# CIS 240 Fall 2021: Midterm

Oct 25, 2021

Name :

_____

Please write your name on the exam. You can answer the questions on this exam sheet in the space provided. Please put your initials at the top of each page in case the pages become separated.

Please read the following pledge and sign in the space below:

*I neither cheated myself nor helped anyone cheat on this exam*

Sign Here :

**Do not write in the table below:**

| 1 | 2.1 | 2.2 | 2.3 | 3 | 4.1 | 4.2 | 5 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

**Question 1 {10 pts}**

Your job is to design a proper CMOS circuit which has 3 inputs, A, B and C and where the output is 1 if and only if exactly one of these 3 inputs is high. You can assume that you also have access to negated versions of all the input signals. Your solution must be a single CMOS circuit consisting of complementary pull up and pull down transistor networks. Please label the inputs and outputs of your circuit clearly on your schematic.

**Answer:**

**Question 2**

In class we discussed the fact that adding 2C numbers can result in arithmetic overflow which can cause the final result to be horrendously incorrect. Some computers try to improve on this state of affairs by implementing **saturating arithmetic**. When the result of an addition would be larger than the largest positive n-bit 2C value the saturating adder would produce this maximum n-bit 2C value, similarly if the result would be more negative than the most negative n-bit 2C value the saturating adder produces the most negative n-bit 2C value. The result of the addition still isn't correct, but it's the closest answer that could be produced given the constraints of the 2C representation. In this question you will design such a saturating adder. Please help the grader by clearly labeling the inputs and outputs of each of the circuits you produce. Feel free to add comments to explain your designs

**Part 1 {10 pts}:** Produce a gate level circuit that produces a 1 if and only if the addition of two 5-bit 2C numbers, A and B, would result in an overflow situation. That is a situation where the correct result would be outside the range of numbers that can be represented in 5-bit 2C format. Your circuit can use any of the following signals as inputs: MSB_A – Most significant Bit of input number A, MSB_B – Most significant Bit of input number B, MSB_SUM – Most significant bit of the n-bit sum produced by a standard adder circuit.

**Answer:**

**Part 2 {10 pts}:** Produce a gate level circuit that produces an 5-bit output corresponding to the output that the saturating adder should produce if an overflow condition is detected when the input numbers, A and B, are added. Your circuit can use any of the following signals as input: MSB_A – Most significant Bit of input number A, MSB_B – Most significant Bit of input number B, MSB_SUM – Most significant bit of the n-bit sum produced by a standard adder circuit.

**Answer:**

**Part 3 {10 pts}:** Using your answers to part 1 and part 2 as modules, produce a schematic showing how you would implement the desired saturating adder that takes the 5-bit inputs, A and B, and produces the 5-bit result of the saturating addition. Note you can also use standard 5-bit adder circuits and 5-bit multiplexers as modules in your circuit. Clearly label each module in your circuit along with any inputs, outputs, or relevant internal signals.

**Answer:**

**Question 3 {10 pts}**

Having taken CIS 240 you know that common arithmetic operations like addition and multiplication can lead to arithmetic overflow. Can division cause overflow? other than the case of division by zero where the result isn't defined can the DIV instruction in LC4 result in an arithmetic overflow of any kind? Explain your answer briefly below, just saying yes or no won't get you many points.

**Answer:**

**Question 4**

Consider the assembly program shown below.

```
        .CODE               ; This is a code segment
        .ADDR  0x0000       ; Start filling in instructions at address 0x00

        CONST R2, #0        ; Initialize C to 0

LOOP    CMPI R1, #0         ; Compare B to 0
        BRnz END            ; if (B <= 0) Branch to the end

        ADD R2, R2, R0      ; C = C + A
        ADD R1, R1, #-1     ; B =  B - 1

        BRnzp LOOP ; Go back to the beginning of the loop

END     NOP
```

**Part 1 {10 pts}**

Use the table below to show what the contents of memory would be once this program is assembled and loaded into PennSim. Each entry should be a 16- bit binary value. We have provided the first entry as a template.

**Answer:**

| Memory Address | Contents (in binary) |
|---|---|
| 0x0000 | 1001010000000000 |
| 0x0001 | |
| 0x0002 | |
| 0x0003 | |
| 0x0004 | |
| 0x0005 | |
| 0x0006 | |

**Part 2 {20 pts}**

Each column below shows the state of all registers at the start of each instruction cycle. Simulate the action of the assembly program by filling in the rest of the entries in this table Fill in the NZP entries with N for negative, Z for zero and P for positive. You must fill in the PC and NZP for every clock cycle, for the register values R0-R7 you only need to fill in the values of the registers that have changed from the previous cycle, if any. You should enter all values as decimal numbers, not hex.

| Instruction Cycle | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PC | 0 | | | | | | | | | | | | | | |
| NZP | P | | | | | | | | | | | | | | |
| R0 | -3 | | | | | | | | | | | | | | |
| R1 | 4 | | | | | | | | | | | | | | |
| R2 | 17 | | | | | | | | | | | | | | |
| R3 | 0 | | | | | | | | | | | | | | |
| R4 | 0 | | | | | | | | | | | | | | |
| R5 | 0 | | | | | | | | | | | | | | |
| R6 | 0 | | | | | | | | | | | | | | |
| R7 | 0 | | | | | | | | | | | | | | |

**Question 5**

Your job in this question is to design part of the Decoder circuit for our Single Cycle LC4 implementation.

a) **{10 pts}** Produce a PLA circuit that takes as input the relevant bits from the current instruction and produces as output the rdMux.CTL signal.
b) **{10 pts}** Produce a second circuit that generates the rtMux.CTL signal.

Please use the convention $I_{15}$, $I_{14}$, … ,$I_0$ to refer to bits in the instruction word where $I_{15}$ is the MSB and $I_0$ the LSB.

**Answer {part a}:**

**Answer {part b}:**