CS2100 Computer Organization
Tutorial 10: Pipelining
(Week 12: 6 Nov – 10 Nov 2023)

1. Let's finish what we started… design a Boolean circuit that will decode the following control bits for the MIPS **`add, and, beq, lw, sw, nor, or, slt, sub`** instructions. You may use AND, OR, NAND, NOR, XOR gates with up to 4 inputs and 'NOT' must be explicitly performed. Also the opcode should be labelled $op_5op_4op_3op_2op_1op_0$ – where op_i is the i-th bit.

    a) RegDst
    b) MemtoReg
    c) MemWrite
    d) ALUOp
    e) ALU Control

2. Using Powerpoint animation (or just a number of slides), give the full details of the signals and data bits (highlighting those lines that are active) of the following 3 instructions are executed together in a pipelined MIPS (tut10.ppt, slide 2):

    a) `sub $1, $2, $3`
    b) `lw $6, 4($7)`
    c) `beq $4, $5, L2`

    Assume that L2 at PC=0x100 instruction while the beq is at PC=0x1000.

3. Consider the following sequence of MIPS instructions:

```
add $1, $2, $3 # PC = 0x100
add $1, $1, $3 # PC = 0x104
add $1, $1, $1 # PC = 0x108
```

    In the datapath with forwarding (tut10.ppt, slide 3 – and you may need to refer to slide 4 for more detailed marking of the lines, which unfortunately are messy and may not quite line up), trace the execution of these 3 instructions as they pass through the pipeline with attention paid especially to the forwarding of operands. Clearly identify which rule(s) of the forwarding (or hazard detection) was fired, if any.

4. Repeat Q3 for the following instruction sequence:

```
add $1, $2, $3 # PC = 0x100
lw  $1, 0($1)  # PC = 0x104
add $1, $1, $1 # PC = 0x108
add $3, $2, $1 # PC = 0x10c
```

5. Using the graphical notation for pipeline introduced in class, show how the following sequence of instructions would be exeucted in a datapath with forwarding, and branch resolution at the ID stage by

```
add $1, $2, $3 # PC = 0x100
lw  $1, 0($1)  # PC = 0x104
beq $1, $0, L  # PC = 0x108
add $3, $2, $1 # PC = 0x10c
```

(a) assuming that the **beq** turns out to be not taken.

(b) assuming that the **beq** turns out to be taken.