11/07/2023

Noah Semashkewich

COSC 211 Lab 5/6 Written

1) a) 3.6: 185 - 122 = 63

63 is within range of 0 and 255 so there is neither overflow or underflow.

b) 3.20: First convert hexadecimal representation to binary

0x0C000000 = 0000 1100 0000 0000 0000 0000 0000 0000

Then invert bits to get two's complement

= 1111 0011 1111 1111 1111 1111 1111 1111 + 1

= 1111 0100 0000 0000 0000 0000 0000 0000

Now convert to decimal

= $\boxed{201326592}$

c) 3.21: To determine RISC-V instruction executed when the bit pattern 0x0000006F is placed into instruction register, you would need RISC-V ISA.

d) 3.22: To represent bit pattern 0x0C000000 as a floating point number using IEEE 754

0000 1100 0000 0000 0000 0000 0000 0000

1. Sign bit (bit 31): 0 (positive)
2. Exponent bits (bit 30-23): 00001100
3. Fraction bits (bits 22-0): 0000 0000 0000 0000 0000 000

0000 1100 = 12 in decimal

In IEEE 754, exponent is biased by 127

12 - 127 = -115

Fraction bits represent 0

So...

$(-1)^0 (2)^{-115} (0) \boxed{=0}$

e) 3.23: To represent number 63.25 in IEEE 754 format

1. Sign bit: Sign bit is 0 since 63.25 is positive
2. Exponent bit: 63.25 in binary is 111111.01, so
   $= 1.11111101 \times 2^5$
   5 + 127 = 132
   132 = 10000100
3. Fraction bits: Fraction bits after normalization is 0.11111101 then add 0's

= 11111010000000000000000

Final Representation

$\boxed{0\,10000100\,11111010000\,0000\,0000\,0000}$

2 a) $111111.01_2 = 63.25_{10}$

1. Sign bit: 0 since num is positive
2. Exponent bit: 1.1111101×2^5

$5 + 2^7 - 1 = 132$

$132 = 10000100$

So exponent is 10000100

So full floating point representation for 111111.01_2 is

$\boxed{01000010011111010000000000000000}$

b) 100011111000101011.01101

$= 146987.40625_{10}$

$(1.00011111000101011.01101) \times 2^{17}$

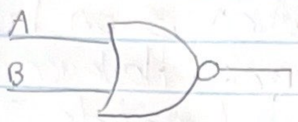
$\underline{5 + 2^{17} - 1 = 144}$

$144 = 10010000$

So full fp representation

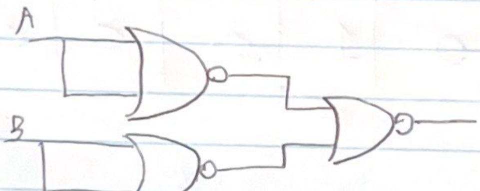
$\boxed{010010000001111100010101101101 0}$

③ NOR Gate

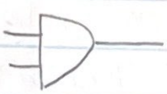


$Y = \overline{A + B}$

Using two input, 3 NOR gates, we can create an AND gate

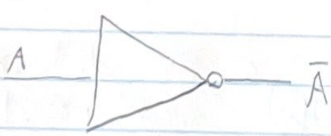


$Y = \overline{\overline{A} + \overline{B}} = \overline{\overline{A}} \cdot \overline{\overline{B}} = A \cdot B$



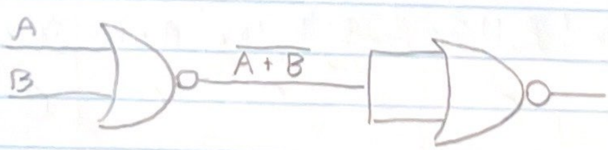
$Y = A \cdot B$

Using two input, 1 NOR gate, we can create a NOT gate



$Y = \overline{A + A} = \overline{A}$

Using two input, 2 NOR gates, we can create an OR gate



$Y = \overline{\overline{A + B}} = A + B$

(4) There will be no change to the datapath diagram because the subi instruction is an I-type instruction, which the datapath can handle. However there will be a change in the control signals. The added row of control signals for subi is:

instruction	RegDst	ALUSrc	MemtoReg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
subi	0	1	0	1	0	0	0	0	1

- RegDst is rt
- ALUSrc is 32 bit sign extended offset
- MemtoReg, Reg-Write, MemRead and Mem-write are all 0 because subi doesn't involve memory
- Branch is 0 because subi is not a branch
- ALUOp0 is 1 to perform subtraction.

(5) There will be a change in the datapath diagram. A new control signal called "branch" will be added to the control unit. This will be connected to a multiplexor. The Branch signal will be set to 1 for the "bne" instruction and 0 for all other instructions. The updated table of control signals is:

instruction	RegDst	ALU Src	MemToReg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
bne	X	0	X	0	0	0	1	0	0

Noah Semashkewich