# PROJECT I : INTRODUCTION TO OS

Our team for this project for the Operating System Course is composed of

Trần Tinh Chí (1351063) and Thibault PHILIPPE.

This document is the report of our work. It concerns essentially the modules numbers three and four according to the statement of the asked exercise. Through this paper you will find the solutions that we propose to the given questions of the statement of the project number one. We decided to gather each solution with a justification or some pieces of explanations in order to make easier the understanding of our work and lines of code.

## MODULE 3: Implement exceptions and system calls

***Question a)***

First of all we decided to list clearly what were the system exceptions present in the file entitled ***machine.h***. Those last are the following ones:

* NoException
* SyscallException
* PageFaultException
* ReadOnlyException
* BusErrorException
* AddressErrorException
* OverflowException
* IllegalInstException
* NumExceptionTypes

All of these exceptions are described and comments directly in our codes.The aim of this question was to modify the file Exceptions.cc in order to take into account all of these errors that could potentially occur. For each of those nine exceptions we processed in the same manner: we create a switch that was composed of all of them and if one exception occurred a DEBUG message or error message is printed followed by an interrupt of the simulation. Here you can find quickly an example that illustrates perfectly our previous speech:

case ReadOnlyException:

DEBUG('a', "\n Write attempted to page marked 'read-only'");

interrupt->Halt();

break;

*Management of the ReadOnlyException*

**Question b)**

To answer this question we create again a “swtich-and-cases” structure. More precisely we had to create an interlinked system of switch-case structures. Indeed every time there was a system call exception we had to catch the kind of the system calls (for instance: SC\_EXIT, SC\_EXEC, SC\_JOIN, SC\_FORK and SC\_YIELD). We implement SC\_Sub to verify it.

Testing this structure needed to write in the terminal the following lines:

gmake all  
/userprog/nachos –rs1023 –x /test/sub

*We implement SC\_Sub for testing this structure*

**Question c)**

As we want that NachOS execute the system call forever we were asked to implement a program counter that NachOS would have incremented. To this end we created a function named “IncreaseProgramCounter()”. As described in the comments of our code, this function read firstly the current pc and the next pc and then makes the updates.  
We invoke this function at the end of each system calls and at the end of the “SyscallException” structure.

To verify that at this step everything was okay we just had to write the following commands in the terminal:

gmake all  
/userprog/nachos –rs1023 –x /test/sub

***NOTE:*** For all the next functions when we encounter an error we write into the register 2 the value -1 as a return value for the error.

**Question d)**

In this question we have to implement int ReadInt() in order to allow the user to enter an integer. In this question we had to take into account the fact that the user could make a mistake and not enter an integer.

First we have to create an array of chars with MAX\_LENGHT length (we defined it in the source code in exception.cc file). We have to do it to guarantee that the user cannot write over the buffer. We use this for all the reading function.

We implemented SC\_ReadInt. First of all we verify there was not any problem of memory. Then we made a second verification: if the string contains ‘\001’ then we return an error. If we do not have these two errors then we retrieve the input digit by digit (one by one). Naturally we verify also that each digit is an integer. If not we print an error message, delete the input of the users and provoke a break. After having calculated the value, if this value is negative then we print another error message about the underflow issue because we took already away the negative sign from the input string. Hence the calculated value must be a positive an integer.

We underline that to manage implementing this function we had to create a file “readint.c”, to define “SC\_READInt” in the syscall.h file . First we define the SC code number and the declaration of each function in the syscall.h file. The following questions required us also to implement system call so we have the same kind of declarations in syscall.h file.

We write the value multiplied by the sign into the register 2 as a return value.

**Question e)**

In this question we have to create a function that prints out an integer on screen. First of all we stock the value got from the user in an integer we named “value”. This value was put as the parameter by the user. As value is an integer we have to divide it into several digits. We split the integer value into several digits. We use an algorithm that gives the reversal version of the value so we had do make a loop to reverse the array composed of those digits. We have to reverse the string because we put the digits from the left to the right of the output string but we split the integer from the right to the left. Finally we write the output in the console to form the integer on the screen. We remember that if the sign is less than 0 it means that we have the minus character: ‘-‘ in the first position of the string. So it means that our string we will have as length i+1 from the beginning of the array (one more slot for the minus sign). Otherwise it means the sign is correct or equal to zero. Hence we have to write to the console the string beginning from the second element of the array which length i ( i is the number of digits)

We write 0 to the second register for returning nothing.

**Question f)**

First we have to create an array of chars with MAX\_LENGHT length (we defined it in the source code in exception.cc file). We have to do it to guarantee that the user cannot write over the buffer. We use this for all the reading function. The method to implement t ReadChar is quite similar from the method to implement ReadInt except that we consider characters instead of digits and we have to check if the length of the string input by the user is greater than 1 it means that the user input more than one character and then an error message is printed on the screen. Otherwise we just write the first element of the array of characters (the characters input by the user) into the register 2 as a return value.

**Question g)**

After some allocations’ checking we pick up the value from the register 4 as the parameter. And we use synch console to write this character with length 1 to the console. Then we write 0 to the machine to return nothing.

**Question h)**

After having checked some allocations properties we have to read the address of the input buffer given by the user as a parameter and then we get the maximum length given by the user as a parameter. After this we just read the input string from the console input by the user into the input buffer. We call then the function “System2User()” in order to copy the input buffer from the kernelspace to the userspace and we get the number of copied bytes after having invoked this function. Finally we write this value into the register 2.

Now we will now focus on the “PrintString” function. We call the User2System function to get the input string from the user’s space to kernel’s space. We read the address of the input string given by the user as a parameter.

After that we find the end position of the input string. It is ‘i’ and it is also the length of the string and then we write the input string into the console from the beginning to the end of the string and then we just write ‘i’ in register 2 as a return value. If any other system class that we have not caught it we print an error message and immediately we provoke the halt of the system.

Then we implemented the help program that includes the information of the both team members. The name of this file is “help.c” and the instructions to use the program ASCII and the program Sort.

We implemented again the sort function that allows the user to input the array of integers by themselves. And then print out the sorting array which is sorted by the bubble sort. In this program we just used “ReadInt()” and “PrintSring()” and “PrintInt()”.