

Midterm Exam
CSC 252
7 March 2025
Computer Science Department
University of Rochester

Instructor: Yuhao Zhu

TAs: Jack Cashman, Ethan Chen, Weikai Lin, Jiaqi Nie, Chenrui Wang, Boyi Zhang

Name: _____

Problem 0 (2 points):

Problem 1 (12 points):

Problem 2 (21 points):

Problem 3 (27 points):

Problem 4 (28 points):

Total (90 points):

Remember “**I don’t know**” is given 15% partial credit, but you must erase everything else. This does not apply to extra credit questions (if any).

Your answers to all questions must be contained in the given boxes. Use spare space to show all supporting work to earn partial credit.

You have 75 minutes to work.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature: _____

Have a good spring break and GOOD LUCK!!!

Problem 0: Warm-up (2 Points)

What's the most surprising thing you've learned so far?

Problem 1: Fixed-Point Arithmetics (12 points)

Part a) (2 points) Represent decimal number 31 in the hexadecimal form.

Part b) (4 points) Represent octal (base 8) number 73 in the decimal form **and** binary form.

Part c) (6 points) Consider two signed binary numbers $A = 10101$ and $B = 11001$, both are represented using two's complement representation. Do the math below.

(2 points) What's the result of $\neg A \vee B$ (\neg is NOT and \vee is OR)?

(2 points) What's the result of $A \oplus B$ (\oplus is XOR)?

(2 points) How to write A in base 7?

Problem 2: Floating-Point Arithmetics (21 points)

Part a) (4 points)

(2 points) Write $10\frac{7}{8}$ using the normalized scientific notation.

(2 points) Write $32\frac{5}{16}$ using the normalized scientific notation.

Part b) (8 points) The engineering team is designing a new **12 bit floating point** standard. The format follows the principles of the IEEE-style floating point representation we discussed in the class. The engineering team is considering two possible standard:

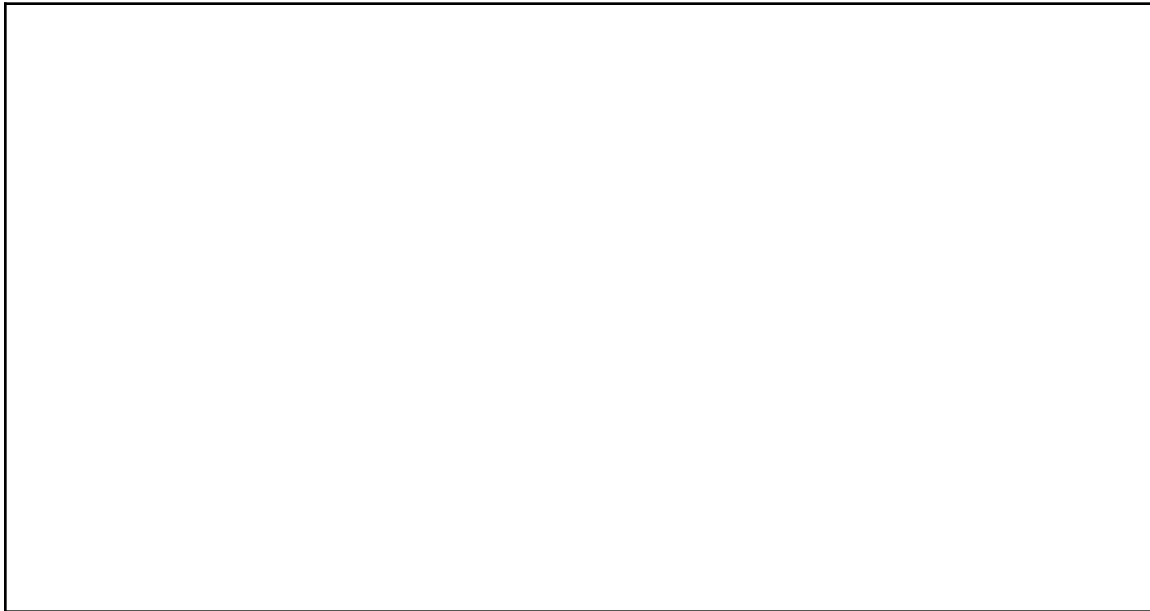
- Standard A: 6 fraction bits
- Standard B: 9 fraction bits

(4 points) What are the biases for Standard A and Standard B?

(4 points) Given the requirement that the smallest gap between two representable numbers is at most $\frac{1}{2^{24}}$, which of the two options meets the requirement? Show your work.

Part c) (9 points) Now the engineering team wants to design another 12-bit floating point representation. They want the representation to be able to precisely represent -100 and 100.

(5 points) What is the representation that meets this requirement while having the smallest gap between two consecutive, representable numbers?



(4 points) What is the smallest positive number that can be precisely represented in this format?



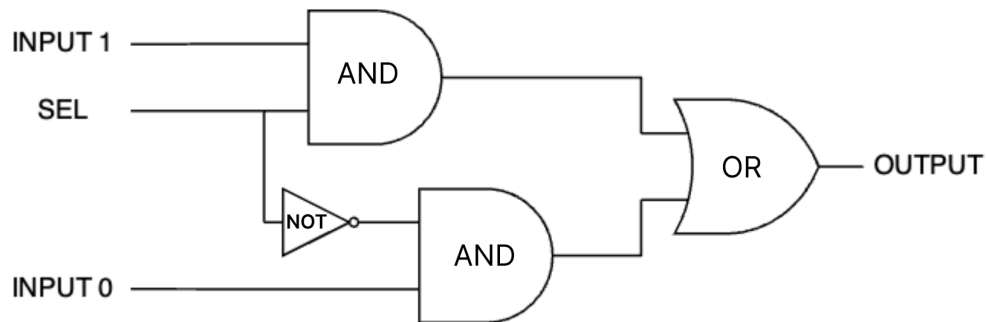
Problem 3: Logic Design (27 points)

Part a) (4 points)

(2 points) What is the result of a bitwise XOR operation between 01100 and 00111?

(2 points) What is the result of a bitwise NAND operation between 01000 and 01010?

Part b) (13 points)



(8 points) Given the circuit above, complete the following truth table. Additional columns are provided to show your partial work (for partial credit).

INPUT 0	INPUT 1	SEL	OUTPUT				
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

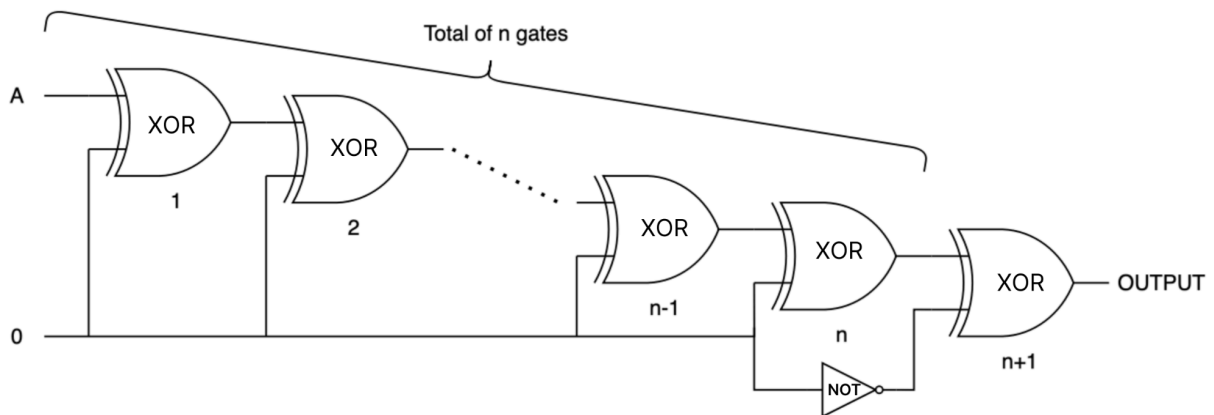
(2 points) Briefly describe the behaviour of the circuit above in words.

(3 points) Using the same circuit, it is possible to implement the equivalent of an AND gate. Complete the following table by assigning each of the three input wires to variables A, B or constants 0, 1 such that $\text{OUTPUT} = A \text{ AND } B$.

Input	Assigned variable/constant
INPUT 0	
INPUT 1	
SEL	

Part c) (8 points)

Consider the circuit below, which consists of $N+1$ XOR gates cascaded with a single NOT gate placed immediately before the $(N+1)^{\text{th}}$ XOR gate.



(6 points) What is the output of the circuit for the following values of N ? You can write the output as a function of A .

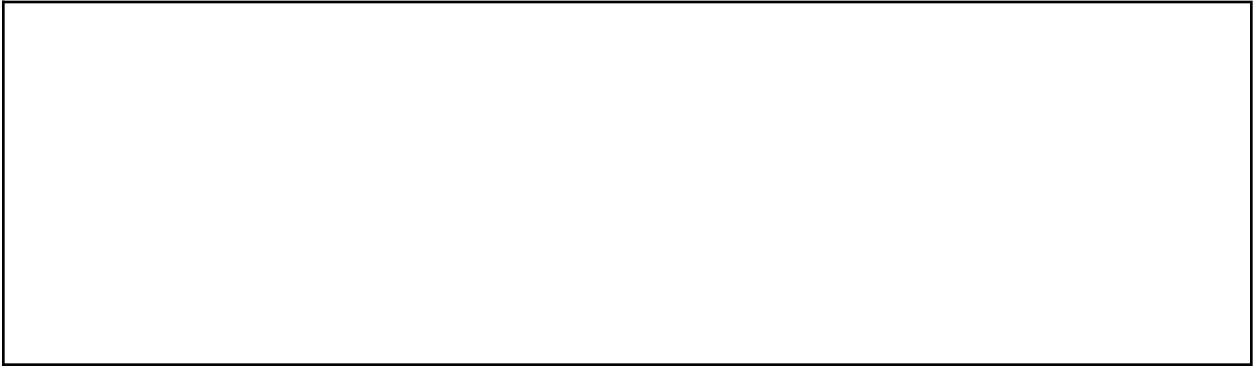
$N = 50$

A large, empty rectangular box with a black border, intended for drawing a circuit diagram.

$N = 63$

A large, empty rectangular box with a black border, intended for drawing a circuit diagram.

(4 points) Draw a simpler circuit that will achieve the same output as the circuit above.

A large, empty rectangular box with a black border, intended for drawing a circuit diagram.

Problem 4: Assembly Programming (28 points)

Conventions:

1. For this section, the assembly shown uses the AT&T/GAS syntax **opcode src, dst** for instructions with two arguments where **src** is the source argument and **dst** is the destination argument. For example, this means that **mov a, b** moves the value **a** into **b**.
2. All C code is compiled on a **64-bit machine**, where arrays grow toward higher addresses.
3. We use the x86 calling convention. That is, for functions that take two arguments, the first argument is stored in **%edi (%rdi)** and the second is stored in **%esi (%rsi)** at the time the function is called; the return value of a function is stored in **%eax (%rax)** at the time the function returns.
4. We use the **Little Endian** byte order when storing multi-byte variables in memory.

Part a) (10 points)

Consider the following function `fibonacci` and the assembly code that implements the C function.

```
int fibonacci(int n) {
    if (n <= 1)    return n;
    return  fibonacci(n - 1)  +  fibonacci(n - 2);
}
```

```
0000000000401106 <fibonacci>:
401106:    push    %rbp
401107:    mov     %rsp,%rbp
40110a:    push    %rbx
40110b:    sub     $0x18,%rsp
40110f:    mov     %_A_, -0x14(%rbp)
401112:    _B_     $0x1, -0x14(%rbp)
401116:    _C_     40111d <fibonacci+0x17>
401118:    mov     -0x14(%rbp), %eax
40111b:    jmp     40113b <fibonacci+0x35>
40111d:    mov     -0x14(%rbp), %eax
401120:    sub     $0x1, %eax
401123:    mov     %eax, %edi
401125:    call    401106 <fibonacci>
40112a:    mov     %eax, %ebx
40112c:    mov     _D_(%rbp), %eax
40112f:    _E_
401132:    mov     %eax, %edi
401134:    call    401106 <fibonacci>
401139:    add     %ebx, %eax
40113b:    mov     -0x8(%rbp), %rbx
40113f:    leave
401140:    ret
```


Your Task: Fill in the missing instructions A, B, C, D, and E to complete the assembly code.

(2 points) A:

(2 points) B:

(2 points) C:

(2 points) D:

(2 points) E:

Part b) (18 points)

Consider the following unknown x86_64 assembly function:

```
00005555555516d <unknown>:
0x5555555516d:      mov     $0x1,%eax
0x55555555172:      cmp     $0x1,%rsi
0x55555555176:      jbe     0x555555551aa
0x55555555178:      mov     %rdi,%rdx
0x5555555517b:      lea     -0x4(%rdi,%rsi,4),%r8
0x55555555180:      mov     $0x1,%r9d
0x55555555186:      mov     $0x0,%edi
0x5555555518b:      jmp     0x55555555199
0x5555555518d:      mov     %edi,%r9d
0x55555555190:      add     $0x4,%rdx
0x55555555194:      cmp     %r8,%rdx
0x55555555197:      je      0x555555551a7
0x55555555199:      mov     0x4(%rdx),%esi
0x5555555519c:      mov     (%rdx),%ecx
0x5555555519e:      cmp     %ecx,%esi
0x555555551a0:      jg      0x5555555518d
0x555555551a2:      cmovl    %edi,%eax
0x555555551a5:      jmp     0x55555555190
0x555555551a7:      or      %r9d,%eax
0x555555551aa:      ret
```

At the start of the function, the first argument is set to a pointer to the following integer array: [1, 1, 3, 2]. The second argument is set to 0x4.

The `cmovl` instruction conditionally moves a value from the source register to the destination register only if the sign flag is not equal to the overflow flag.

(3 points) Which line of assembly sets the condition codes for the `cmovl` operation on line 0x555555551a2?

(3 points) How many times is the jump at 0x555555551a5 taken?:

(3 points) How many times is the jump at 0x555555551a0 taken?

(3 points) Assume that the following line of assembly was executed immediately after the first execution of the instruction located at 0x5555555517b:

```
mov (%r8), %r9
```

What would the value in r9 be after this instruction?

(3 points) What is the meaning of the second argument of the function?

(3 points) Describe the function. What does it do?