CS2100 (AY2018/9 Semester 2)
Assignment #1

/15

Name: Daniel Alfred Widjaja	Tutorial Grp: 20

You are to do this assignment **on your own**. (Students found copying will be penalised.) Please fill in your **name** and **tutorial group number** in the boxes above, and your answers in the space indicated below. You are not required to show workings.

Please submit this assignment by **15 February 2019, Friday, 23:59** to the submission File on LumiNUS according to your tutorial group**.** Please submit either a .docx or .pdf file. **Late submission and email submission will not be accepted**.

1. Although C language does not have the **NOR** operator, it can be emulated as follows:

```
int NOR(int x, int y) {
  return !(x || y);
}
```

Consider the truth tables of **NOR**, **AND**, and **OR** below (from Lecture #7, Slide 35):

X	Y	X NOR Y	X AND Y	X OR Y
0	0	**1**	**0**	**0**
0	1	**0**	**0**	**1**
1	0	**0**	**0**	**1**
1	1	**0**	**1**	**1**

Suppose the **NOR** operator has been implemented as a **NOR** function. We can then create **NOT**, **OR** and **AND** using solely the **NOR** function without using the operators corresponding to each of those (i.e. for logical operation, **!** for **NOT**, **|** for **OR**, and **&** for **AND**).

Fill in the code below (the code for NOT has been filled in to provide an example). Note that you can only use the **NOR** function above as well as any constants. You are not to use any other functions or operators. Assume the parameters x and y take in values 0 or 1 only.

```
int NOT(int x) {
  return NOR(x, 0); // alternatively: return NOR(x, x);
}

int AND(int x, int y) {
  return NOR(NOR(x, 0), NOR(y, 0));
```

[1 Mark]

```
}
int OR(int x, int y) {
  return NOR(NOR(x, y), 0);
}
```
[1 Mark]

2. How do you represent **-2100** in the IEEE 754 single-precision floating-point representation? Write your answer in the *hexadecimal* form.

   Answer: `0xC5034000` [3 Marks]

3. Certain numbers cannot be represented in *decimal* number system unless we use repeating decimal. For instance, the number $\frac{1}{3}$ in *decimal* will be $(0.333333\ldots)_{10}$. Using the repeating decimal representation, we can then write it as $(0.\overline{3})_{10}$ since the 3s are repeating.
   Similarly, you have numbers in *binary* number system that cannot be represented unless repeating decimal is used. For instance, the number $(0.2)_{10}$ in *binary* will be $(0.001100110011\ldots)_2$. In repeating decimal, it will be $(0.\overline{0011})_2$ because 0011s are repeating.

   (a) What is $(0.7)_{10}$ in *binary*? Write your answer using repeating decimal.

   Answer: $(0.1\overline{0110})_2$ [1 Mark])

   (b) What is $(0.0\overline{1001})_2$ in *decimal*? Note that only 1001s are repeating.
   Hint: *Perform expansion up to some number and approximate.*

   Answer: $(0.3)_{10}$ [2 Marks]

For Questions 4-6, refer to the MIPS code below. The code reads an integer array ***A*** and modifies the array. Consider the variable to register mapping that maps variable ***A*** to register **$s0**. Assume that the array is initialized with `int A[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };` at the start.

```
      addi $t0, $s0,  0
      addi $t1, $s0, 40
Top:  lw   $t2, 0($t0)
      lw   $t3, 0($t1)
      sw   $t2, 0($t1)
      sw   $t3, 0($t0)
      addi $t0, $t0,  4
      addi $t1, $t1, -4
      beq  $t0, $t1, Bot
      j    Top
Bot:
```

4. Assume that the array starts at address **0x2100AB10**.

   (a) How many elements of array ***A*** are NOT loaded to either **$t2** or **$t3**?
   Answer: 1 [1 Mark]

   (b) What are the final values in array ***A*** after the program finished execution?
   Answer: { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 } [1 Mark]

5. Assume that the first instruction `addi $t0, $s0, 0` is at address **0x00000070**.

   (a) What is the *hexadecimal* representation of the `beq $t0, $t1, Bot` instruction?
   Answer: 0x11090001 [2 Marks]

   (b) What is the immediate value, in *decimal*, in the `j Top` instruction?
   Answer: 30 [2 Marks]

6. How many **addi** instructions are performed during the execution of the program?
   Answer: 12 [1 Mark]