Hw2

T1

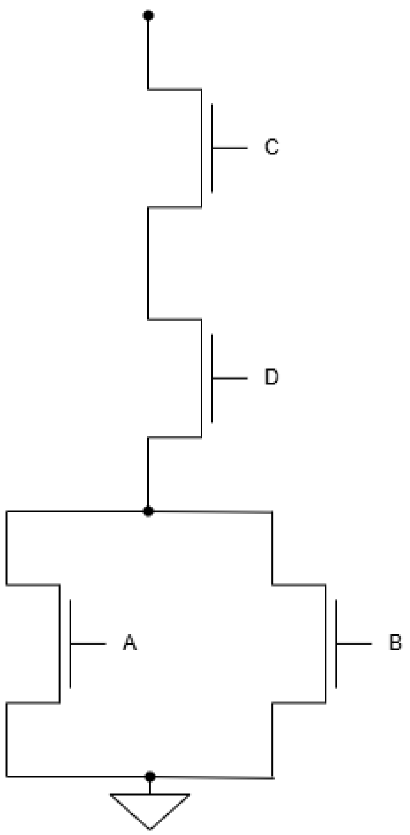
- X ;
- 1 ;
- 0 ;
- X ;
- 0 .

T2

1. ◦

A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2. ◦



T3

1. - ① 证明可以表示 **NOT**：

     由于 $NOT(A) = NOT(A \ \ OR \ \ A) = A \ \ NOR \ \ A$，故 **NOR** 可以实现 **NOT**。
   - ② 证明可以表示 **OR**：

     由于 $A \ \ OR \ \ B = NOT(NOT(A \ \ OR \ \ B)) = NOT(A \ \ NOR \ \ B) = (A \ \ NOR \ \ B) \, NOR \, (A \ \ NOR \ \ B)$，故 **NOR** 可以 实现 **OR**。
   - ③ 证明可以表示 **AND**：

     由于 $A \ \ AND \ \ B = NOT((NOT \ A) OR (NOT \ B)) = (NOT \ A) \, NOR \, (NOT \ B) = (A \ \ NOR \ \ A) \, NOR \, (B \ \ NOR \ \ B)$

     故 **NOR** 可以实现 **AND**。
   - 综上，**NOR** 是逻辑完备的。
2. - 由于 **XOR** 满足交换律和结合律，故而多个异或的运算可以写为 $F(A, B) = aA \ \ XOR \ bB \ \ XOR \ \ c$ (其中 $a, \ b, \ c \in \{0, 1\}$);
   - 反证法。若 **XOR** 是逻辑完备的，那么多个 **XOR** 必然能表示 **OR**。
   - 于是，$A \ \ OR \ \ B = aA \ \ XOR \ bB \ \ XOR \ \ c$。
   -

A	B	LEFT	a	b	c	RIGHT
0	0	0			0	0
0	1	1		1		1
1	0	1	1			1
1	1	1	1	1	0	0

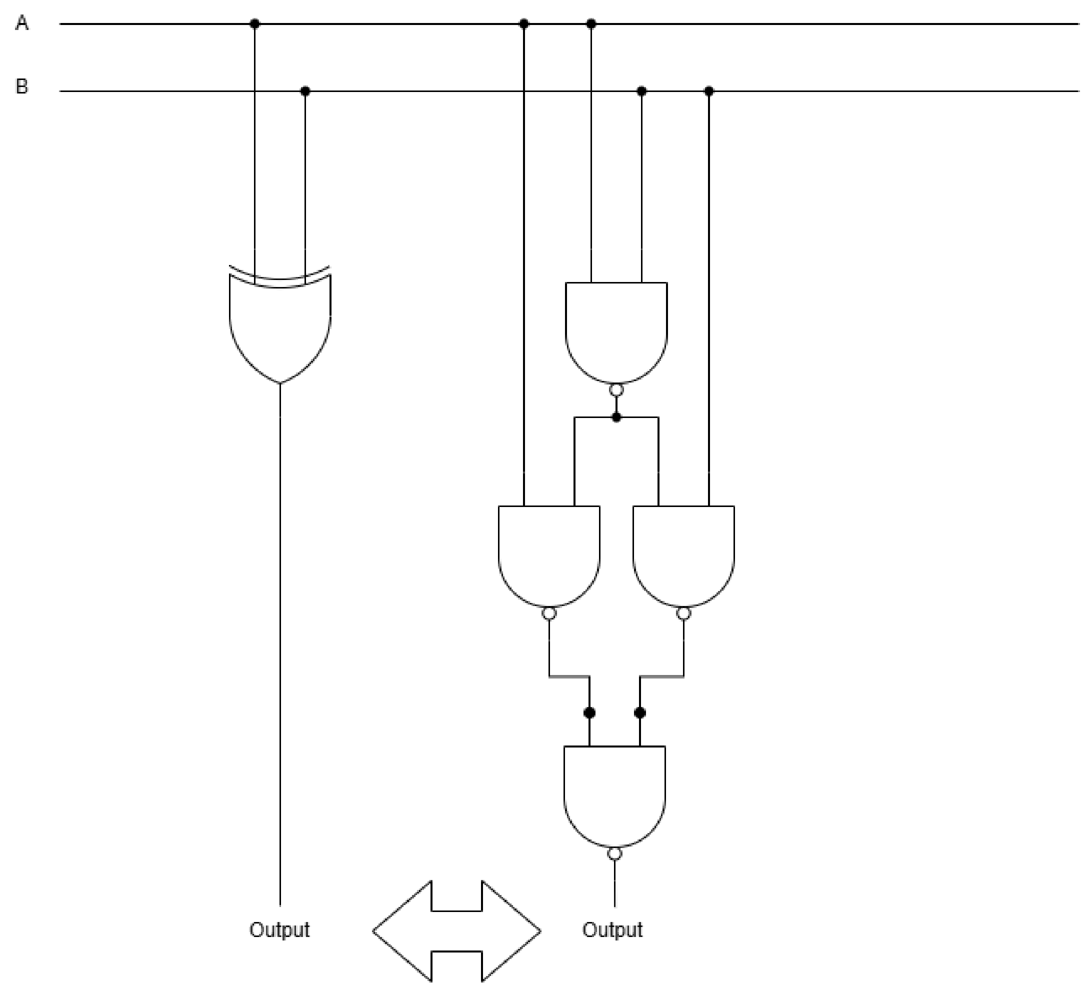
     注：a, b, c 是依据前三条计算出来的，然后用第四条验证。
   - 由于在第四条中 $LEFT \neq RIGHT$，即出现矛盾——故而，**XOR** 不能表示 **OR**。
   - 即 **XOR** 是不完备的。

T4

1. - 2^{-126}
2. - $2^{-149} \sim 2^{-126} - 2^{-149}$
3. - ① 如果非规格化数部分也根据规格化数的规则编码，那么浮点数无法表示 0，能表示的绝对值最小的数是 $\pm 2^{-127}$；
   - ② 如果非规格化数部分也根据规格化数的规则编码，那么浮点数无法表示接近 0 的数。即使将 0x80000000 和 0x00000000 均单拎出来定义为 ± 0，此时绝对值最小的数也为 $\pm(2^{-127} + 2^{-150})$，相较于非规格化数的 $\pm 2^{-149}$，距离 0 明显较远，导致较小数的精度较低。
4. - 不均匀。
   - 绝对值越小，数值分布越密集；绝对值越大，数值分布越分散。

T5

- - $A \oplus B = (A \cdot \overline{B}) \ + \ (\overline{A} \cdot B) = \overline{\overline{A \cdot \overline{B}} \ \cdot \ \overline{\overline{A} \cdot B}}$
    - $\overline{A \cdot \overline{B}} = \overline{A \cdot (\overline{A} + \overline{B})} = \overline{A \cdot \overline{A \cdot B}}$
    - 且 $NAND(A, \ B) = \overline{A \cdot B}$，则 $A \oplus B = NAND(NAND(A, \ NAND(A, \ B)), \ NAND(NAND(A, \ B), \ B))$
- 电路图如下：

- 

T6

1. A[1:0] = 00; WE = 1;
2. - 寻址空间 (address space): 指内存中可独立识别的位置总数。通常由地址总线的宽度（位数）决定。
   - 寻址能力 (addressability): 指每个地址可以存储的数据位数。
   - 寻址空间：$2^2 - 1 = 3$；
   - 寻址能力：4 bits。
3. 将存储器阵列的行数从 4 行增加到 k 行，并相应增加数据输入线和数据输出线，使得从 $D_i[3:0]$ 到 $D_i[k-1, 0]$，$D[3:0]$到 $D[k-1:0]$。

T7

- $2^5 < 48 < 2^6$，$2^4 < 28 < 2^5$ 。
- $32 - 6 - 5 \times 2 = 16$
- $2^{15} - 1 = 32767$，$-2^{15} = -32768$
- **IMM** 表示的范围是 $-32768 \sim 32767$ 。

T8

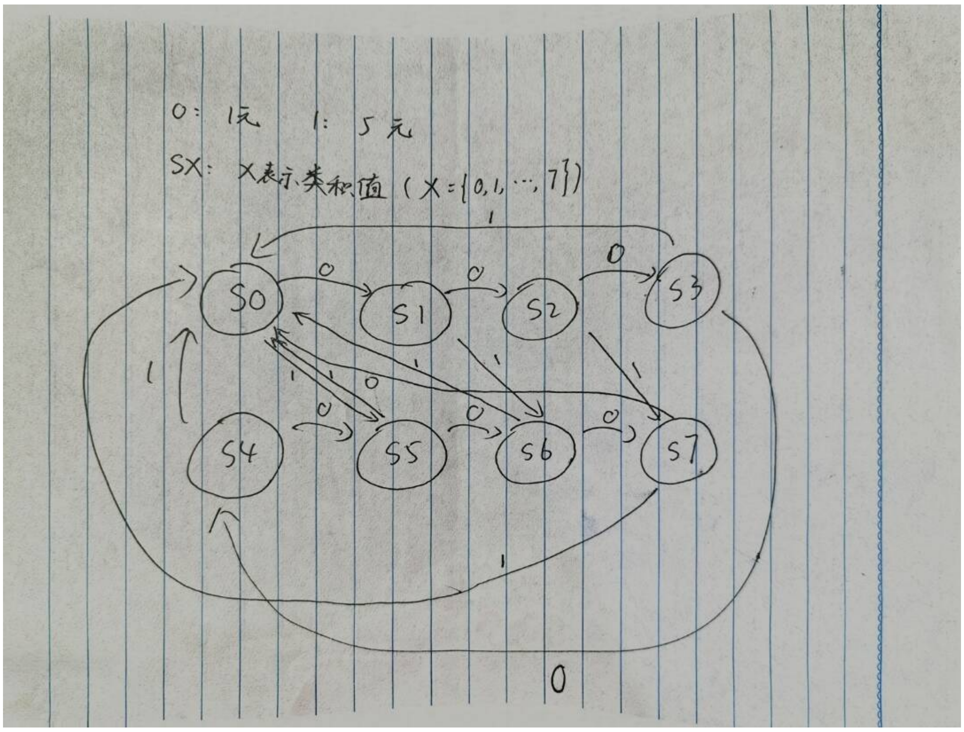
1. 冯·诺依曼模型包括5个组成部分：内存、处理单元、输入、输出、控制单元。
   - **内存**：用于存储程序指令和数据。
   - **处理单元**：负责执行算术和逻辑运算。
   - **输入**：用于将外部数据或指令输入计算机。
   - **输出**：用于将计算机处理后的结果输出到外部环境。
   - **控制单元**：负责协调和管理计算机各组件的工作，保证指令的有序执行。
2. - 冯·诺依曼模型的核心是“存储程序概念”，即程序指令和数据都以二进制形式存储在内存中，控制单元按顺序读取和执行指令。
   - - **并行性冲突**：量子计算基于叠加态并行处理，而冯·诺依曼是顺序执行。
       - **概率性输出**：量子计算结果具有概率性，而冯·诺依曼要求确定性。

- **内存限制**：量子不可克隆定理禁止复制量子态，无法实现内存的随机访问和复制数据。

T9

- 000111；
- 12；
- 6；
- 6。

T10

- 

T11

1. - 将八杯酒进行二进制编号，从0001，一直到1000；（如第1杯酒对应0001）
   - 4个小鼠分别对应一个二进制位，让1号小鼠喝第一位为1的酒，依次顺延，直到4号小鼠喝第四位为1的酒；
   - 到第二天，可根据小鼠死亡情况，判断是第几杯酒有毒。（如1，2，3号小鼠都死了，说明是第7杯酒）。
2. 16杯酒（将第16杯酒编号到0000）。
3. - 1.625。
   - 可以修改策略，更改编号为0000，0001，0010，0100，1000，0011，0110，1100，此时预期死亡量为1.25。
