**Chapter 4**

**3.31 If a computer has eight-byte addressability and needs three bits to access a location in memory, what is the total size of memory in bytes?**

Since it needs three bits to access a location in memory, the Memory Space of this computer is 8. In addition, it has eight-byte addressability.

2^3\*8=64

**3.33 Using Figure 3.21, the diagram of the 4-entry, 22-by-3-bit memory.**

**a. To read from the fourth memory location, what must the values of A[ 1:0] and WE be?**

|  |  |
| --- | --- |
| **1st** | **00** |
| **2nd** | **01** |
| **3 th** | **10** |
| **4 th** | **11** |

The fourth memory location: 11

To read from the memory location: WE=0

**b. To change the number of entries in the memory from 4 to 60, how many address lines would be needed? What would the addressability of the memory be after this change was made?**

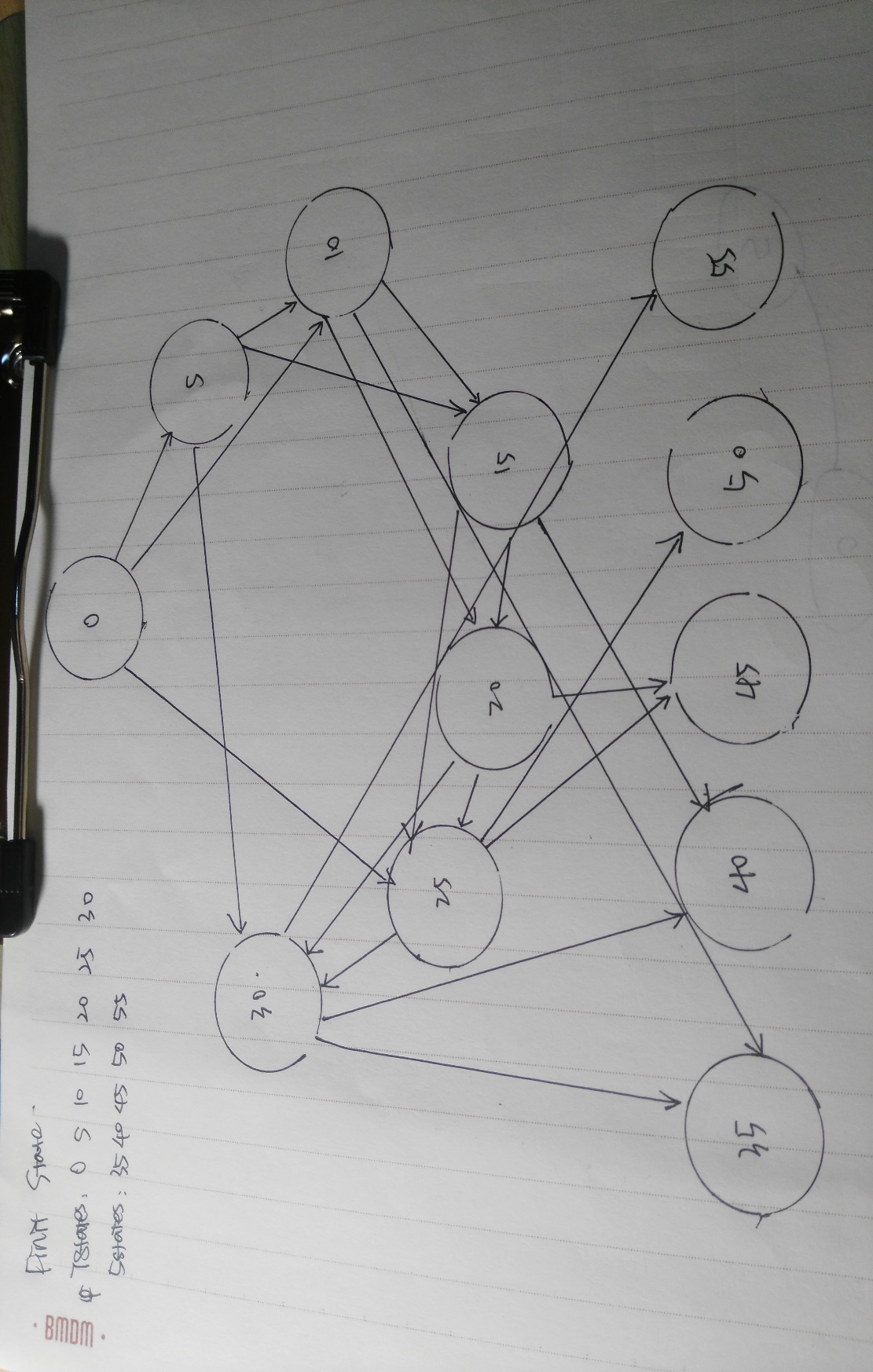
60<64=2^6 🡪 6 address lines would be needed

The addressability of the memory will not change, but the address space will change.

**c. Suppose the minimum width (in bits) of the program counter (the program counter is a special register within a CPU, and we will discuss it in detail in, the next chapter) is the minimum number of bits needed to address all 60 locations in our memory from part (b). How many additional memory locations could be added to this memory without having to alter the width of the program counter?**

64-60=4

**3.41 The IEEE campus society office sells sodas for 35 cents. Suppose they install a soda controller that only takes the following three inputs: nickel, dime, and quarter. After you put in each poin, you push a pushbutton to register the coin. If at least 35 cents has been put in the controller, it will output a soda and proper change (if applicable). Draw a finite state machine that describes the behavior of the soda controller. Each state will represent how much money has been put in (Hint'. There will be seven of these states). Once enough money has been put in, the controller will go to a final state where the person will receive a soda and proper change (Hint: There are five such final states). From the final state, the next coin that is put in will start the process again.**

****

**3.43 Shown in Figure 3.43 is an implementation of a finite state machine with an input X and output Z.**

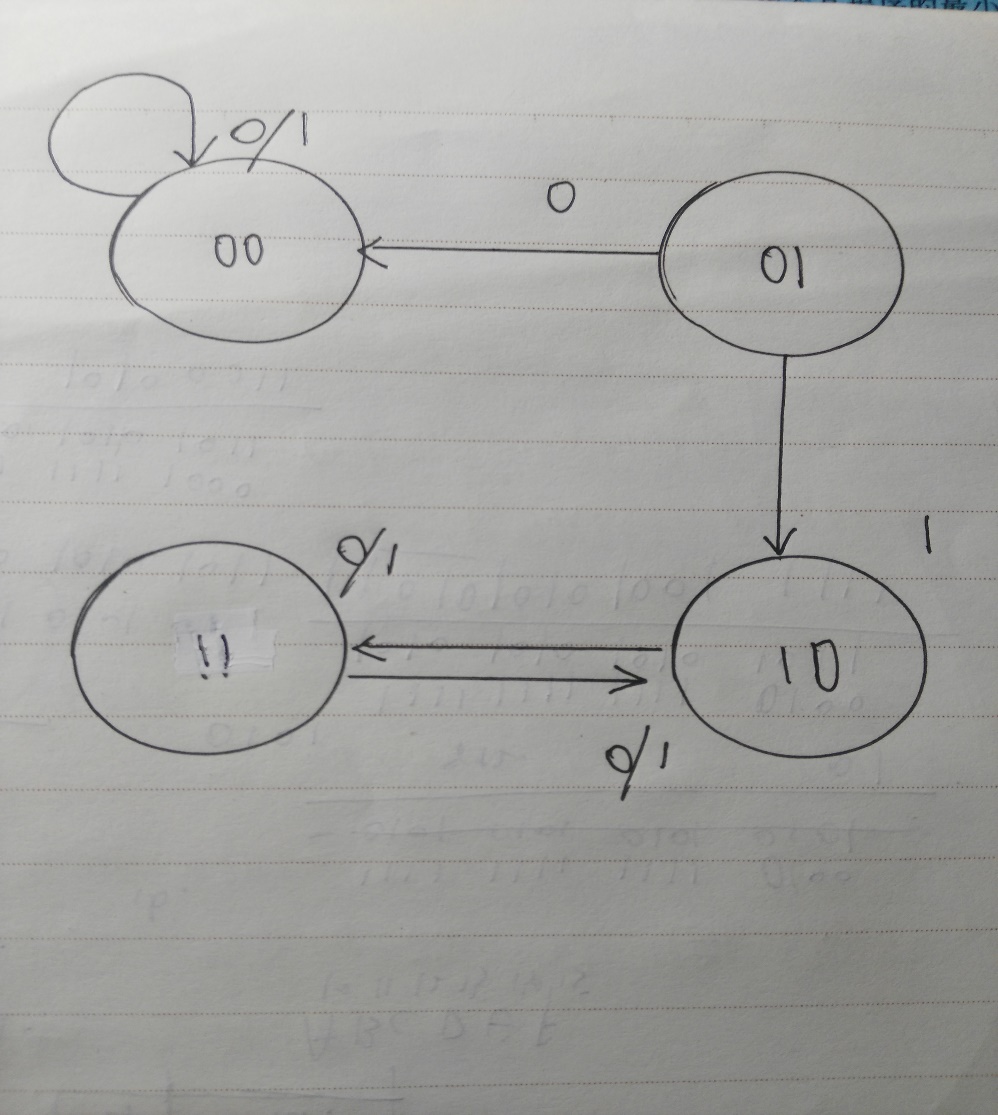
**a. Complete the rest of the following table.**

**SI, SO specifies the present state.**

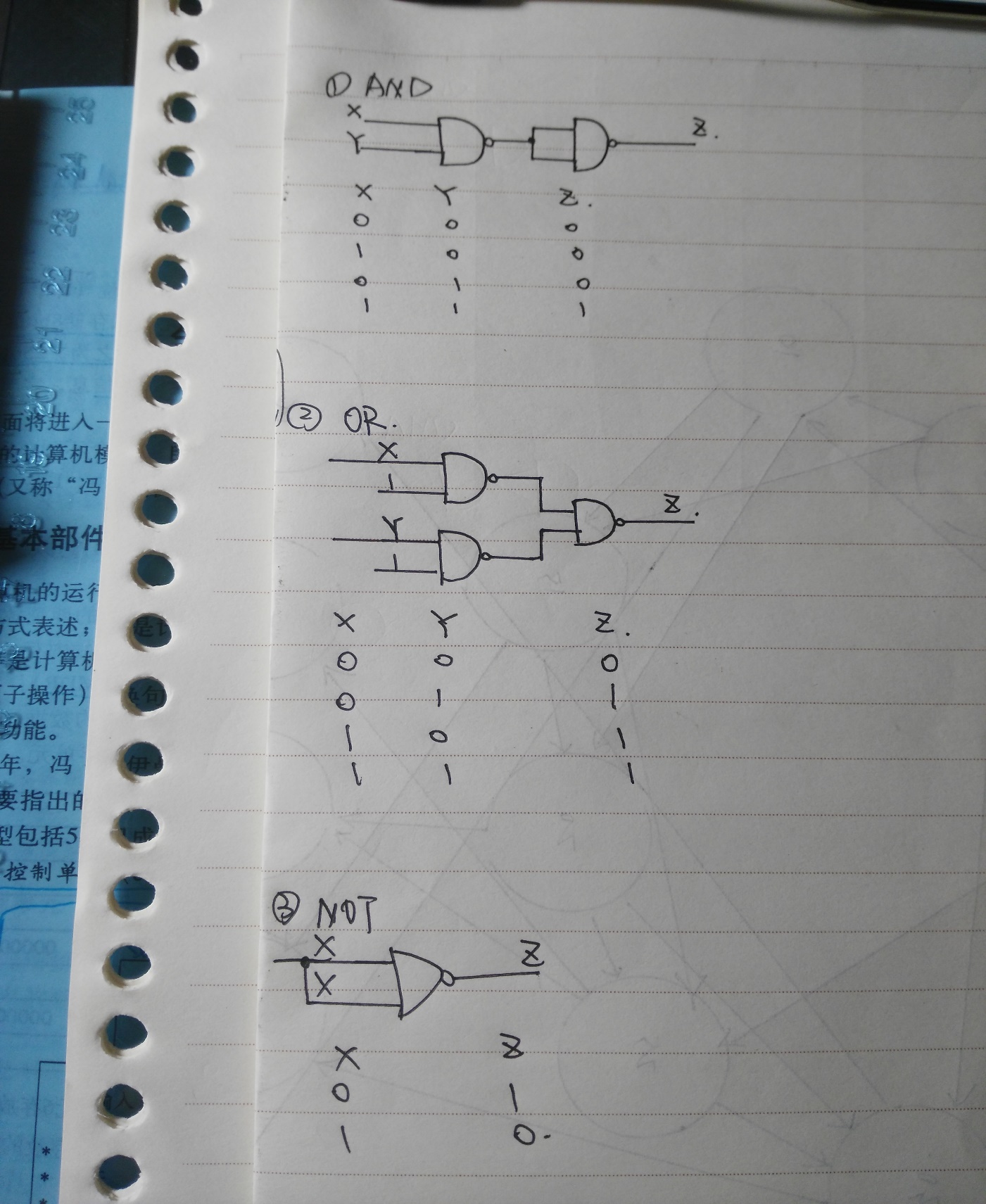
**DI, DO specifies the next state.**

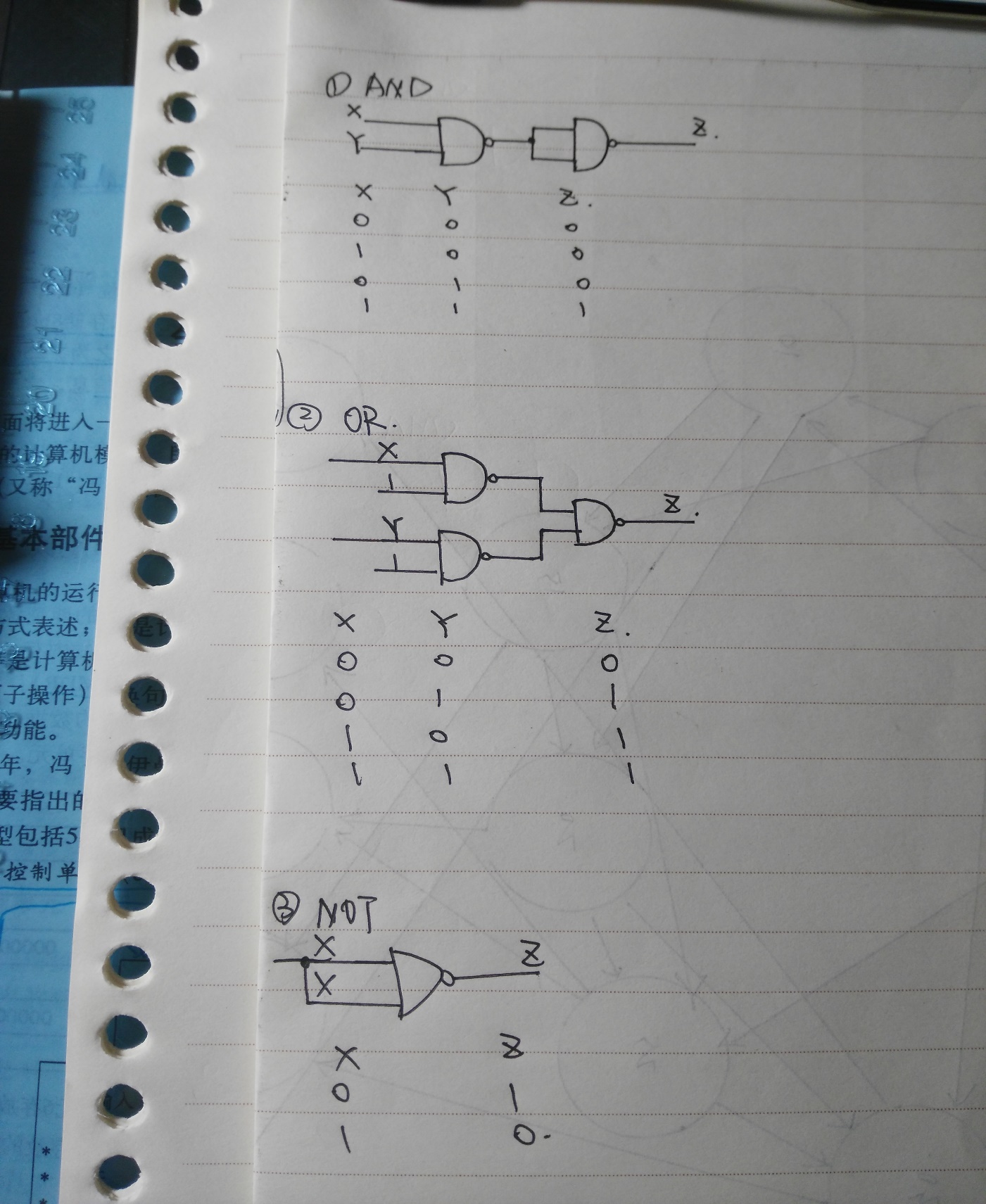
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S1 | S2 | X | D1 | D0 | Z |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |

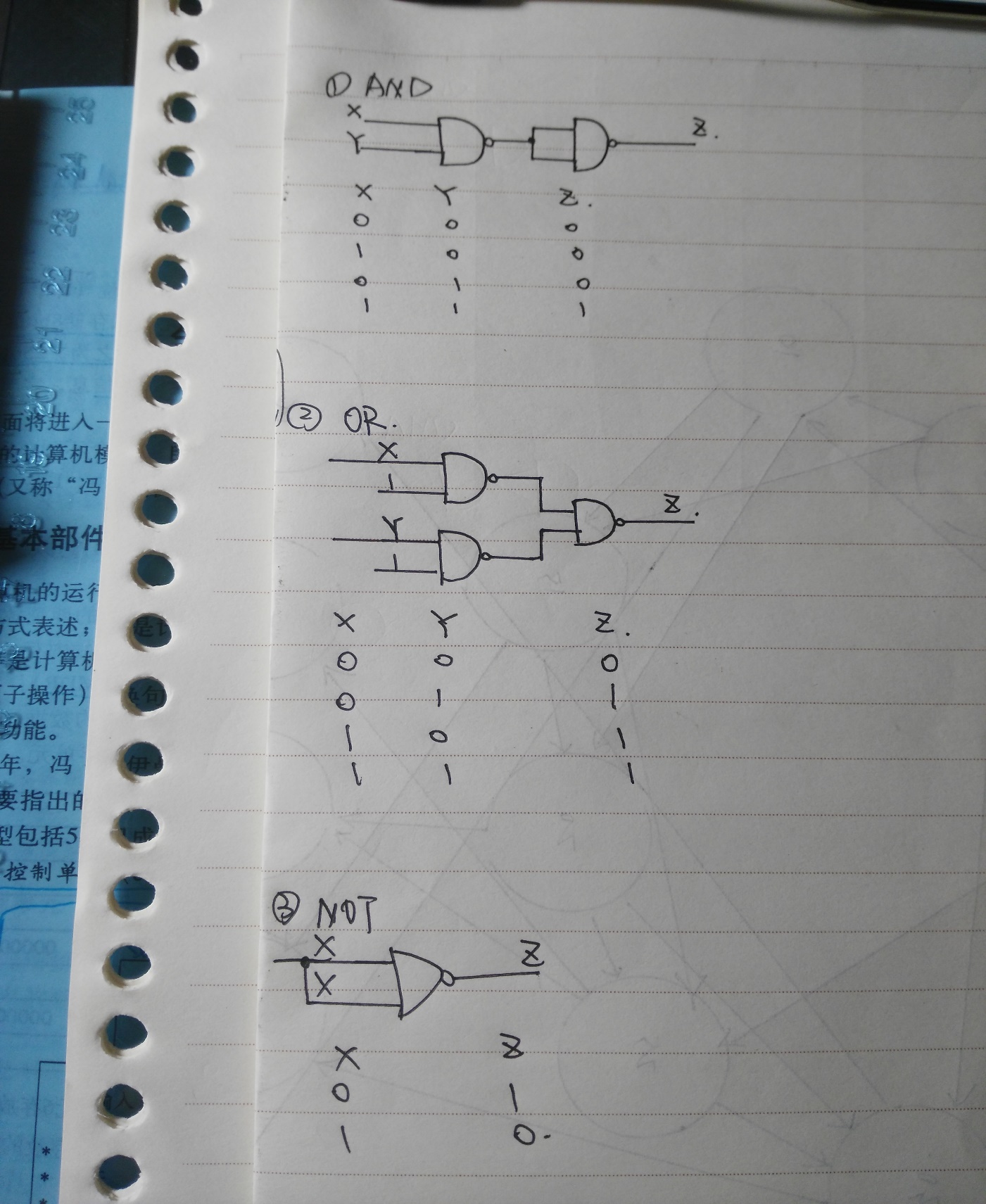
**b. Draw the state diagram for the truth table from part a.**

****

**3,44 Prove that the NAND gate, by itself, is logically complete (see Section 3.3.5) by constructing a logic circuit that performs the AND function, a logic circuit that performs the NOT function, and a logic circuit that performs the OR function. Use only NAND gates in these three logic circuits.**

****

****

****

**Chapter 5**

**5.34 Using the overall data path in Figure 5.18, identify the elements that implement the NOT instruction of Figure 5.4.**

Reg File, the IR, NZP and logic with it, ALU

**5.35 Using the overall data path in Figure 5.18, identify the elements that implement the ADD instruction of Figure 5.5.**

Reg File, the IR, SEXT unit, SR2MUX, NZP and logic with it, ALU