CPSC 457 – T02
Luke Iremadze – 10163614
February 1, 2019

Question 1

a) One instruction takes 12 ns, in one second, we can run **83,333,333 instructions**
b) One instruction takes ~6 ns in one second, we can run **166,666,666 instructions**

Question 2

a)
b) Provides a safe testing area for software or simulating different versions of OS to check compatibility of the app being developed
c) Allows the use to run application that are not compatible with the users current OS or allows a safe environment to run a suspicious application without affecting the users existing OS.
d) Allows for the system administrator to set up a test to simulate many computers interacting with each other or have designated virtual machines that to a specific task, so that when you have to do maintenance you can just interact with that one specific OS without affecting the other.

Question 3

a) The process in which the CPU is alerted (interrupted) by task that was set forth by the CPU, this happens on the hardware level triggered by I/O, timer, or user input.
b) Is an intentional software interruption that was set in the executed program, its purpose is to execute a predefined routine in the kernel, such as system calls.
c) Traps start in the user mode which then escalates to the kernel mode by interrupting the CPU. The interrupts happen from an external event that is delivered to the CPU whereas traps occur as a result of an execution code.
d) The kernel mode is necessary for handling the traps and interrupts as the process requires direct interaction with CPU which is on hardware level. User mode has restricted permission with The hardware level and is forced to communicate with the kernel which has direct access to the computers hardware.

Question 4

a)
```
[luke.iremadze@csx a1]$ time ./countLines romeo-and-juliet.txt
4853 romeo-and-juliet.txt

real    0m0.265s
user    0m0.061s
sys     0m0.202s
```

b)
```
[luke.iremadze@csx a1]$ time wc -l romeo-and-juliet.txt
4853 romeo-and-juliet.txt

real    0m0.002s
user    0m0.000s
sys     0m0.002s
```

c) The wc program is faster because it calls the memchr method that accesses the memory directly whereas the read is a more indirect method which can be seen from the time spent in user mode (0.061 s vs 0s). (coreutils, 2019)

Question 5

Mode	countLines			myWc		
	Trial1	Trial2	Trial3	Trial1	Trial2	Trial3
User	0.059	0.067	0.050	0.028	0.005	0.006
System	0.189	0.204	0.195	0.004	0.000	0.002
Real	0.250	0.272	0.247	0.028	0.005	0.003
Avgerage (Real)	0.256			0.012		

Table 1 - comparision of execution times

We can see a dramatic improvement from the new myWc program which differs in that the countLines had an infinite loop but, in myWc I changed the loop condition into checking that the read method was reading bytes from the specified argument file. In addition I implemented a buffer storage which will read 128 bytes at a time instead of going character by character. The improvement is 0.244 s which is more than a 95% improvement!

References

coreutils. (2019, January 1). *Github.* Retrieved from https://github.com/coreutils/coreutils/blob/master/src/wc.c