REPORT -

Name: Đào Minh Châu

ID: 24560025

1. Classwork: Answer all question in the instruction.

Question 1.1.

The LOOP_CHECK_INPUT checks if the value of $t0 is greater than 10 or not, if true then jump to BREAK label, otherwise the program will ask the user to reinput the number.

Question 1.2.

```
.data
string_input: .asciiz "Please input number n: "
return: .asciiz "Sum =  "
.text
main:
li $v0, 4
la $a0, string_input
syscall
li $v0, 5
syscall
add $t0, $v0, $zero
addi $t1, $t1, 1
addi $t2, $t2, 0
bgt $t0, $zero, sum
j result
```

```
sum:
add $t2, $t2, $t1
addi $t1, $t1, 1
bgt $t1, $t0, result
j sum
result:
li $v0, 4
la $a0, return
syscall
li $v0, 1
add $a0, $t2, $zero
syscall
```

Question 2.1.

The value 14 is stored in the word 0x10010004.

Question 2.2.

The value 1 is stored in the third byte of the word 0x10010018.

The value in 2.1 is stored in a whole word(which contains 4 bytes) while the value in 2.2 is stored in a byte.

Question 2.3.

```
.data
.word 13, 14
array2: .space 16
.byte 2, 4, 1, 5
.text
addi $t1, $t1, 100
```

la $t2, array2

sb $t1, 2($t2)

2. Exercise

2.1. Write a program which allow user to input a character, print the to the I/O window according to each of the following requirements:

- The character preceding and following the input character

Example:

```
Input a character: b
Preceding character: a
Following character: c
```

- Only three types of input data are allowed: number, normal letter, capital letter. If the input is one of above types, print the type of input, if not print "Invalid type".

.data

input: .asciiz "Input a character:"

reinput: .asciiz "Invalid input. Please enter again:"

output1: .asciiz "Preceding character: "

output2: .ascii "Following character: "

endl: "\n"

buffer:

.text

Start:

li $v0, 4

la $a0, input

```
syscall
li $v0, 12
syscall
add $t0, $v0, $zero
addi $t2, $t0, 1
addi $t1, $t0, -1

Check_num:
bgt $t0, 57, Check_upper
blt $t0, 48, BREAK
bne $t0, 48, eol
addi $t1, $zero, 0

eol:
bne $t0, 57, P_NUM
addi $t2, $zero, 0

P_NUM:
li $v0, 4
la $a0, endl
syscall
li $v0, 4
la $a0, output1
syscall
li $v0, 11
la $a0, ($t1)
```

```
syscall
li $v0, 4
la $a0, endl
syscall

F_NUM:
li $v0, 4
la $a0, output2
syscall
li $v0, 11
la $a0, ($t2)
syscall
li $v0, 4
la $a0, endl
syscall
j end

Check_upper:
bgt $t0, 90, Check_lower
blt $t0, 65, BREAK
bne $t0, 65, special_upper
addi $t1, $zero, 0

special_upper:
bne $t0, 90, P_upper
addi $t2, $zero, 0
```

```
P_upper:
li $v0, 4
la $a0, endl
syscall
li $v0, 4
la $a0, output1
syscall
li $v0, 11
la $a0, ($t1)
syscall
li $v0, 4
la $a0, endl
syscall

F_upper:
li $v0, 4
la $a0, output2
syscall
li $v0, 11
la $a0, ($t2)
syscall
j end

Check_lower:
bgt $t0, 122, BREAK
```

```
blt $t0, 97, BREAK
bne $t0, 97, special_lower
addi $t1, $zero, 0

special_lower:
bne $t0, 122, P_lower
addi $t2, $zero, 0

P_lower:
li $v0, 4
la $a0, endl
syscall
li $v0, 4
la $a0, output1
syscall
li $v0, 11
la $a0, ($t1)
syscall
li $v0, 4
la $a0, endl
syscall

F_lower:
li $v0, 4
la $a0, output2
syscall
```

```
li $v0, 11
la $a0, ($t2)
syscall
j end

BREAK:
li $v0, 4
la $a0, endl
syscall
li $v0, 4
la $a0, reinput
syscall
li $v0, 12
syscall
add $t0, $v0, $zero
addi $t2, $t0, 1
addi $t1, $t0, -1

j Check_num
end:
```

2.2. Given a C++ program as below, write the corresponding assembly program?

```
#include <iostream>
#include <stdio.h>
using namespace std;
```

```
int main()
{
   int n, i;
   int product = 1;
   cout << "Input a number (<=5): ";
   cin >> n;
   while (n > 5)
   {
      cout << "Invalid number, please try again: ";
      cin >> n;
   }
   for (i = 1; i <= n; i++)
   {
      product = product * i;
   }
   cout << "Product = " << product;
   return 0;
}
```

```
.data
string_input: .asciiz "Please input a number (<=5): "
string_reinput: .asciiz "Invalid number, please try again: "
result: .asciiz "Product is: "
.text
main:
```

```
li $v0, 4
la $a0, string_input
syscall
li $v0, 5
syscall
add $t0, $v0, $zero
addi $t1, $t1, 1
addi $t5, $t5, 1
addi $t2, $t0, 1

LOOP_CHECK_INPUT:
addi $t3, $t0, -5
blez $t3, , Calc
li $v0, 4
la $a0, string_reinput
syscall
li $v0, 5
syscall
add $t0, $v0, $zero
j LOOP_CHECK_INPUT

Calc:
beq $t1, $t2, BREAK
mult $t1, $t5
mflo $t5
addi $t1, $t1, 1
```

```
j Calc

BREAK:
li $v0, 4
la $a0, result
syscall
li $v0, 1
la $a0, ($t5)
syscall
```