ALU Design

Prof. Naga Kandasamy
ECE Department, Drexel University

This problem worth 10 points is due October 25, 2023, by 11:59 pm. Please submit original work.

You are asked to design the arithmetic and logic unit (ALU) of the Hack computer in HDL using the various built-in chips available under nand2tetris/tools/builtInChips. The Hack ALU computes a fixed set of functions $out = f_i(x, y)$ where x and y are the two 16-bit inputs, out is the 16-bit output, and f_i is an arithmetic or logical function selected from a fixed set of eighteen possible functions. The ALU is instructed to compute a specific function by setting six control bits appropriately.

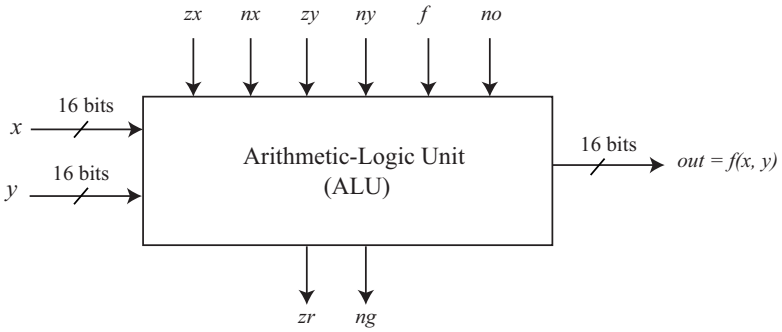

Fig. 1: Block diagram of the Hack ALU

Your ALU should support the following operations specified in the truth table.

zx	nx	zy	ny	f	no	out
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x + 1
1	1	0	1	1	1	y + 1
0	0	1	1	1	0	x - 1
1	1	0	0	1	0	y - 1
0	0	0	0	1	0	x + y
0	1	0	0	1	1	x - y
0	0	0	1	1	1	y - x
0	0	0	0	0	0	x & y
0	1	0	1	0	1	x | y

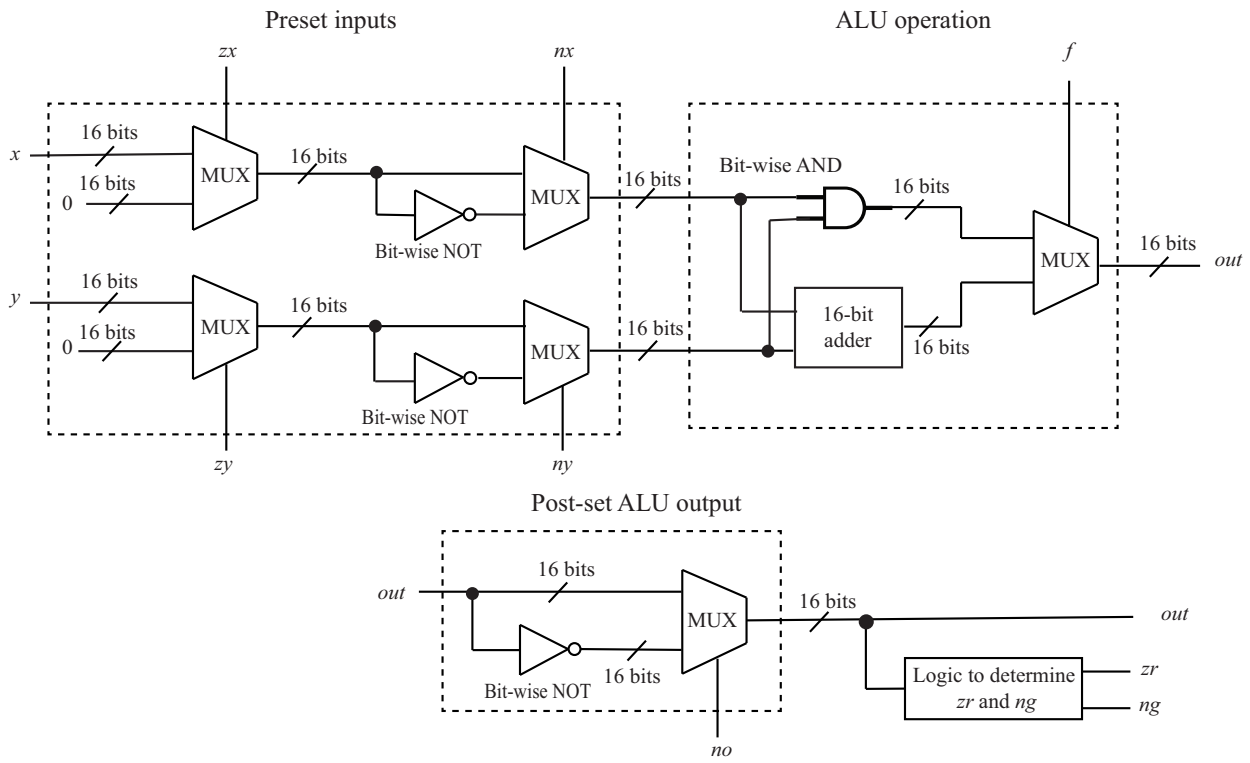

Fig. 2: Internal design of the Hack ALU.

The above schematic provides a starting point for your design. The basic idea is to use the various control inputs as select lines into multiplexers that guide the data flow through the overall system.

You have been provided with the `ALU.hdl` file as a starting point to develop your solution. Your chip design will be tested using the supplied `ALU.tst` file. When loaded into the hardware simulator, `ALU.tst` loads your HDL design, and supplies a battery of test inputs to it and appends the output responses as they are received in a text file called `ALU.out`. The contents of `ALU.out` must match the outputs listed in the supplied `ALU.cmp` file exactly, line-by-line. If not, the simulator will display an error, indicating the line number in the `.out` file at which the mismatch occurred.

Submit the completed `ALU.hdl` file via BBLearn. Do not submit any of the other files.

Note the following implementation tips:

- Though you have developed your own implementation of a 16-bit adder as part of the previous project, use the built-in `Add16` chip in your ALU design.
- The *zr* line is asserted if all 16 lines of the *out* bus, obtained after the post-set stage, are zeros. The nand2tetris toolkit provides a built-in `Or8Way` chip that can be used as a component to build this logic. It takes an 8-bit input and returns the corresponding bit-wise OR as output.[1]
- Finally to access a specific internal line or a subset of internal lines from an n-bit output generated by a chip part, think of this process as akin to "soldering" tap-out lines to the original n-bit bus. For example, given the 16-bit output line *out* from a hypothetical `ChipPart`, if we wish to tap-out the

[1] A useful document detailing the API, schematic design, and HDL implementation of the various built-in chips used in nand2tetris can be accessed at `nand2tetris-hdl.github.io/`.

eight least-significant bits and the eight most-significant bits as two separate 8-bit buses *lsb* and *msb*, respectively, as well as a single line *l* from bit 15, the HDL syntax is as follows:

```
ChipPart(a=, b=, ..., out = out, out[0..7] = lsb, out[8..15] = msb,
         out[15]= l);
```