# Computer Organization, Spring 2019

## Lab 1: 32-bit ALU

## Due: 2019/4/15

## 1. Goal

The goal of this LAB is to implement a 32-bit ALU (Arithmetic Logic Unit) with 32 1-bit ALUs. ALU is the basic computing component of a CPU. Its operations include AND, OR, addition, subtraction, etc. This series of LABs will help you understand the CPU architecture.

## 2. HW Requirement

(1) Please use Icarus Verilog and GTKWave as your HDL simulator.

(2) Please attach your name and student ID as comment at the top of each file.

(3) Please use the Testbench we provide you, and DO NOT modify it.

(4) The names of top module and IO ports must be named as follows:

Top module: alu.v

```verilog
module alu(
        rst_n,          // negative reset              (input)
        src1,           // 32 bits source 1            (input)
        src2,           // 32 bits source 2            (input)
        ALU_control,    // 4 bits ALU control input    (input)
        result,         // 32 bits result             (output)
        zero,           // 1 bit when the output is 0, zero must be set (output)
        cout,           // 1 bit carry out            (output)
        overflow        // 1 bit overflow             (output)
        );
```

ALU starts to work when the signal rst_n is 1, and then catches the data from src1 and src2.

In order to have a good coding style, please obey the rules below:

One module is written in one file.

Module name and file name must be the same.

For example: The file "alu.v" only contains the module "alu".

(5) Basic instruction set (60%)

| ALU action | Description | ALU control input |
|---|---|---|
| AND | Bitwise and | 0000 |
| OR | Bitwise or | 0001 |
| ADD | Addition (with overflow) | 0010 |
| SUB | Subtract | 0110 |
| NOR | Bitwise nor | 1100 |
| NAND | Bitwise nand | 1101 |
| SLT | Set less than (signed) | 0111 |

(6) ZCV three flags: zero, carry out, and overflow (30%)

| Flag name | Operation |
|---|---|
| Zero | Output 1 when the output of ALU is 0<br>Otherwise, output 0 |
| Cout | Output 1 when the carry-out is 1 (for ADD SUB)<br>Otherwise, output 0 |
| Overflow | Output 1 when signed overflow happens (for ADD SUB)<br>Otherwise, output 0 |

# 3.  Architecture diagram
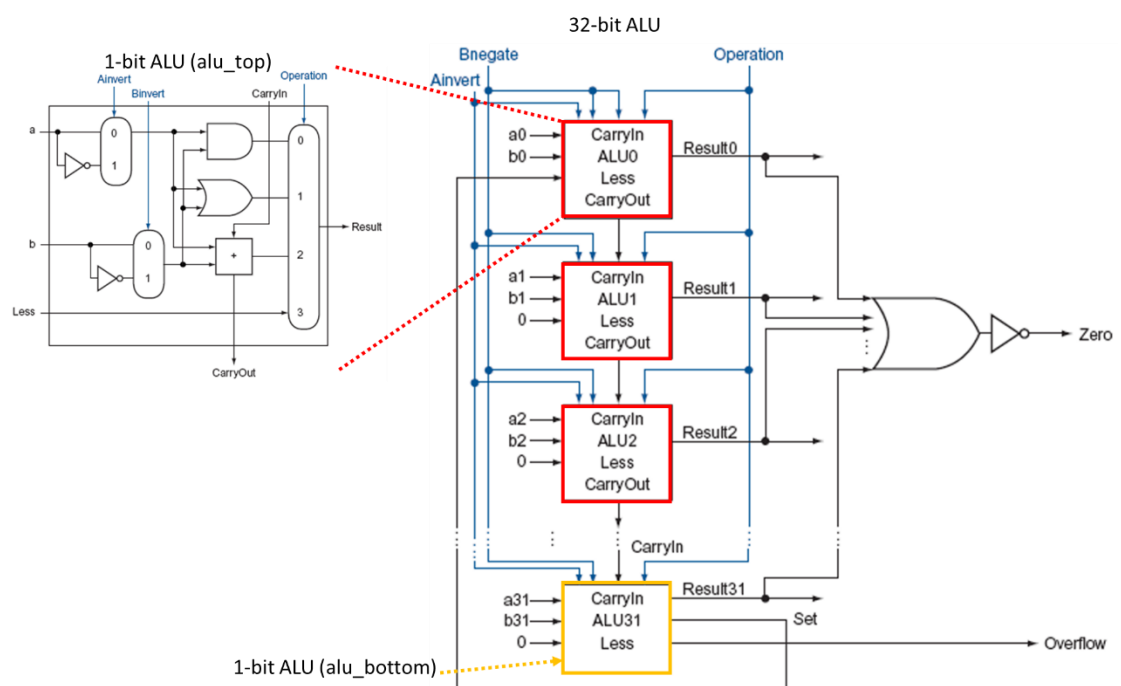


# 4.  Bonus: Extra instruction set (10%)

| ALU action | Description | ALU control input |
|---|---|---|
| SLT | Set less than | 0111_000 |
| SGT | Set greater than | 0111_001 |
| SLE | Set less equal | 0111_010 |
| SGE | Set greater equal | 0111_011 |
| SEQ | Set equal | 0111_110 |
| SNE | Set not equal | 0111_100 |

Hint: Add a module named "Compare" in 1-bit ALU!
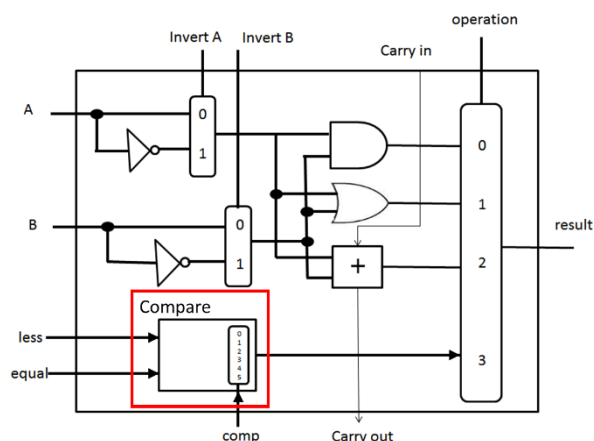
## 5. Report (10%)

Please hand in a PDF file with the following contents:

(1) Your architecture diagram

(2) Detailed description of the implementation

(3) Command for compiling your source codes

(4) Problems encountered and solutions

(5) Lesson learnt (if any)

## 6. Grade

(6) Total: 110 points, including 10 points for your report

(7) Late submission: 10 points off per day

(8) No plagiarism, or you will get 0 point.

## 7. Hand in

(1) Zip your folder and name it as "ID.zip" (e.g., 0616xxx.zip) before uploading to New e3. Other filenames and formats such as *.rar and *.7z are NOT accepted! Multiple submissions are accepted, and the version with the latest time stamp will be graded.

(2) Please include ONLY Verilog source codes (*.v) and your report (0616xxx.pdf) in the zipped folder.

## 8. How to test

The function of testbench is to read input data automatically and output erroneous data. Please put all the .txt files and your program in the same folder, after simulation finishes, you will get some information.



Partial error:

Or all cases pass:

```
*****************************************************
Congratulation! All data are correct!
*****************************************************
```

## 9. Q&A

For any questions regarding Lab 1, please contact Ning-Chi Huang (blackitty321@gmail.com).