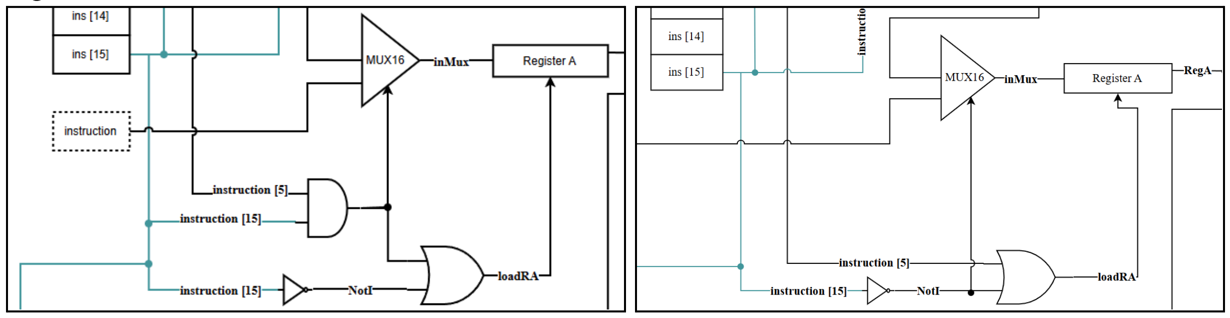
YOUR Group Details:

Group No:	10
Group Members (*Student IDs*):	Selina Yeoh Yun Ci (20620222) Tan Wei Sin (20619586) Wong Zi Xin (20603953) Wong Cheuk Kei (20602511)

Justification of YOUR Circuitry Diagram Design:

To reduce costs and power consumption, we performed logical optimisation on the decode section utilising tabular optimisation. In the original design, the opcode (Instruction[15]) AND d1 (Instruction[5]) determines whether register D or the 16-bit instruction should be used. If both bits are 1, it chooses the 16-bit instruction; otherwise, it selects the D register. If the bit was an A-instruction (Instruction[15]=0), it would always load the D register.

Figure 1.0: Before and After



Before After

Before optimisation, the original design had an extra AND gate. The opcode (Instruction[15]) and d1 (Instruction[5]) controlled whether the system used the ALU output or the 16-bit instruction. If both are 1, the inMux selects the 16-bit instruction; otherwise, it selects the ALU output. Register A will always load register A unless the opcode was a C-instruction and d1 was false (loadRA = 0), allowing Register A to retain its prior value.

After optimisation, the AND gate was deleted, but the design's behaviour remained unchanged. Even if the inMux selects the ALU output during C-instruction, the value in Register A will be the same as the original design. This is because, in the same scenario when the opcode had a C-instruction and d1 was false, Register A preserves its previous value. This simplification minimises redundancy without compromising functionality.

The rest of the CPU stays the same.

The second multiplexer (MUX) decides whether to select Register A or inM. The selected value will enter the ALU, along with the D Register, to perform arithmetic and logical operations. Instructions [6], [7], [8], [9], [10], and [11] will function as control bits for the ALU, representing zx, nx, zy, ny, f, and no. The ALU will output to outM, and if the output is 0, the zr flag is set to 1. If the output is negative, the ng flag is set to 1.

When the instruction is C (instruction[15] = 1) and d2 is (instruction[4] = 1), the ALU output is loaded into the D register. When d3 (Instruction[3]) is 1 and the instruction is in C, the WriteM signal instructs the memory to perform a write operation.

When the jump bits, [0], [1], and [2], are set to one, the PC is instructed to jump. (when instruction[0]<0, instruction[1]=0, or instruction[2]>0)

For ALU outputs less than 0, we check if both instruction[2] and ng are set to 1; for outputs equal to 0, we check if both instruction[1] and zr are set to 1. When the output is more than zero, we verify that it is not negative or zero, and if so, we proceed with the load by setting instruction[0] to 1. We then apply the or gate to the three conditions to see if any of them are true before evaluating whether it is a C-instruction by comparing it to an instruction[15]. If everything is correct, it will instruct the computer to load Register A. If not, the PC will increment by one.

References:

Balasanyan, S., Aghagulyan, M., Wuttke, H.-D., Henke Bachelor, K., & Systems, E. (n.d.). Digital Electronics. https://ec.europa.eu/programmes/erasmus-plus/project-result-content/120e4810-0d29-4397-9ad4-b4091c2e3d19/Digital%20Electronics.pdf

nand2tetris, Part 1 — fkfd.me. (2022). Fkfd.me. https://fkfd.me/projects/nand2tetris_1/