- REPORT -

Name: **Đào Minh Châu**

ID: **24560025**

1. Classwork: Answer all question in the instruction.

Question 2.1.

The basic instructions which are converted from the original code are:

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x01404827	nor $9,$10,$0	1: not $t1, $t2
	0x00400004	0x20010002	addi $1,$0,0x00000002	2: subi $t1, $t1, 2
	0x00400008	0x01214822	sub $9,$9,$1	

-The values at $t10 and $t0 is compared by NOR(which is NOT OR) command

-The values is compared by the command OR and since both of the values are 0x00000000, the result will be 00000…000

- After that, the command NOT inverts the result to 1111111…111 and then the result is converted into hexadecimal 0xffffffff and stored in $t9.

-The addi command uses the value stored in $zero($0) and add it to 0x00000002, then store the result to $at(which is $1)

-The value in $t9(which values 0xffffffff) is subtracted to the value at $t1(which is 0x00000002) by the command sub and the result is stored at $t9.

Question 2.2.

-The program do nothing since Pseudo code is not permitted.

-The error appears in ‘MARS Message section’:

Error in D:\dl\LAB1\test2.asm line 1 column 1: Extended (pseudo) instruction or format not permitted. See Settings.

Question 3.1.

-The values in the registers did not change, however, in the .data section, the value of $t0(which is 0x0000000c) is stored in a cell addressed 0x1001000(+c). This means the command: sw $t0, 12($t1) will get the value of $t0 and store it in row 0x10010000(the value of $t1) and column +c(because 12 equals to c in hexadecimal) so the address will be 0x1001000+12.

Question 3.2.

Instruction	Sw $t0, 12($t1)	Lw $t1, 12($t1)	Add $t2, $t0, $t1
Machine code	0xad28000c	0x8d29000c	0x01095020
Address	0x00400000	0x00400004	0x00400008

Question 3.3.

- Sw $t0, 12($t1): the value of $t0 is accessed and stored in 0x1001000c.
- Lw $t1, 12($t1): the data in 0x1001000c is loaded and stored in $t1, so the value of $t1 is 0x0000000a.
- Add $t2, $t0, $t1: get the values of $t0 and $t1, then add them together and store the result 0x00000014 in $t2.

2. Exercise

2.1. Simulate the program below and describe its use?

```
1  .data
2  var1: .word 23
3
4  .text
5  __start:
6          lw $t0, var1
7          li $t1, 5
8          sw $t1, var1
```

1. .data: declare data which is used in the program
2. Var1: .word 23: declare that var1 has the value of 23(which is 0x00000017 in hexadecimal)
3.
4. .text: declare the text which is used in the program
5. _start: announce the beginning of the program
6. lw $t0, var1: load the data of var1 to $t0
7. li $t1, 5: load 0x00000005 to $t1
8. sw $t1, var1: store the value of $t1 to var1

The program changes the value of var1 from 23 to 5.

2.2. Simulate the program below and describe its use?

```
1   .data
2   array1: .space 12
3
4   .text
5   __start:
6           la $t0, array1
7           li $t1, 5
8           sw $t1, ($t0)
9           li $t1, 13
10          sw $t1, 4($t0)
11          li $t1, -7
12          sw $t1, 8($t0)
```

1. Data: declare data which is used in the program
2. Array1: .space 12: declare that array1 needs 12 bits
3.
4. .text: declare text in the program
5. _start: announce where the program starts
6. la $t0, array1: load the address of array1 to $t0

7. li $t1, 5: load the value 0x00000005 to $t1(since 5 equals to 0x00000005 in hexadecimal).

8. sw $t1, ($t0): access $t1 and store the value of $t1 to 0x10010000.

9. li $t1, 13: load the value 0x0000000d to $t1(since 13 equals to 0x0000000d in hexadecimal).

10. sw $t1, 4($t0): access $t1 and store the value of $t1 to 0x10010004.

11. li $t1, -7: load the value 0xfffffff9 to $t1(since -7 equals to 0xfffffff9 in hexadecimal).

12. sw $t1, 8($t0): access $t1 and store the value of $t1 to 0x10010008.

This program stores 5, 13(which is d), -7(which is fffffff9) in the array 0x10010000+0, +4, and +8.