ALU

How to write this chapter

`Where we are?`

The chapter 1 gives an introduction and the fundation ,and from this chapter we will foucs on **building the components of a real computer**.

To build a computer, there is only 2 things we are going to consider about, **how to maintain data,** and **how to operate on data**. The latter one can be further divided into what's the **basic operations** and **how to do these operations**, and in this chapter we will foucs on all the basic operations needed in a computer.

The outcome of this chapter will be an ALU which contains all of the basic operations, which can be viewed as an **2-in 1-out function**. So where do we get the input and store the output? What shall we do? (Hardware Level: Who tells us? Language Level:In which form we tell a machine to do something?)We will deal with these questions in chapter3, 4, 5 and 6.

`Ideas delivery`

By writing this chapter, we will focus on deliverying the follow ideas:

- Computer is all about **operations on the state of data**, and great amout of operations can be **rooted to several meta operations**. Only add. And this kind of thinking method can be implemented into other areas.
- Again, it's a kind fof **translation** to analyze how to do add and substraction, translaing how we do things to binary logic.
- In building the complex system, we will foucs on building realizable **basic units**, and then to **abstract** them to build more complex things. Abstraction means that we will only focus on the port, the surface, the overall function, without considering about the inside structure and realization.

`Project developing`

2 parts:

- Basic: Add and substraction
- Final outcome: ALU

Try to develop this part to make the reader think that it is something easy to understand: it's not a hard part, and one can understand the important ideas and clear structure of ALU easily. Let the readers try to do the project by theirself, and provide essential hints to help them and deepen their understanding.

Introduction: completenss of add

`How we look at a computer?`

Computer is all about **operations on the state of data**. The OS and all the software we see can be divided into 2 parts , frontend and backend. Backend by nature is about data structure and logic. Frontend deals with what users see and how we interact with computer, and these are also data: RGB, (x,y), a character.

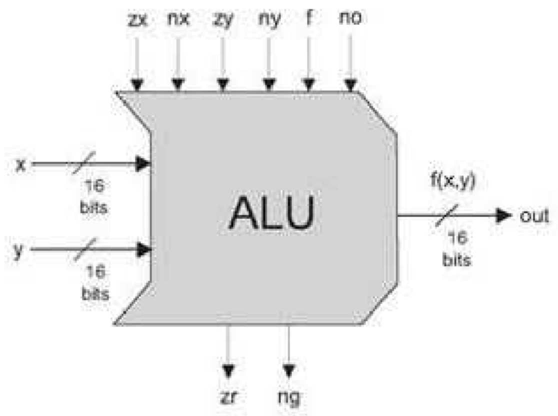
`Everything is add`

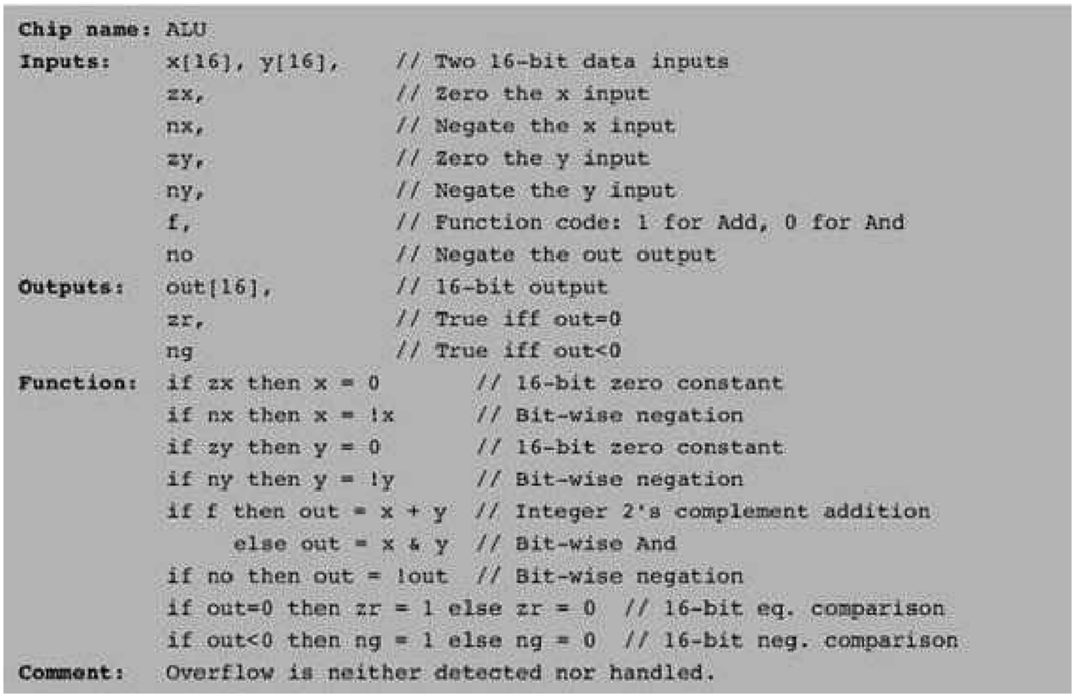
How many kinds of operations do we need in a computer? 1000, or 10000? Actually, we only need one(roughly): add.

Consider about **math** we learn from childhood to now. Substraction can be viewed as adding a negetive number, and we get "×"and "÷" by adding or substracting in a common procedure. Even those in the Advanced Mathematics, like matrix operation and integration, is no more than adding and substraction.Well, maybe add can't do everything, shifting for example, but almost everything.

Project development

- "**+**"
    - Intro to the binary number, provided to those never know it.(in hyberlink)
    - Hint: Think about **how we** do adding. Using **truth table** in chapter 1. "**Add twice**"
- "**-**"
    - Intro: Point out that the length of a number in computer is limited. It's like a clock.
    -
    - Hint: What's do we get by adding 6 and 7 when the max is 12? How to get the number in binary means?
    - Disscussion: difference between this one and how we do substraction. Why this?
- "**ALU**"
    - Intro: the picture:



- 

```
Chip name: ALU
Inputs:    x[16], y[16],     // Two 16-bit data inputs
           zx,               // Zero the x input
           nx,               // Negate the x input
           zy,               // Zero the y input
           ny,               // Negate the y input
           f,                // Function code: 1 for Add, 0 for And
           no                // Negate the out output
Outputs:   out[16],          // 16-bit output
           zr,               // True iff out=0
           ng                // True iff out<0
Function:  if zx then x = 0        // 16-bit zero constant
           if nx then x = !x       // Bit-wise negation
           if zy then y = 0        // 16-bit zero constant
           if ny then y = !y       // Bit-wise negation
           if f then out = x + y   // Integer 2's complement addition
                else out = x & y   // Bit-wise And
           if no then out = !out   // Bit-wise negation
           if out=0 then zr = 1 else zr = 0  // 16-bit eq. comparison
           if out<0 then ng = 1 else ng = 0  // 16-bit neg. comparison
Comment:   Overflow is neither detected nor handled.
```

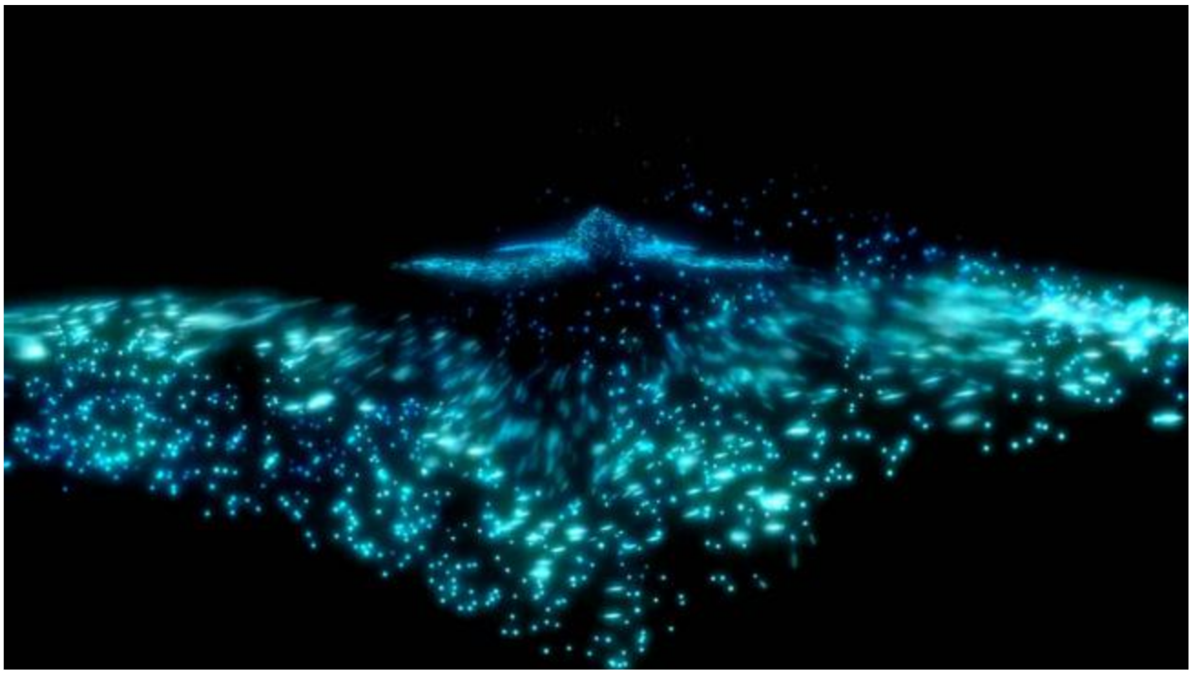
- (from the book)
- Hint: divide it into several logic parts and conquer them.

Futher reading

```
Universality and waste: compared to TPU unit.(maybe)
```

summary

- Point out the ideas in the chapter.
- Tell the reader that these ideas are powerful tools in all areas and give some examples.
  - example: particle system in after effect. With the basic unit of a particle, we can create amazing video.

- 

(from: http://weili.ooopic.com/weili_11291382.html)