Computer Architecture (2024-2025) **Assignment** 1 November 24, 2024

George Theodoridis (S5951178)

INSTRUCTIONS

1. Submission is only via Brightspace. Deadlines are strict.
2. The exercises in this assignment add up to 100 points. To calculate your grade simply divide the number of points by 10.
3. You must submit a pdf typeset in (La)TeX (no handwritten solutions) using **this** template.
4. Seeking solutions from the internet, from any external resource, or from any other person is prohibited.
5. Please note that the course lecturer reserves the right to ask the student submitting the assignment to explain the answers to any or all questions. If the student is unable to provide a satisfactory answer then that question may receive partial/no credit.
6. Of course, university policies on plagiarism always apply. In particular, any suspected plagiarism will be reported to the Board of Examiners.

---

1. Convert the following decimal numbers to their 2's complement representations (use 8 bits for each number and truncate the fractional digits if necessary): **(10 points)**

   a) 42.3
   b) 6.3
   c) -13
   d) 0.04

Solution:

a) 00101010
b) 00000110
c) 11110011
d) 00000000

2. Perform the following operation in 2's complement. Indicate when an overflow happens. **(10 points)**

   a) 11001010 + 11101010
   b) 01011010 + 00110101
   c) 0011.1100 + 0100.0100
   d) 11010111 – 00001011

Solution:

a) 10110100 no overflow

b) 10001111 overflow

c) 1000.0000 overflow

d) 11001100 no overflow

3. Write IEEE floating point representation of the following decimal numbers. **(10 points)**

   a) 19.45

   b) -0.028

Solution:

a) 0 10000011 00110111001100110011001

b) 1 01111001 11001010110000001000010

4. The following numbers are in IEEE floating point representation. Write their decimal equivalents. **(10 points)**

   a) 1 01111001 10101010000000000000000

   b) 0 00000000 00001000000000000000000

Solution:

a) -0.0260009765625

b) $3.67341985 * 10^{-40}$

5. Design a circuit that outputs a 1 only if exactly two out of three inputs (A, B, C) are 1. (Use only NOT gates or logic gates with 2 inputs.) **(15 points)**

Solution:

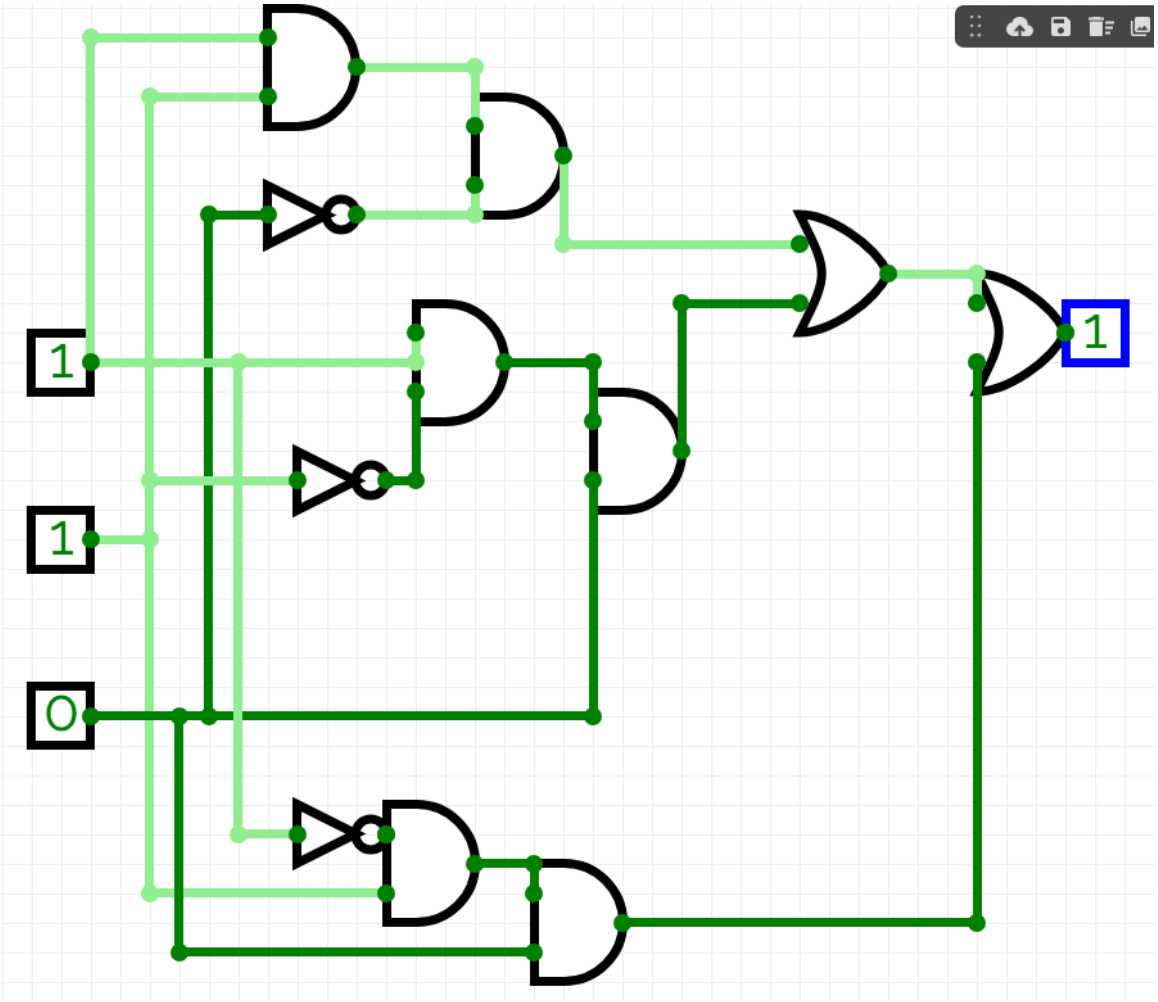

Figure 1: ex5

6. Using a decoder and an encoder, design a circular incrementor circuit for 4-bit positive binary numbers. The input (a number between 0 and 7) will be incremented by one if it is less than 7. If the input is 7, the output should become zero. **(15 points)**

Solution:

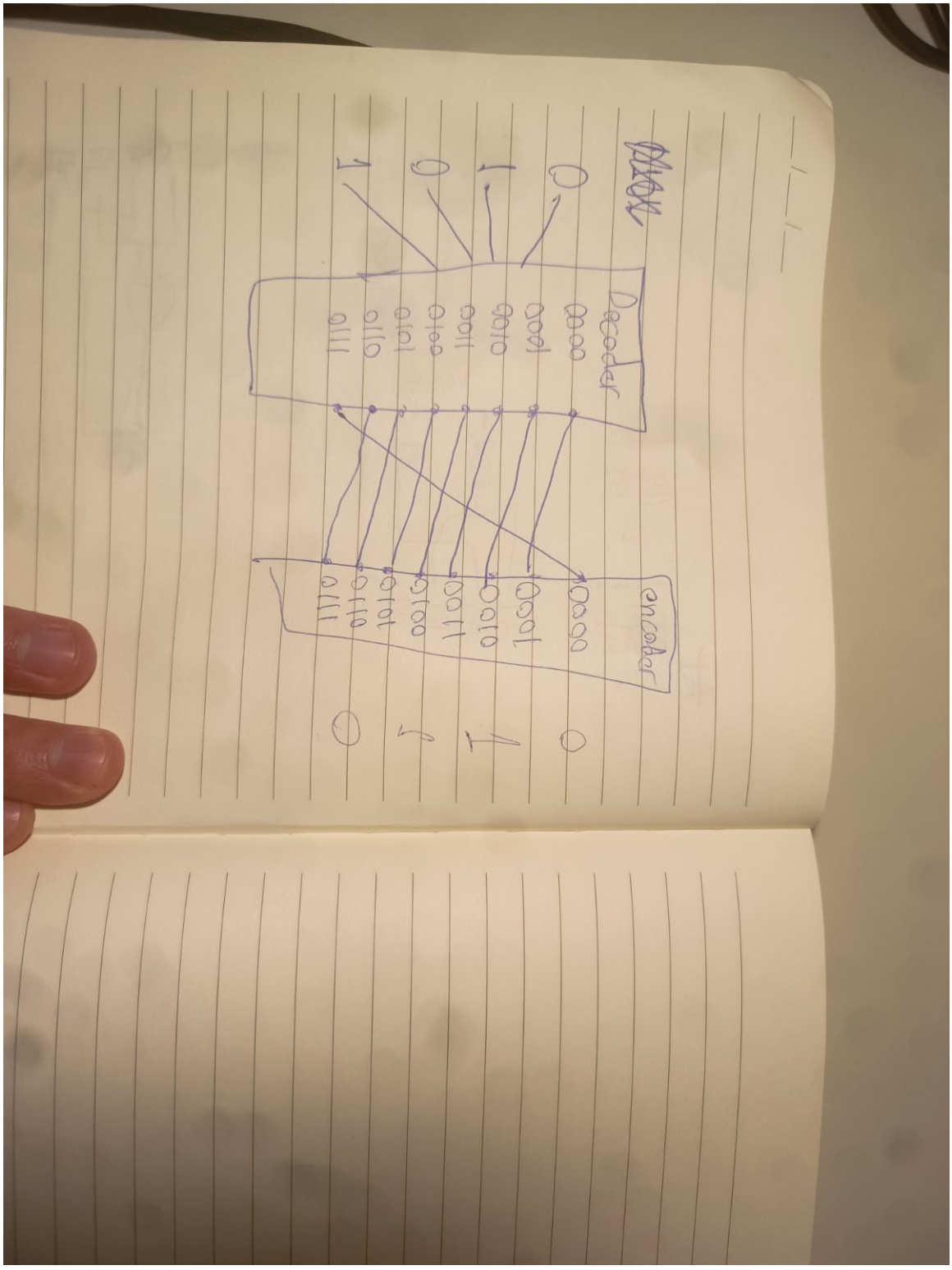
Decoder
0000
0001
0010
0011
0100
0101
0110
0111
encoder
0000
0001
0010
0011
0100
0101
0110
0111

Figure 2: ex6

7. Adding two numbers represented in scientific notation requires shifting floating points to make exponents equal before adding the fractions. **(15 points)**
   Consider the following example:

   - Start with: $1.34 \cdot 10^3 + 2.1 \cdot 10^{-1}$
   - Then, shift the decimal point of the smaller number to equalize the exponents: $1.34 \cdot 10^3 + 0.00021 \cdot 10^3$
   - Then add the fractions, and normalize the result if necessary: $1.34021 \cdot 10^3$

   Assume we have two numbers $N = 14.25$ $M = 0.5$

   a) Convert both numbers to IEEE floating point format.

   b) Add the numbers using the described procedure (include implicit 1).

   c) Normalize the result and convert it to IEEE format.

Solution:

a)

$$14.25 = 0\ 10000010\ 11001000000000000000000$$
$$0.5 = 0\ 01111110\ 00000000000000000000000$$

b)
- Start with $0\ 01111110\ 00000000000000000000000$.
- To convert 2^{-1} to 2^3, we need to move the point by 4 positions.
- The mantissa will become $0.00010000000000000000000$ (but without the leading 0).
- The exponent will match that of 14.25.

Hence, we need to add:

$$1.11001000000000000000000 + 0.00010000000000000000000 = 1.11011000000000000000000$$

Since the result is positive and the exponent remains the same, the IEEE format will be:

$$0\ 10000010\ 11011000000000000000000$$

8. Design a circuit using logic gates to compare two 2-bit positive numbers. The output should be: **(15 points)**
   - 10 if the first number is greater,
   - 01 if the second number is greater,
   - 00 if the two numbers are equal.

Solution:

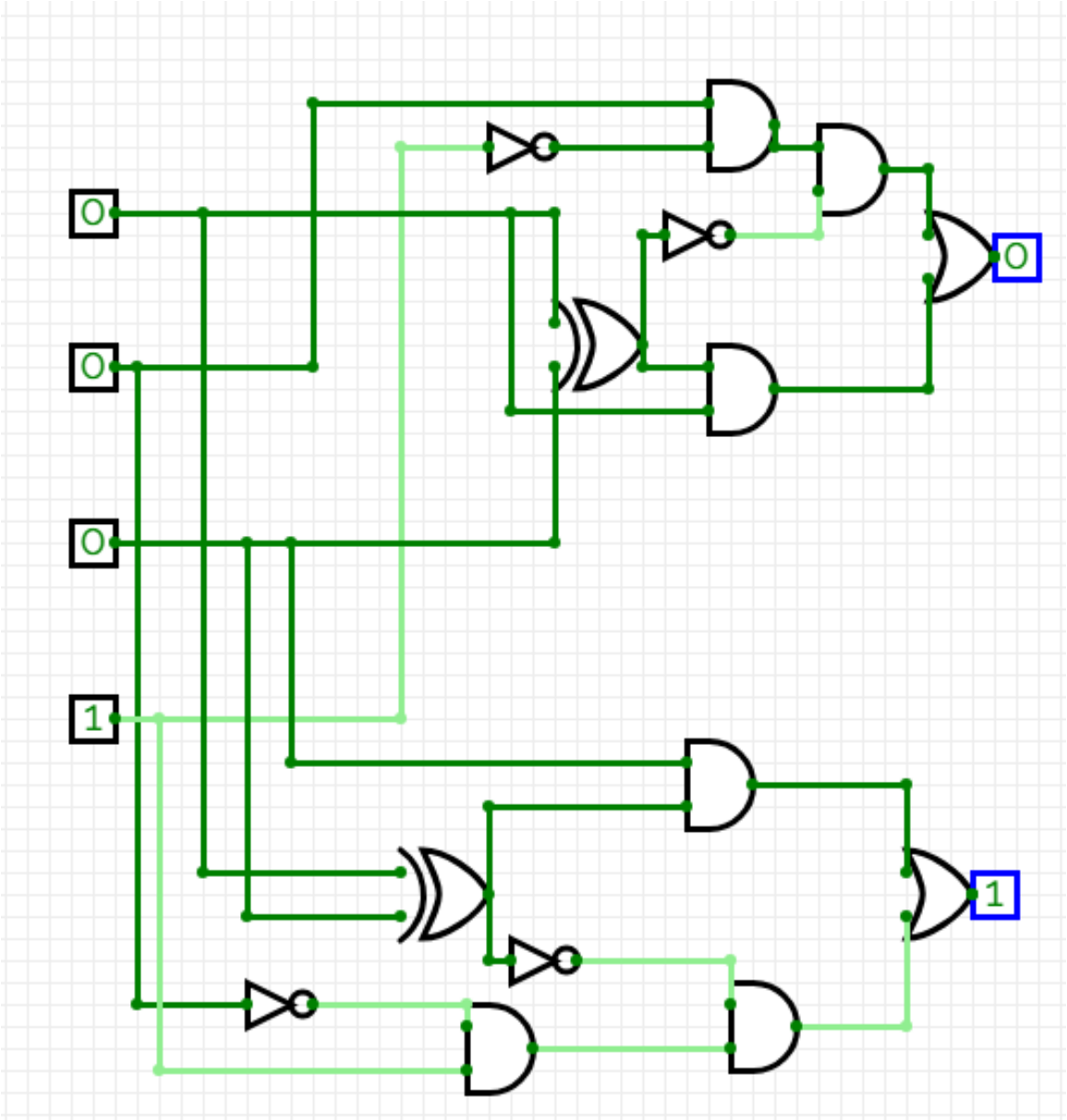

Figure 3: ex8

I split problem into the first bit and the second bit. the first output bit must be 1 only when the first number is greater. So i create a circuit for when the first number is greater. I start with xor because i split the task

into when the first bits are the same and when they are not. If they are not the same then I make the output 1 when the first is the biggest (i just NOT the second bit and then AND both of them). If they are the same then i check for the second bits the same way. In the end I OR the 2 results because in case one of them outputs 1, it means that the above number is bigger. Then the same for the second output bit