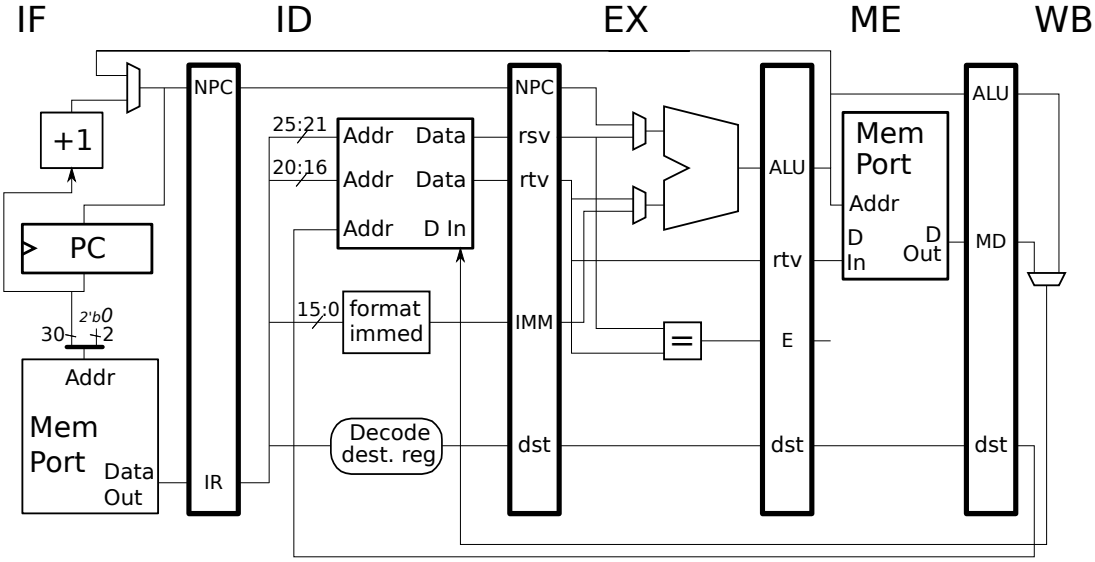
LSU EE 4720 **Homework 2** **Due: 13 February 2019**

The solution to several of the problems in this assignment requires material about to be covered in class, in particular, stalling instructions to avoid hazards. For coverage of this material see slide set six, `https://www.ece.lsu.edu/ee4720/2019/lsli06.pdf`*. For a solved problem see 2014 Homework 1 Problem 3. Feel free to look through old homework and exams for other similar problems, but when doing so make sure that the MIPS implementation matches the one in this problem: the muxen at the ALU inputs should each have just 2 inputs.*

Problem 1: *Note: The following problem was assigned in each of the last three years (though not in color), and its solution is available. DO NOT look at the solution unless you are lost and can't get help elsewhere. Even in that case just glimpse.* Appearing below are **incorrect** executions on the illustrated implementation. For each one explain why it is wrong and show the correct execution.



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

```
LOOP: # Cycles     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
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

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

```
LOOP: # Cycles     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
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

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

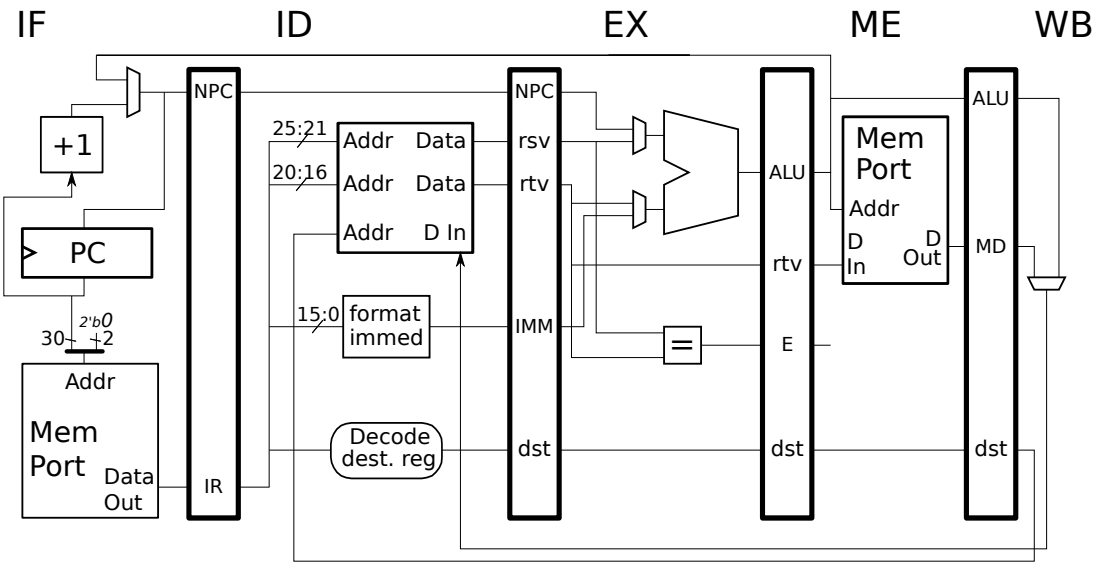
```
LOOP: # Cycles     0  1  2  3  4  5  6  7
 add r1, r2, r3    IF ID EX ME WB
 sw r1, 0(r4)         IF ID -> EX ME WB
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

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

```
LOOP: # Cycles     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
LOOP: # Cycles     0  1  2  3  4  5  6  7
```

Problem 2: The MIPS code below is taken from the solution to 2018 Homework 1. Show the execution of this MIPS code on the illustrated implementation for two iterations. The register file is designed so that if the same register is simultaneously written and read, the value that will be read will be value being written. (In class we called such a register file *internally bypassed*.)

- Check carefully for dependencies.
- Focus on when the branch target is fetched and on when wrong-path instructions are squashed.
- Be sure to stall when necessary.




```
CLOOP:
 lbu $t0, 0($t4)
 sb $t0, 0($a1)
 addi $t4, $t4, 1
 bne $t4, $t5, CLOOP
 addi $a1, $a1, 1
```