CIS 240 Fall 2018: Midterm

Oct 31, 2018

Name : ____

Please write your name on the exam **and** the exam booklet and turn in both. You can answer the questions on this exam sheet or in the exam booklet. Please number the questions you are addressing clearly in the exam booklet.

Question 1 {20 pts}

The table below shows the contents of a region of User Code memory in PennSim. First convert the machine instructions you see here to an equivalent sequence of assembly instructions so you can read them. Note the assembly program on this answer sheet or in your test booklet. After you have done this, show what would happen when the program is executed by filling in the second table which shows the state of all of the registers at the **start** of each instruction cycle. Fill in the NZP entries with N for negative, Z for zero and P for positive. For the register values R0-R7 you only need to fill in the value of the register that has changed from the previous cycle, if any. You should enter all values as decimal numbers, not hex.

Hint: start by looking at the four bit opcode – be careful how you break up the 16 bit fields, one bit can make a big difference.

Address	Machine Instruction	Assembly Instruction
0	1001001111111110	CONST R1, #-2
1	0001000000001001	MUL R0, R0, R1
2	0000100000000011	BRn 3
3	0010000100001010	CMPI R0 #10
4	0000100111111100	BRn #-4
5	0000111111111111	BRnzp #-1
6	0010000101110110	CMPI R0, #-10
7	0000001111111001	BRp #-7
8	0000111111111111	BRnzp #-1

Instruction Cycle	0	1	2	3	4	5	6	7	8	9	10	11
PC	0	1	2	6	7	1	2	3	4	5	5	5
NZP	P	N	N		P		P		P			
R0	3		-6				12					
R1	0	-2										
R2	0											

R3	0											
R4	0											
R5	0											
R6	0											
R7	0											

Question 2 {5 pts}

In a given C program the variables x, y, z and sum are all declared as doubles. Can you assume that these two C statements:

sum = (x + y) + z;

and

sum = x + (y + z);

always yield the same value for sum? Explain your answer, just saying yes or no won't earn you many points.

Answer

Not always, these two statements could yield different results because of the **rounding** inherent in floating point operations. This would typically be implemented with two floating point additions in the first case you add x and y first then z in the second you add y and z first then x. Consider the following examples

(1e200 + (-1e200)) + 1e-30 = This would probably produce a result around 1e-30
Since the first addition would result in a zero.

This sequence

1e200 + ((-1e200) + 1e-30) would probably produce a result of 0 since the inner addition would produce a result of around 1e-200 because of rounding effects.

Question 3 {5 pts}

True or false, can the absolute value every n-bit 2C number be contained in an n-bit unsigned number? Please explain your answer, simply answering true or false won't get you many points. (Remember the absolute value of a signed number is simply it's magnitude, egs. abs(-7) = 7, abs(23) = 23)

Answer

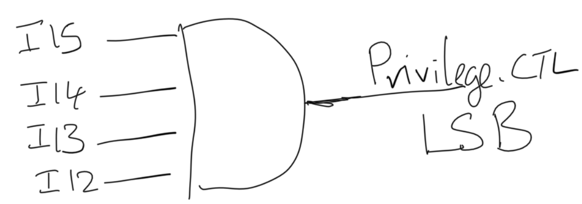
True, the largest magnitude of an n-bit 2C number is 2^(n-1) this is achieved by the most negative possible value. This number can readily be represented as an n bit unsigned number. It is simply a 1 followed by (n-1) zeros in binary.

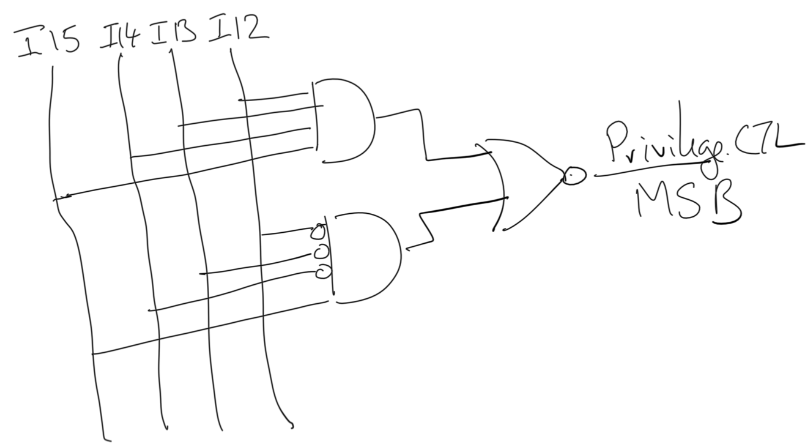
Question 4 {10 pts}

In the LC4 single cycle implementation that we have studied the Decoder block is responsible for generating all of the control signals required to execute the current instruction. For this question you are asked to design a small portion of this circuit. Specifically, you are asked to design a circuit that takes bits from the current instruction as input and generates the two bit Privilege.CTL signal as output. Please indicate which of your 2 output bits is the MSB and which the LSB. Please use the convention I15, I14, … ,I0 to refer to bits in the instruction word where I15 is the MSB and I0 the LSB. More points will be given for simpler solutions.

Answer

Here we observe that the Privilege.CTL signal should be 10_2 for all instructions except TRAP and RTI where it should be 01_2 and 00_2 respectively. We can conclude then that the MSB should be 1 unless the insn is TRAP or RTI and the LSB should be 0 unless the instruction is TRAP. That leads to the following solution which examines the opcode bits I15..I12. You could use the output of the LSB logic as an input to the MSB logic.





Question 5 {10 pts}

Design a **PLA** circuit that takes as input a 4 bit 2C value and returns a logical 1 when that input is a **non-zero** multiple of 4. Label your input bits I3 thru I0 where I3 is the MSB and I0 is the LSB

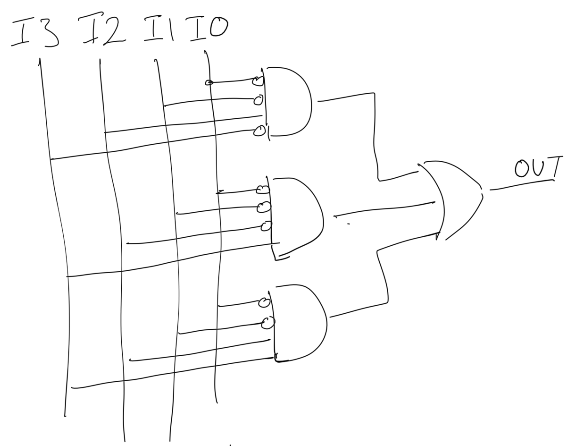
Extra Credit {2 pts}.

If you are not constrained to a PLA structure you can actually implement this function using no more than 3 two input gates (AND, OR, NAND, NOR, XOR, XNOR). Can you find such a solution?

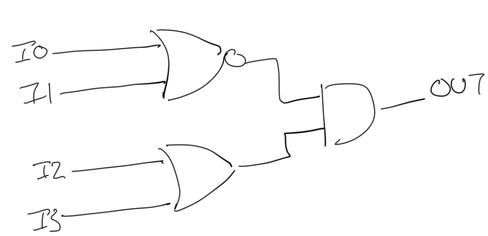
Answer

First we should consider in which cases the output should be 1. The three values are 4, -4 and -8. The corresponding bit patterns are 0100, 1100, 1000. From there you can synthesize the requisite PLA.

The extra credit solution looks for situations where the lower 2 bits are both 0 and either of the upper two bits is on. Note we can check for situations where both lower bits are 0. by employing DeMorgan's law and using a NOR gate which is on the list of legal gates that was provided.



Extra Credit



Question 6 {10 pts}

One of the great things about 2C representation is that we are able to use exactly the same circuit to add both unsigned and 2C values. In effect the addition circuit does not know or care whether the user thinks of the inputs as unsigned or 2C since the same algorithm is applied in both cases. Is it possible to design a single circuit that would be able to correctly detect **arithmetic** overflow for both 2C and unsigned addition in a similar manner? That is a circuit that would be able to properly detect arithmetic overflow when we perform 2C or unsigned addition without any additional inputs. Explain your answer, just saying yes or no won't earn many points. Remember that arithmetic overflow refers to a situation where the output value of the addition circuit is incorrect.

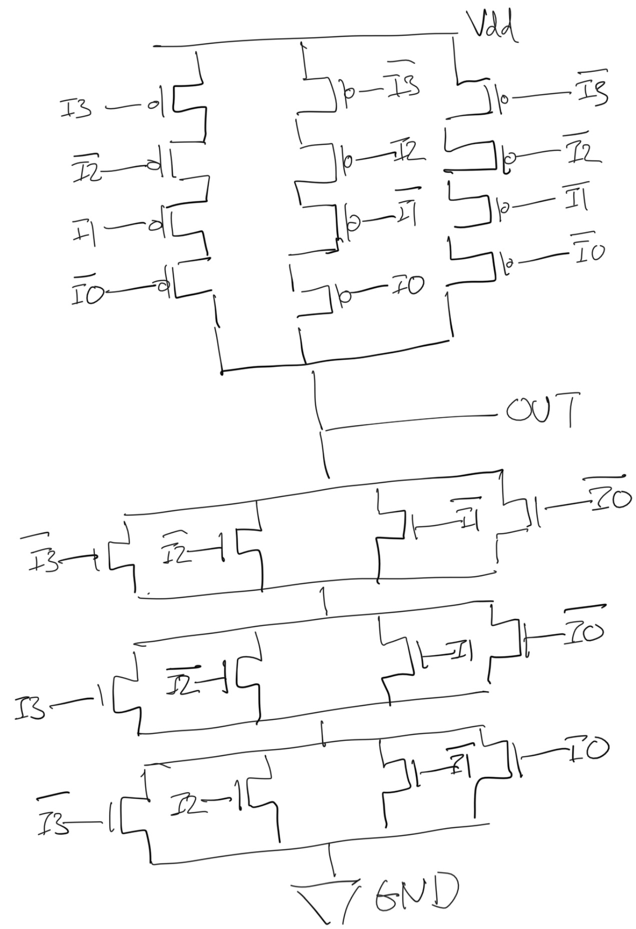
Answer:

This is not possible because arithmetic overflow means different things for 2C and unsigned addition. For unsigned arithmetic any operation that causes a CarryOut from the most significant position is a problem. For 2C addition this isn't always a problem, for example you may well add two 2C values like -1 and +2 which will generate a CarryOut but the answer is still correct. Basically this is not possible because unlike addition, the overflow detection function is different for unsigned and 2C so you can't have one circuit that can handle both cases.

Question 7 {10 pts}

Design a proper CMOS circuit that takes a 4 bit unsigned number as input and produces a High(1) output if the number is a **non-zero** multiple of 5. You should refer to the bits of the input as I3 thru I0 where I3 is the MSB and I0 is the LSB. You can assume that you are also provided with the negated versions of all of these inputs. Please produce a neat, well-labeled diagram – we can't grade what we can't read. Your circuit should only have one pull down and one pull up network – you should <u>not</u> be cascading multiple gates to get the desired behavior.

Answer:
We want the output of the circuit to be high if the input value is 5, 10 or 15. One way to approach the design is by designing the pull up network first but to do this we should remember that the pmos transistors are active if the input is LOW which means we need to use the inverted versions of the inputs as shown below. Once you have the pull up network you just need to make sure that the pull down network is complementary to that.



Question 8 {10 pts}

Attached to this exam you will find a diagram depicting an implementation of the LC4 ISA. You will also find another sheet listing all of the control signals in that implementation and what happens when they are set to different values.

Your job is to fill in the table below to show precisely how those control signals should be set to execute each of the following LC4 instructions. You <u>must</u> use X's to denote situations where you don't care what the control signal is.

	PCMux.CTL	rsMux.CTL	rtMux.CTL	rdMux.CTL	regFile.WE	regInputMux.CTL	NZP.WE	DATA.WE	Privilege.CTL	ALUInputMux.CTL	ALU.CTL
TRAP	4	X	X	1	1	2	1	0	1	X	X
CMPU	1	2	0	X	0	0	1	0	2	0	17
JMPR	3	0	X	X	0	X	0	0	2	X	X