Problem 4. ALU design.

Your task for this problem is to design the ALU represented by the box in the above diagram. The ALU has two 64-bit inputs and three or four control inputs. (As we discussed on Monday evening, it is possible to use only 3.)

The ALU performs the following operations on the inputs:

- ADD — two’s complement add (2sC). It assumes the two inputs are in 2sC form. There are four flags that are set: carry (C), negative (N), zero (Z), and overflow (V). These are set assuming the inputs and the result is a 2sC number.
• **SUB** — 2sC subtract. Flags are set.
• **AND** — bitwise AND. Flags are set.
• **XOR** — bitwise exclusive OR. Flags are set.
• **SADD** — saturated 2sC add. Flags are set. In saturation arithmetic, a result that overflows or underflows is saturated to the range limit for that data type. If the result of an operation exceeds the allowable range, it saturates to the maximum value of the range. If a result is less than the allowable range, it saturates to the minimum value of the range. Say you're adding two 16 bit numbers on a 16 bit machine. If you add them and get a result that's greater than what you can represent with 16 bits in 2sC the result should just be the maximum possible value ($2^{15} - 1$). Consider the example in class when we thought of this as a “volume knob.” If you go past the maximum volume, instead of going back to the most negative number ($-2^{16}$ — wraparound), you stay at maximum volume. In the case of negative numbers the same idea applies thus $\text{MOST\_NEG\_NUMBER} - 1 = \text{MOST\_NEG\_NUMBER}$. Flags are set.
• **8-bit byte parallel operations** — your ALU should be able to perform any of the above operations by splitting the inputs into eight 8-bit slices and performing the specified operation on each of them in parallel). When parallel operations occur assume they set their own copies of the CNZV$_i$ ($i = 0, 1, ..., 7$) flags and that these flags are bitwise inclusive OR-ed into the CNZV flags.

There are thus 8 separate operations: byte parallel AND and XOR are the same as 64-bit AND and XOR.