

CPSC457 - PRINCIPLES OF OPERATING SYSTEMS

University of Calgary
Assignment 1

Instructor: Pavol Federl
TA: Sina Keshvadi
Student: Omar Qureshi
ID: 10086638

Q1 - Assume a CPU instruction cycle with three stages: fetch, decode and execute. For every instruction, the fetch stage takes 5ns, the decode stage takes 3ns, and the execute stage takes 2ns.

- a) How many instructions per second can this CPU execute on average if the stages are not parallelized?

If the stages are not parallelized then we simply add the times each stage takes and then take the reciprocal;

$$\begin{aligned}\text{instructions/s} &= 1/(t_{\text{fetch}} + t_{\text{decode}} + t_{\text{execute}}) = 1/(5\text{ns} + 3\text{ns} + 2\text{ns}) = 1/10\text{ns} = 1/1.0\text{e-8s} \\ &= 100,000,000 \text{ instructions/s}\end{aligned}$$

- b) How many instructions per second can this CPU execute on average if all stages are operating in parallel?

If the stages are parallelized, the instructions per second can be executed as fast as the slowest stage. We then take the maximum of the values at each stage and then take the reciprocal;

$$\begin{aligned}\text{instructions/s} &= 1/(\max(t_{\text{fetch}} + t_{\text{decode}} + t_{\text{execute}})) = 1/(5\text{ns}) = 1/5.0\text{e-9s} \\ &= 200,000,000 \text{ instructions/s}\end{aligned}$$

Q2 - Describe one benefit of using virtual machines for each of the following:

- a) from a company's perspective;

One benefit companies can experience with running virtual machines is system consolidation. This helps save a lot of money because they can buy one big server instead of many smaller ones.

- b) from a programmer's / developer's perspective;

One benefit developers experience with running virtual machines is running multiple different OSs or OS versions concurrently. This can help the developer ensure that a common code base is running smoothly on all operating systems side by side and troubleshoot as needed saving a lot of time, instead manually booting into another OS one by one.

- c) from a regular user's perspective; and

One benefit users experience with running virtual machines is that users can run software they otherwise couldn't on their original operating system. An example of such software may be games an operating system like macOS may not support though through the use of a virtual machine they can still run it on their system without having to purchase a dedicated Windows machine.

d) from a system administrator's perspective.

One benefit sys admins experience with running virtual machines is that they can run the mix of OSs that are distributed throughout the company on possibly one machine. This can allow the admin to tackle issues that arise with different operating systems daily on a single machine rather than having to switch between, for example, individual Linux and Windows environments.

Q3 - Answer the following;

a) Define interrupts.

An interrupt is a signal sent by a piece of hardware (such as I/O devices) that notifies CPU of an event needing attention. The processor pauses current activity and executes appropriate interrupt handler from kernel mode. After event is handled, processor unpauses its previous activity and continues it.

b) Define traps.

A trap is a software generated interrupt in the form of special instructions that switch processor from user mode to kernel mode. They are invoked by conditions predefined by OS (such as division by 0). The OS then performs some kernel routine and then returns control back to originating process.

c) What are some key differences between hardware interrupts and traps?

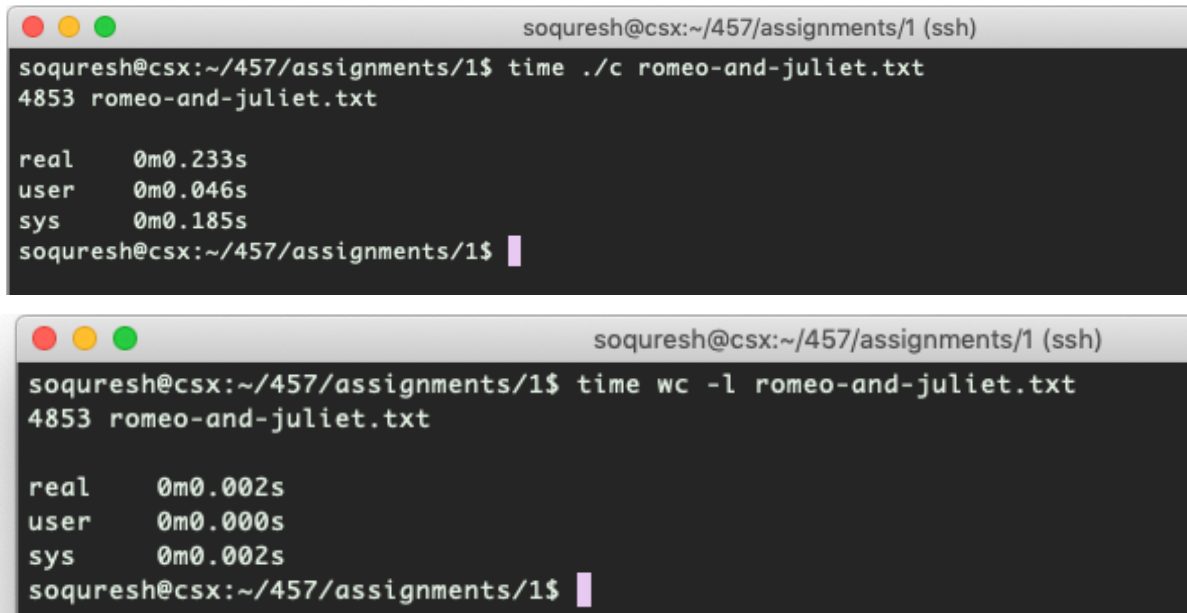
Interrupts are due to signals generated by external events (like the timer) which are delivered to CPU. Traps are invoked by error conditions or sys calls, which are internal events. Interrupts are also asynchronous with the processor while traps are synchronous with activity of the processor. Interrupts are unpredictable while traps occur as a result of a machine instruction and are predictable (such as sys calls).

d) Why are interrupts handled in kernel mode instead of user mode?

User mode does not have unrestricted access to the system hardware and user mode also runs a slightly limited instruction set. Kernel mode on the other hand provides unrestricted access to the hardware. This is what the system needs in that interrupted state; direct access with memory and hardware to handle those exceptional states fast as possible.

Q4 - Answer the following;

a) What are the outputs of the time commands? Copy/paste this from the terminal output to your report.



```
soquresh@csx:~/457/assignments/1 (ssh)
soquresh@csx:~/457/assignments/1$ time ./c romeo-and-juliet.txt
4853 romeo-and-juliet.txt

real    0m0.233s
user    0m0.046s
sys     0m0.185s
soquresh@csx:~/457/assignments/1$

soquresh@csx:~/457/assignments/1 (ssh)
soquresh@csx:~/457/assignments/1$ time wc -l romeo-and-juliet.txt
4853 romeo-and-juliet.txt

real    0m0.002s
user    0m0.000s
sys     0m0.002s
soquresh@csx:~/457/assignments/1$
```

b) How much time did the C++ program and 'wc' spend in the kernel mode and user mode, respectively?

C++ program spent 0.185s kernel mode, 0.46s in user mode.

wc util spent 0.002s kernel mode, 0.000s in user mode.

c) Why is the 'wc' program faster than the C++ program?

On line 45, there is a while loop utilizing a system call for each character in the .txt file. This results in a huge number of system calls which constantly change the mode from user to kernel mode. On the other hand the wc utility instead uses a single buffer for all the contents of the file, resulting in minimal system calls. This results in much faster execution times.

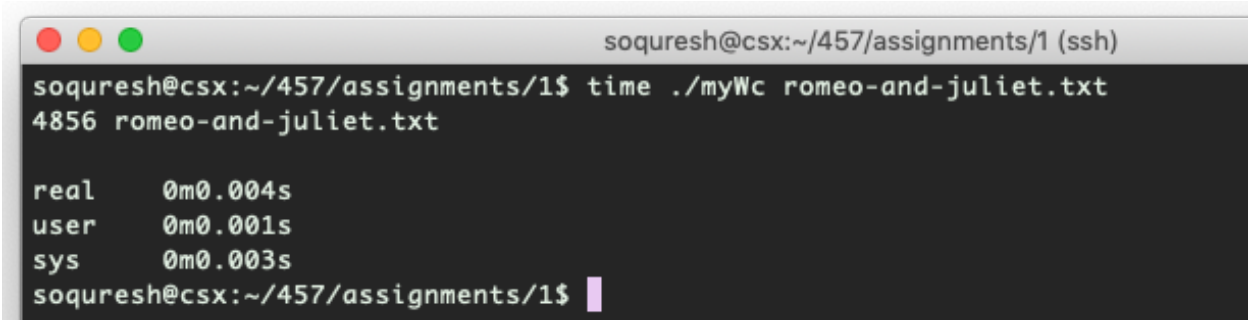
Q5 - Programming question

See .cpp file

Q6 - Written question

Obtain the timings of your program from Q5 and compare it to the timings you obtained for `countLines.cpp` and `wc`. Is your program from Q5 faster than `countLines.cpp`? Why do you think that is? Is it faster or slower than '`wc -l`' and why?

We focus on the area of the while loop, where instead of executing a system call for every character, we instead use a larger character array to hold a lot more characters instead of a single one. This reduces the usage of system calls utilized as compared to the original program, which results in faster execution times as shown below;



```
soquresh@csx:~/457/assignments/1 (ssh)
soquresh@csx:~/457/assignments/1$ time ./myWc romeo-and-juliet.txt
4856 romeo-and-juliet.txt

real    0m0.004s
user    0m0.001s
sys     0m0.003s
soquresh@csx:~/457/assignments/1$
```

The C++ program is still, however, slower than the `wc` utility. This is because the `wc` utility uses a single buffer to read in all the contents of the text file, whereas in `myWc` it is a 256 char buffer, which is smaller and thus more than one buffer is used, resulting in more sys calls.

A comparison between all the programs is summarized in the chart below;

	myWc.cpp	countLines.cpp	wc -l
real	0.004s	0.233s	0.002s
user	0.001s	0.046s	0.000s
sys	0.003s	0.185s	0.002s