Computer Science and Engineering & Information Technology (2^{nd} Year B.Tech.)
IIIT Kalyani, West Bengal

Operating System Lab (CS 411): (Spring: 2019-2020)

Assignment - 3 *Marks: 20*
Assignment Out: *31st January, 2020*

1. Write a C++ program (`myawk.c++` $\Rightarrow$ `myawk`) that creates a child process and loads the `awk` program using **execvp()**. The path of `awk` may be `/usr/bin/awk` or `/bin/awk` (check on your machine). The parent does nothing but waits for the child to terminate. **Do not use** `system()` in your program that executes a shell command.

   A few sample output are:

```
$ awk 'BEGIN { print "IIIT Kalyani" }'
IIIT Kalyani
$ myawk 'BEGIN { print "IIIT Kalyani" }'
IIIT Kalyani
$ awk '{ print }'
India
India
Bharat
Bharat
West Bengal
West Bengal
$ myawk '{ print }'
India
India
Bharat
Bharat
West Bengal
West Bengal
$ ls -l | awk /o/
total 60
-rwxr-xr-x 1 gb gb 13440 Jan 24 07:53 a.out
-rw-r--r-- 1 gb gb   804 Jan 24 05:31 clockFreq1.c++
-rw-r--r-- 1 gb gb  1193 Jan 24 07:52 clockFreq.c++
-rw-r--r-- 1 gb gb     4 Jan 25 17:50 prog
$ ls -l | myawk /o/
total 60
-rwxr-xr-x 1 gb gb 13440 Jan 24 07:53 a.out
-rw-r--r-- 1 gb gb   804 Jan 24 05:31 clockFreq1.c++
-rw-r--r-- 1 gb gb  1193 Jan 24 07:52 clockFreq.c++
-rw-r--r-- 1 gb gb     4 Jan 25 17:50 prog
```

2. Write a C++ program to get a rough estimate of frequency of the CPU clock.
   Use the x86-64 machine instruction `rdtscp` (read time stamp counter and processor ID)[1] The instruction loads the processor's *time-stamp counter* `MSR (64-bit)` into `edx:eax` pair. The lower order 32-bit of MSR goes to `eax` (lower 32-bit of `rax`) and the higher order order 32-bit of MSR goes to `edx` (lower 32-bit of `rdx`). Higher order bits of `rax` and `rdx` are initialized to zero.

   The time stamp counter `MSR` is incremented at every clock cycle of the processor, and is set to zero when the processor is *reset*.

   A program can get the content of the time stamp counter from `eax` at the beginning and at the end of an **event** (make sure that `edx` has not

[1] You will get more information from *Intel 64 and IA-32 Architectures Developer's manual: Vol. 2B* from internet or Reading Material.

changed in the mean time). Their difference is the number of clock cycles (c) during the event.

Also use the system call `clock_gettime()` to find the time elapsed during the same **event** (approximately). The system call
`int clock_gettime(clockid_t clk_id, struct timespec *tp);` when invoked with the first parameter `CLOCK_REALTIME` stores the time (t) in the object `*tp` of type

```
struct timespec {
               time_t   tv_sec;        /* seconds */
               long     tv_nsec;       /* nanoseconds */
           };
```

Invoking it before and after the **event** can give the approximate time elapsed during the event. Estimate the approximate clock frequency using the time elapsed and the number of clock cycles during the **event**.

The event may be a simple for-loop with large number of iterations. We may use several runs to get an average value of the clock frequency.

The output should be as follows:

```
$ a.out
Enter no of iterations of delay loop: 1000
Enter no of runs to average: 10
Clock: 2344.29
$ a.out
Enter no of iterations of delay loop: 10000
Enter no of runs to average: 10
Clock: 2388.96
```

Use the `lscpu` command to verify your result (`/usr/bin/lscpu`).