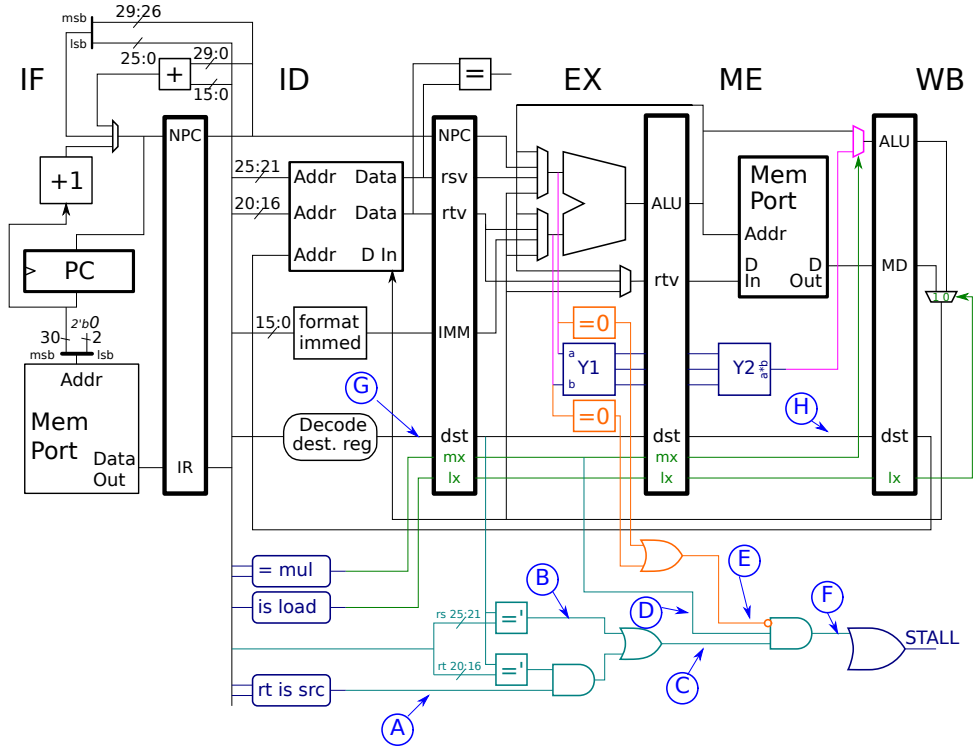
LSU EE 4720 **Homework 3** **Due: 30 March 2020**

Please E-mail solutions of this assignment to koppel@ece.lsu.edu by the evening of the due date. PDF files are preferred. These can be generated by scanning software that you might have installed with a multifunction printer. A PDF can also be assembled from photos of a hand-completed copy. The disorganized homework penalty will be ignored for the remainder of the semester (unless we return early) so an E-mail with multiple image attachments will be accepted without penalty. **Do not** *physically mail them to my office address, I will not be able to pick them up.*

Problem 1: Appearing below is the solution to Homework 2 with labels added to some wires, which is followed by an execution of the code showing values on those labeled wires. The execution is based on the code fragment shown plus `nop` instructions before the first instruction (`addi`) and after the last instructions (`nop`).




```
# Cycle          0     1     2     3     4     5     6
addi R2, r0, 0   IF    ID    EX    ME    WB
mul R1, R2, r3         IF    ID    EX    ME    WB
add r4, R2, R1               IF    ID    EX    ME    WB
# Cycle          0     1     2     3     4     5     6
A                      0     1     1
B                      0     1     0
C                      0     1     1
# Cycle          0     1     2     3     4     5     6
D                            0     1     0
E                            1     1     1
F                      0     0     0     0
# Cycle          0     1     2     3     4     5     6
G                      2     1     4
H                                  2     1     4
# Cycle          0     1     2     3     4     5     6
```

(*a*) Refer to the table on the previous page for this problem. Notice that the value in the `B` row (above) in cycle 1 is `0`. According to the problem statement the instruction before `addi` is a `nop`.

Why would that value be `0` regardless of what instruction came before `addi`?

Suppose the `addi r2, r0, 0` were changed to `addi r2, r7, 0`. Why would the value in the `B` row still be `0`?

(*b*) Appearing below is a different code fragment. Complete the table so that it shows the values on the labeled wires.

```
# Cycle          0     1     2     3     4     5     6     7
ori r2, r6, 7    IF    ID    EX    ME    WB
sub r1, r2, r2         IF    ID    EX    ME    WB
mul r3, r8, r1               IF    ID    EX    ME    WB
mul r5, r3, r4                     IF    ID    EX    ME    WB
```

```
# Cycle          0     1     2     3     4     5     6     7
A

B

C
# Cycle          0     1     2     3     4     5     6     7
D

E

F
# Cycle          0     1     2     3     4     5     6     7
G

H
# Cycle          0     1     2     3     4     5     6     7
```

(*c*) Appearing below are completed tables, but without a code fragment. Show a code fragment that could have produced those table values.

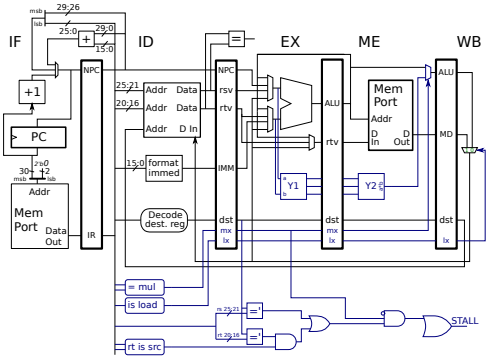
The originally assigned problem contained an error which made it difficult to solve. Shown below is the originally table, followed by the intended table. The solution uses the intended table.

```
# As originally assigned. Contains an error in F at cycle 3.
# Cycle          0     1     2     3     4     5     6     7
A                      0     1           0     1
B                      0     1           1     0
C                      0     1           1     1
# Cycle          0     1     2     3     4     5     6     7
D                            0     1           0     0
E                            0     0           0     0
F                            0     0           0     0
# Cycle          0     1     2     3     4     5     6     7
G                      2     3           4     8
H                                  2     3           4     8
# Cycle          0     1     2     3     4     5     6     7


# Intended problem.
# Cycle          0     1     2     3     4     5     6     7
A                      0     1           0     1
B                      0     1           1     0
C                      0     1           1     1
# Cycle          0     1     2     3     4     5     6     7
D                            0     1     0     0     0
E                            0     0           0     0
F                            0     1     0     0     0
# Cycle          0     1     2     3     4     5     6     7
G                      2     3           4     8
H                                  2     3           4     8
# Cycle          0     1     2     3     4     5     6     7
```

Problem 2: Appearing below and on the next page is the solution to Homework 2 Problem 1. In this problem add hardware to handle a different and less special multiplication special case. Suppose that the middle output of the `Y1` stage of the multiplier held the correct product whenever the high 24 bits of its `b` input are zero. For example, when `b` is 1, 5, or 255. Call such values *small*. In all cases the correct product appears at the output of `Y2`.

Note: All outputs of `Y1` arrive with zero slack, even the center output with the small `b` special case. That means that nothing can be done with these values until the next clock cycle, at least without reducing the clock frequency.



(*a*) Add hardware to bypass the product to the ALU and to the `rtv` mux when `b` is small. (There is a larger diagram on the next page.) The bypass should allow the first code fragment below to execute without a stall.

(*b*) Add control logic to suppress the stall when it is possible to bypass.

In the first code fragment below the stall is avoided because the `b` value (which is the `rtv`) is small, in the second it is too large.

```
# Cycle          0  1  2  3  4  5  6  7
addi r1, r0, 23  IF ID EX ME WB
mul r2, r3, r1      IF ID EX ME WB
sub r4, r2, r5         IF ID EX ME WB
```

```
# Cycle          0  1  2  3  4  5  6  7
addi r1, r0, 300 IF ID EX ME WB
mul r2, r3, r1      IF ID EX ME WB
sub r4, r2, r5         IF ID -> EX ME WB
```

- Make sure that the changes don't break existing instructions.
- As always avoid costly solutions.
- As always pay attention to critical path.

The SVG source for the illustration below is at `https://www.ece.lsu.edu/ee4720/2020/hw03-p2.svg`. It can be edited using Inkscape or any other SVG editor, or (not recommended) a text editor.