# ICS第一次讨论课

Presented by 钟书锐

## 评分标准

所有标准三个平行班级完全统一

**作业（6分）**
按照往年经验，一般是布置6次作业，每次作业1分
学生只要按时提交，认真完成即可得分
发现雷同情况的，当次作业均不得分

**讨论班（6分+2分）**
无故不参加讨论班，每周扣1分
学生每学期有2次请假机会，假签到惩罚每次扣2分
主题报告：不强制要求，完成的同学获得奖励1分，表现特别优秀的可以奖励2分

## 评分标准

**实验（48分+10分）**
6次课程实验，每个实验8分
第一次实验用机器语言完成，第六次实验用C语言完成

附加实验：LC-3汇编器、模拟器，每个5分
英才班同学要求两个实验必做，其他同学可以选做1~2个，也可以不做；

**考试（20+20分）**
考试以**开卷形式**，时长2小时，卷面100分
题目形式可以是选择题、填空题、简答题、复合型题目等；

6+6+2+48+10+20+20=112分
PS：最终总评超过100分时按100分提交教务系统

## 作业讲解

存在问题

1. 部分同学忘记交，不接受补交的

2. 没有按照要求交PDF，有上传图片合集的
可以在本子写，然后相应软件转PDF
也可以直接写latex，markdown 输出PDF

3. 对文件命名没有要求，但是在提交的时候填好姓名学号，自动命名

## 作业讲解

1. (Adapted from problem 1.5 in the textbook)
   Say we had a "black box," which takes two numbers as input and outputs their sum. See Figure 1.10a in the Textbook or the following figure. Say we had another box capable of multiplying two numbers together. See Figure 1.10b. We can connect these boxes together to calculate p * (m + n). See Figure 1.10c. Assume we have an unlimited number of these boxes. Show how to connect them together to calculate:
   a. ax+b
   b. The average of the four input numbers w, x, y, and z
   c. $a^2 + 2ab + b^2$ (can you do it with one add box and one multiply box?)
   d. $a^6$ (can you do it using only 3 multiply boxes?)

## 作业讲解

2. (2.3)
   a. Assume that there are about 400 students in your class. If every student is to be assigned a unique bit pattern, what is the minimum number of bits required to do this?
   b. How many more students can be admitted to the class without requiring additional bits for each student's unique bit pattern?

a) $2^x > 400$

$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$

$x = 9$

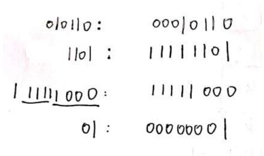
at least 9 bits.

b) $2^9 - 400 = 512 - 400 = 112$

112 students can be admitted.

## 作业讲解

3. (Adapted from 2.13)
Without changing their values, convert the following 2's complement binary numbers into 8-bit 2's complement numbers.
    a. 010110
    b. 1101
    c. 1111111000
    d. 01



| | | |
|---|---|---|
| 010110 | : | 00010110 |
| 1101 | : | 11111101 |
| 1111111000 | : | 11111000 |
| 01 | : | 00000001 |

注意其中数值0和−1的码字，即11111和00000。当我们知道对数值−1做加00001的操作时，我们看到ALU的输出结果确实为00000，但同时还有一个进位。由于进位对结果不产生影响[⊖]，所以我们说−1在加00001之后的结果是0，而不是100000。在此，进位bit被丢弃了，而事实上在补码运算中，进位bit始终是被忽略的。

Note in particular the representations for −1 and 0, that is, 11111 and 00000. When we add 00001 to the representation for −1, we do get 00000, but we also generate a carry. That carry, however, does not influence the result. That is, the correct result of adding 00001 to the representation for −1 is 0, not 100000. Therefore, the carry is ignored. In fact, because the carry obtained by adding 00001 to 11111 is ignored, the carry can *always* be ignored when dealing with 2's complement arithmetic.

## 作业讲解

4. (Adapted from 2.17)

Compute the following. Assume each operand is a 2's complement binary number.

a. 01 + 1011
b. 11 + 01010101
c. 0101 + 110
d. 01 + 10

$$\begin{array}{r} 0101 \\ 1110 \\ \hline 10011 \end{array}$$

×

a.
$$\begin{array}{r} 1011 \\ 0001 \\ \hline 1100 \end{array}$$
1100

b.
$$\begin{array}{r} 01010101 \\ 11111111 \\ \hline 01010100 \end{array}$$
01010100

c.
$$\begin{array}{r} 0101 \\ 1110 \\ \hline 0011 \end{array}$$
0011

d.
$$\begin{array}{r} 01 \\ 10 \\ \hline 11 \end{array}$$
11

## 作业讲解

5. Convert the following 8-bit 2's complement binary numbers into decimal numbers.
   a. 01010101
   b. 10001101
   c. 10000000
   d. 11111111

01010101 $= 1+2^2+2^4+2^6$

$= 1+4+16+64$

$= 85$

10001101 $\quad 1\,\underbrace{1110011}$

$= -(1+2+2^4+2^5+2^6)$

$= -(1+2+16+32+64)$

$= -(80+35)$

$= -115$

10000000 $\quad -2^7 = -128$

11111111 $\quad 1\,0000001$

$= -(1)$

$= -1$

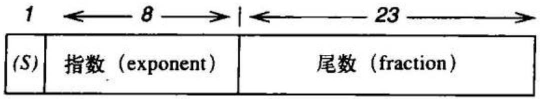
至此，在我们的5-bit系统中，已解决了数值0和15个正数及其对应的15个负数的补码编码分配问题。总共使用了15+15+1=31个码字，我们知道5-bit可以表示$2^5$=32个码字，那么，剩下一个码字10000怎么分配值呢？

我们注意到−1的码字是11111，−2是11110，−3是11101，依次类推。最后，我们发现，−15的码字是10001，如果将10001继续减1，则得到10000。所以，顺应ALU的操作方式，我们可以让码字10000对应为数值−16。

在第5章中，我们将介绍一个被称为LC-3（LC代表Little Computer）的计算机。LC-3将采用16-bit补码方式，因此LC-3的数值表示范围为从−32 768到+32 767之间的所有整数。

We note that −1 is 11111, −2 is 11110, −3 is 11101, and so on. If we continue this, we note that −15 is 10001. Note that, as in the case of the positive representations, as we sequence backwards from representations of −1 to −15, the ALU is subtracting 00001 from each successive representation. Thus, it is convenient to assign to 10000 the value −16; that is the value one gets by subtracting 00001 from 10001 (the representation for −15).

In Chapter 5 we will specify a computer that we affectionately have named the LC-3 (for Little Computer 3). The LC-3 operates on 16-bit values. Therefore, the 2's complement integers that can be represented in the LC-3 are the integers from −32,768 to +32,767.



$$N = (-1)^s \times 1.\text{尾数} \times 2^{\text{指数}-127}, 1 \leqslant \text{指数} \leqslant 254$$

图2-2 浮点数类型

Most ISAs today specify more than one floating point data type. One of them, usually called *float*, consists of 32 bits, allocated as follows:

```
 1 bit for the sign (positive or negative)
 8 bits for the range (the exponent field)
23 bits for precision (the fraction field)
```

In most computers manufactured today, the format of the 32-bit floating point data type is as shown in Figure 2.3.

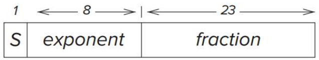


**Figure 2.3** **The 32-bit floating point data type.**

## 作业讲解

6. Express the value 0.3 in the 32-bit floating point format that we discussed in class today. Feel free to only show fraction bits [22:15], rather than all the fraction bits, [22:0]. Notation: The symbol [22:15] signifies all 8 bits from bit 22 to bit 15.

    0 01111101 00110011

⑦ $0.3 = 0.0\,1001\,1001\,1001\,1001\cdots\cdots$

| | |
|---|---|
| $0.3\times2=0.6$ | 0 |
| $0.6\times2=1.2$ | 1 |
| $0.2\times2=0.4$ | 0 |
| $0.4\times2=0.8$ | 0 |
| $0.8\times2=1.6$ | 1 |
| $0.6\times2=1.2$ | 1 |

⋮

正则处理 $0.3 = 1.001\,1001\,1001\,1001\cdots\times2^{-2}$

符号位为 0

指数 1111101 $(125-127=-2)$

尾数 $001\,1001\,1001\,1001\;\cdots$

$[22:15] = \underline{001\,10011}$

## 作业讲解

7. Convert the following floating point representation to its decimal equivalent:

   1 10000010 10101001100000000000000

   -13 19/64

符号位 1 负

$1.0000010 = 2+2^7 = 128+2$ 指数为3

$128+2-127 = 3$

尾数 $10101001100000\cdots\cdots 0$

$\therefore$ 原式 $= 2^3 \times 1.101010011$

$= 1101.010011$

$2^3+2^2+1 + 2^{-2}+2^{-5}+2^{-6}$

$1+2+16 = 19$

$= 8+4+1 + \frac{1}{4} + \frac{1}{32} + \frac{1}{64}$

$= 13\frac{19}{64}$

## 作业讲解

8. Add the two hexadecimal 2's complement integers below:

```
    x90A
+  x4123
```

F90A + 4123 = 3A2D

## 作业讲解

9. (Adapted from 2.50)

Perform the following logical operations. Express your answers in hexadecimal notation.

a. xABCD OR x9876

xBBFF

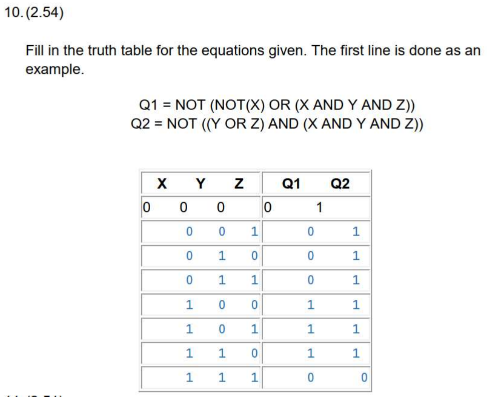
b. x1234 XOR x1234

x0000

c. xFEED AND (NOT(xBEEF))

x4000

## 作业讲解



10. (2.54)

Fill in the truth table for the equations given. The first line is done as an example.

Q1 = NOT (NOT(X) OR (X AND Y AND Z))
Q2 = NOT ((Y OR Z) AND (X AND Y AND Z))

| X | Y | Z | Q1 | Q2 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

## 作业讲解

11. (2.51)

What is the hexadecimal representation of the following numbers?

a. 25,675

x644B

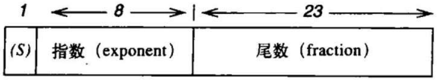
b. 675.625 (i.e. $675\frac{5}{8}$), in the IEEE 754 floating point standard

x4428E800

c. The ASCII string: Hello

x48656C6C6F

## 作业讲解



IEEE754规约浮点数能表示的最大数与最小数分别是多少？
最小正数又是多少

要使表示的数最大,
尾符一定为0表正数,
阶码一定最大，但八位全1不符合规范，最大是11111110表示254，减去偏移量127，得127
尾数23位都取1，小数点之前也可以保存一个1，因此是
2^0+2^-1+⋯+2^-23=2-2^-23
又阶码最大为127，故最大正数为2^127 ∗ (2-2^-23),得
2^128-2^104

$$(2-2^{-23}) * 2^{127}$$

$$1.0 * 2^{-126}$$

### 2.7.1.2 Infinities

We noted above that the floating point data type represented numbers expressed in scientific notation in normalized form provided the exponent field does not contain 00000000 or 11111111.

If the exponent field contains 11111111, we use the floating point data type to represent various things, among them the notion of infinity. *Infinity* is represented by the exponent field containing all 1s and the fraction field containing all 0s. We represent positive infinity if the sign bit is 0 and negative infinity if the sign bit is 1.

### 2.7.1.3 Subnormal Numbers

The smallest number that can be represented in normalized form is

$$N = 1.00000000000000000000000 \times 2^{-126}$$

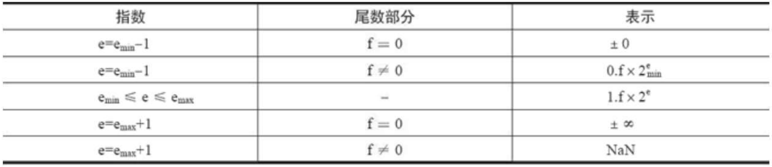
What about numbers smaller than $2^{-126}$ but larger than 0? We call such numbers *subnormal numbers* because they cannot be represented in normalized form. The largest subnormal number is

$$N = 0.11111111111111111111111 \times 2^{-126}$$

The smallest subnormal number is

$$N = 0.00000000000000000000001 \times 2^{-126}\text{, i.e., } 2^{-23} \times 2^{-126} \text{ which is } 2^{-149}.$$

Note that the largest subnormal number is $2^{-126}$ minus $2^{-149}$. Do you see why that is the case?



| 指数 | 尾数部分 | 表示 |
|---|---|---|
| $e=e_{min}-1$ | $f=0$ | $\pm 0$ |
| $e=e_{min}-1$ | $f\neq 0$ | $0.f\times 2^{e}_{min}$ |
| $e_{min}\leqslant e\leqslant e_{max}$ | – | $1.f\times 2^{e}$ |
| $e=e_{max}+1$ | $f=0$ | $\pm\infty$ |
| $e=e_{max}+1$ | $f\neq 0$ | NaN |

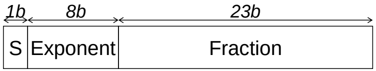
# CS1002A.02 第二次讨论课

王朝晖

2021 秋

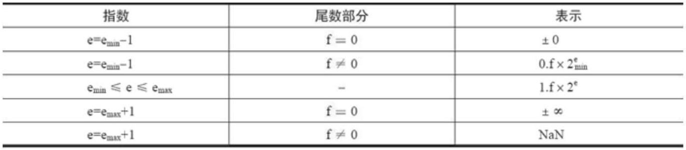
1. What is the smallest positive normalized number that can be represented using the IEEE Floating Point standard?

- 浮点数表示



- Normalized number:
  - Exponent: [00000001, 11111110]

- Smallest positive normalized number
  - S = 0; E = 00000001; F = 000000···000
  - 2^(-126)



| 指数 | 尾数部分 | 表示 |
|---|---|---|
| $e=e_{min}-1$ | $f=0$ | $\pm 0$ |
| $e=e_{min}-1$ | $f \neq 0$ | $0.f \times 2^{e_{min}}$ |
| $e_{min} \leqslant e \leqslant e_{max}$ | – | $1.f \times 2^{e}$ |
| $e=e_{max}+1$ | $f=0$ | $\pm \infty$ |
| $e=e_{max}+1$ | $f \neq 0$ | NaN |

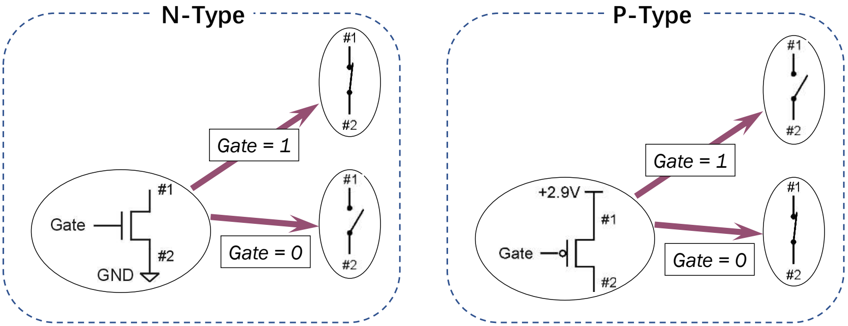
## What about the smallest positive integer that can NOT be represented using IEEE Floating Point standard?

- 规约数

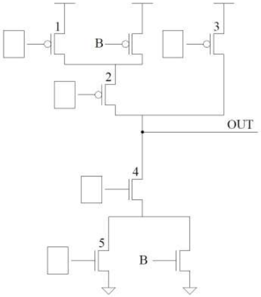
$$1.[23bits] \times 2^{-126\sim127}$$

- 乘$2^{k}$相当于向左移k位; 所以$2^{0}\sim 2^{24}$.都能被表示
- 左移23位之后
  - 111···111→ 111···111000···
- 不能表示的最小正整数:
  - 1.00...000(23个0)1× $2^{24}$
  - 即$2^{24}+1$

3. Draw a transistor-level diagram for a three-input NAND gate and a three-input NOR gate.



## 3.c

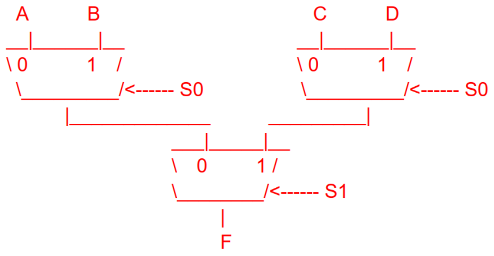


| A | B | C | OUT |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

- OUT == 1, GND不连通, +连通
- OUT == 0, GND连通, +不连通
- OUT不能同时接GND和+

6. (2) Implement the 4-to-1 mux using only 2-to-1 muxes making sure to properly connect all of the terminals.

- 扩展选择器



## 6. (3) Implement F = A XOR B using ONLY two 2-to-1 muxes. You are not allowed to use a NOT gate

- XOR: $\bar{A}B + A\bar{B}$
- 2-1 mux: $\bar{S}A + SB$

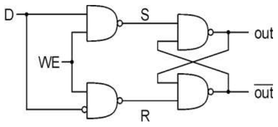
- 2-1 mux 实现 XOR

```
      B          B'
      |          |
  ____|__________|___
  \   0          1  /
   \_______________/<------ A
      |
      F
```

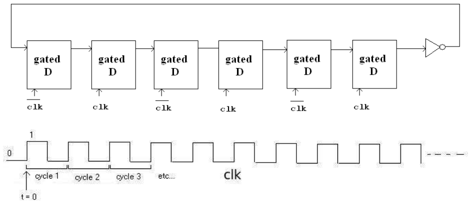
- 2-1 mux 实现 NOT

```
      1          0
  ____|__________|___
  \ 0            1  /
   \_______________/<------ B
      |
      B'
```

10. What is the state after 50 cyles. How many cycles does it take for a specific state to show up again? initially the state is 000000



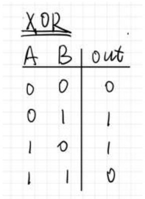
WE == 1, out = D
WE == 0, out Hold



**Cycle1 A: 000000**
**Cycle1 B: 100000**
**Cycle2 A: 110000**
**Cycle2 B: 111000**
**Cycle3 A: 111100**
**Cycle3 B: 111110**
**Cycle4 A: 111111**
**Cycle4 B: 011111**
**Cycle5 A: 001111**
**Cycle5 B: 000111**
**Cycle6 A: 000011**
**Cycle6 B: 000001**

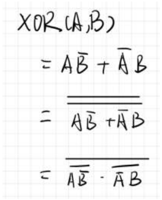
**Cycle7 A: 000000**

## 11. Draw the transistor level circuit of a 2 input XOR gate



XOR

| A | B | out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

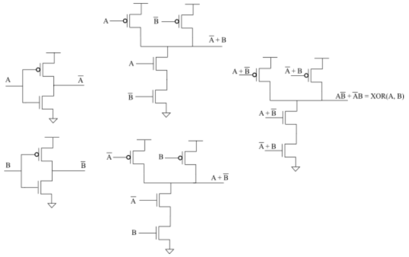


XOR(A,B)

$$= A\bar{B} + \bar{A}B$$

$$= \overline{\overline{A\bar{B} + \bar{A}B}}$$

$$= \overline{\overline{A\bar{B}} \cdot \overline{\bar{A}B}}$$



Thank You!