# Computer Organization
# Lab 2 - ALU

教授:蔡文錦
助教:劉益先、鄭吉呈、黃芷柔、林彧頎

# Objectives

In Lab 2, we are going to implement an **32-bit ALU** (Arithmetic Logic Unit) by Verilog.

Through this lab, students will learn how to design the function unit of a processor to support its instruction set.

Note that you should design these circuits in gate level as combinational logic instead of sequential logic.

The ALU designed in this lab may also be used in most of the succeeding lab units.

# Tools

- **Icarus verilog (to compile and simulate Verilog code)**
  - Mac OSX
    - brew install icarus-verilog
  - Windows
    - Download this file: https://bleyer.org/icarus/iverilog-v11-20210204-x64_setup.exe
    - Reference tutorial: https://www.youtube.com/watch?v=08S6NxUs-Uo
  - Linux
    - sudo apt install verilog
- **GTKWave (unnecessary)**
  - Easy to debug

# Attached Files

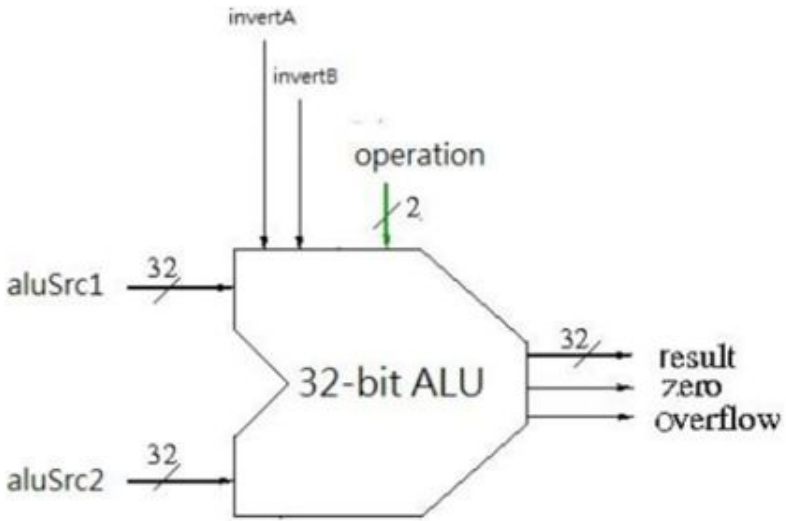
- **You need to do:**
  - MUX_2to1.v
  - MUX_4to1.v
  - ALU_1bit.v
  - ALU.v
- **File to validate the correctness of your implementation:**
  - testbench.v (for 32-bit ALU)
- **Testcase:**
  - *.txt

} **DO NOT modify these files**

※ **Basically, you don't need to add any additional .v file.** We'll test your code using our own testbench.

# 32-bit Arithmetic Logic Unit(ALU) Overview

The block diagram of the 32-bit ALU for this lab is shown in the following figure.

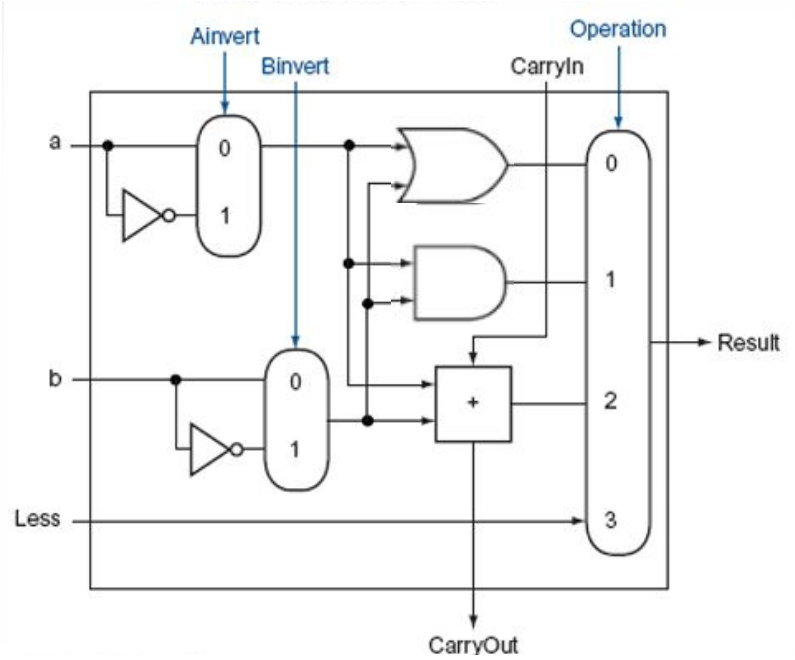



You should generate three outputs of the ALU:

- result: the computation result
- zero: A 1-bit output control signal.
  - set to 1 when the result is 0.
  - set to 0 otherwise.
- overflow: A 1-bit output control signal.
  - set to 1 when the result overflows (add, sub).
  - set to 0 otherwise

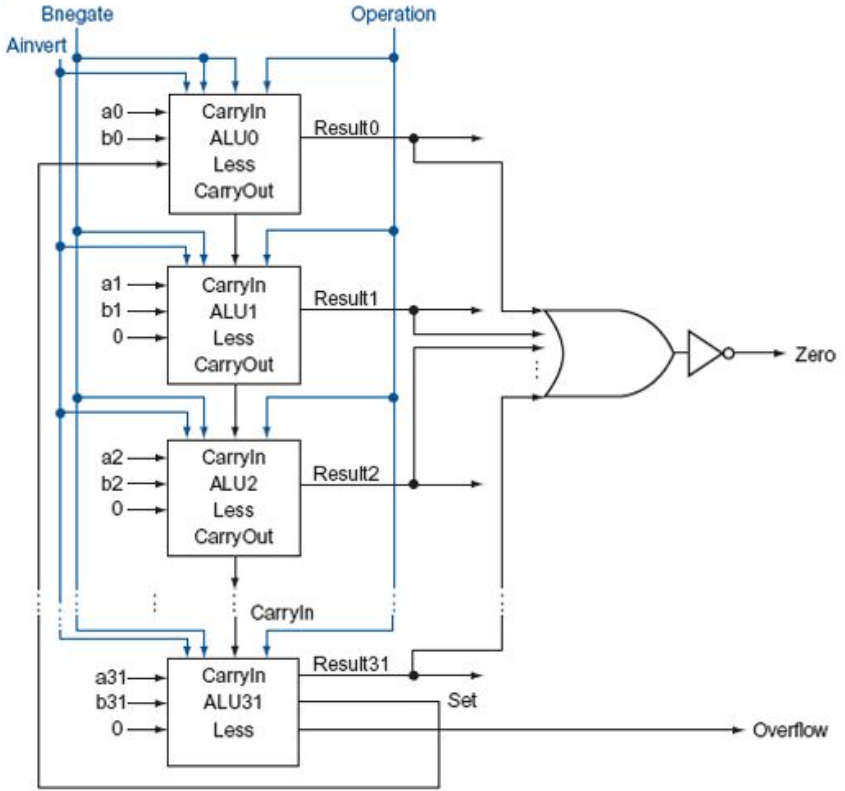
# 1-bit ALU Architecture

The diagram of a 1-bit ALU to be implemented in this lab is shown in the following figure.



# 32-bit ALU Architecture

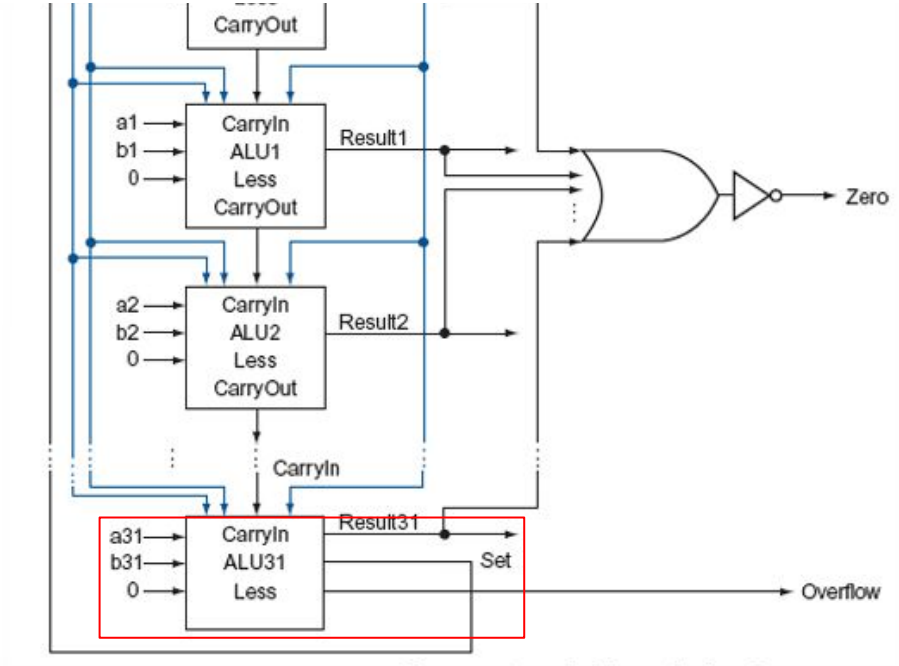
Each input signal of the whole 32-bit ALU will be connected to the corresponding input of 1-bit ALU. Moreover, carryOut of ALUi will be connected to carryIn of ALUi+1.



# Special Design Detail of The Last ALU Unit(ALU31)

The design of ALU31 is different from that of other typical 1-bit ALUs. There are two additional signals generated by ALU31 and are described as follows:

- **set**: A 1-bit control line, it is generated by ALU31 and send to ALU0 as the "**Less**" signal.
- **overflow**: A 1-bit output control signal, it is set to 1 when overflow occurs (add, sub).



# ALU Control Signal

| Operation | Control line | | | Value of result |
|---|---|---|---|---|
| | invertA | invertB | operation | |
| ADD | 0 | 0 | 10 | Src1 + Src2 |
| SUB | 0 | 1 | 10 | Src1 - Src2 |
| AND | 0 | 0 | 01 | Src1 & Src2 |
| OR | 0 | 0 | 00 | Src1 \| Src2 |
| NAND | 1 | 1 | 00 | ~(Src1 & Src2) |
| NOR | 1 | 1 | 01 | ~(Src1 \| Src2) |
| SLT | 0 | 1 | 11 | (Src1 < Src2):? 1:0 |

# Compile & Run

- **Compile**
  - $ iverilog -o lab2 testbench.v ALU.v
  - (windows)iverilog -o lab2 testbench.v  ALU.v

- **Run**
  - $ ./lab2
  - (windows) vvp lab2

**Wrong results:**

```
VCD info: dumpfile alu.vcd opened for output.
*************************************************
*            PATTERN RESULT TABLE               *
*************************************************
* PATTERN *              Result          * ZCV *
*************************************************
* No. 1 error!                                  *
* Correct result: eeeeeeee     Correct ZCV: 000 *
* Your result: xxxxxxxx        Your ZCV: xxx    *
*************************************************
* No. 2 error!                                  *
* Correct result: 00000000     Correct ZCV: 100 *
* Your result: xxxxxxxx        Your ZCV: xxx    *
*************************************************
* No. 3 error!                                  *
* Correct result: 9bf5fea6     Correct ZCV: 000 *
* Your result: xxxxxxxx        Your ZCV: xxx    *
*************************************************
```

**All testcase PASS:**

```
$ ./lab2
VCD info: dumpfile alu.vcd opened for output.
*************************************************
*            PATTERN RESULT TABLE               *
*************************************************
* PATTERN *              Result          * ZCV *
*************************************************
*      Congratulation! All data are correct!    *
*************************************************
Correct Count: 30
testbench.v:91: $finish called at 415000 (1ps)
```

# Grading Policy

- There are **10 hidden cases**, and you will get 10 points for each correct testcase, totally 100 points.
- **Any assignment work by fraud will get a zero point！**
- **No late submission！**

# Submission

- **Please attach student IDs as comments at the top of each file.**
- The files you should hand in include:
  - **MUX_2to1.v**
  - **MUX_4to1.v**
  - **ALU_1bit.v**
  - **ALU.v**
- Compress the above file into one zip file, and name your zip file as **HW2_{studentID}.zip (e.g. HW2_123456789.zip)**
  - **Make sure not to add an extra folder layer.**
- **Wrong format will have 20% penalty！**
- **Deadline: 8/6 23:55**