution and ☐ determine instruction throughput (IPC) based on a large number of iterations.

```
LOOP:
 bne r1, r2, LOOP

 addi r1, r1, 4

 xor r5, r6, r7

 sub r8, r9, r10
```