WES 237A – Assignment 3, Power Management Unit

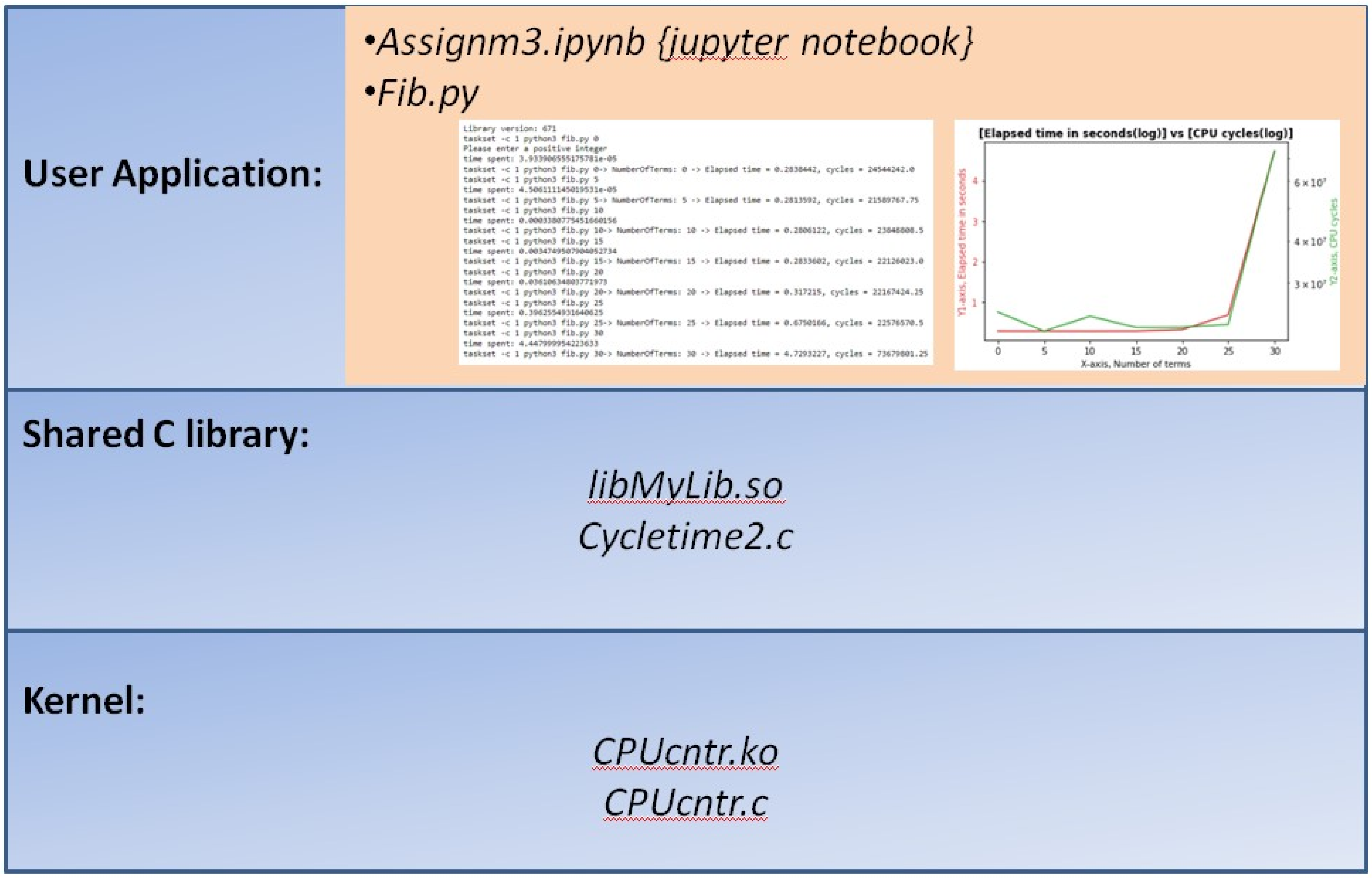
Ricardo Lizarraga, rlizarraga0@gmail.com , *619.252.4157* | **PID: A69028483**

Assignments Rubric

yes	**Report Submitted?**	02/16/24
yes	**Video Uploaded?**	https://drive.google.com/drive/folders/1IloHq0BFUTo66M-6h4H_lWlF5VslN8-k?usp=drive_link
yes	**Pushed to Github?**	https://github.com/RiLizarraga/WES237A_Assign3
ok	**Does the video demonstration show correct execution?**	Same piece of code by a single run click of the mouse, collects, generates and plots data for analysis
ok	**Is the submitted code correct?**	Source code, documentation and readme, public
ok	**How well does the report outlines the design of the code?**	See Solution Layer diagram
ok	**How well does the report describe the results?**	Detailed data results and plotting
yes	**Does the Report detail the student's grasp on the goals/objectives of the assignment?**	Cover from low level Kernel implementation, a Shared C library to all the way to high level User application in Python {Jupyter notebook}

In lab, we experimented with C++ code for initializing the PMU counters and retrieving the cyclecount.
In this assignment, you'll be setting up your **PMU counter** to use in Python.

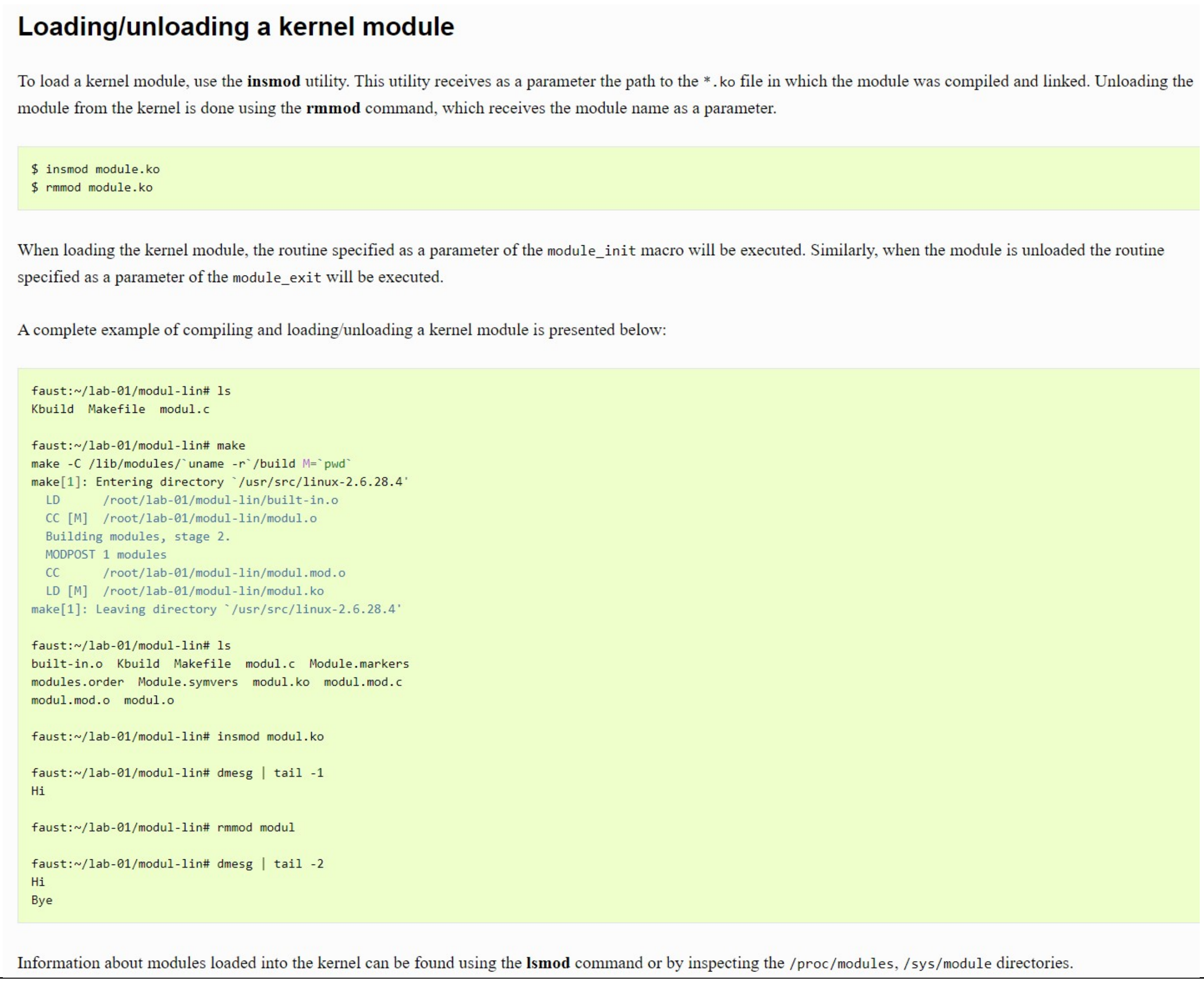
Solution Layer Diagram



```
Library version: 671
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.933906555175781e-05
taskset -c 1 python3 fib.py 0-> NumberOfTerms: 0 -> Elapsed time = 0.2838442, cycles = 24544242.0
taskset -c 1 python3 fib.py 5
time spent: 4.506111145019531e-05
taskset -c 1 python3 fib.py 5-> NumberOfTerms: 5 -> Elapsed time = 0.2813592, cycles = 21589767.75
taskset -c 1 python3 fib.py 10
time spent: 0.0003380775451660156
taskset -c 1 python3 fib.py 10-> NumberOfTerms: 10 -> Elapsed time = 0.2806122, cycles = 23848808.5
taskset -c 1 python3 fib.py 15
time spent: 0.0034749507904052734
taskset -c 1 python3 fib.py 15-> NumberOfTerms: 15 -> Elapsed time = 0.2833602, cycles = 22126023.0
taskset -c 1 python3 fib.py 20
time spent: 0.03610634803771973
taskset -c 1 python3 fib.py 20-> NumberOfTerms: 20 -> Elapsed time = 0.317215, cycles = 22167424.25
taskset -c 1 python3 fib.py 25
time spent: 0.3962554931640625
taskset -c 1 python3 fib.py 25-> NumberOfTerms: 25 -> Elapsed time = 0.6750166, cycles = 22576570.5
taskset -c 1 python3 fib.py 30
time spent: 4.447999954223633
taskset -c 1 python3 fib.py 30-> NumberOfTerms: 30 -> Elapsed time = 4.7293227, cycles = 73679801.25
```

CPUcntr.ko

ko=kernel object: loadable kernel module
In the Linux operating system, a kernel object, or ko file, is an object file that extends the Linux Distribution's kernel. Ko files are linked with automatically generated data structures that the kernel needs.
Ko files are part of the Linux kernel itself, and the kernel can load and unload them. They are also known as Loadable Kernel Modules (LKMs) because they can be loaded without changing the kernel.
Ko files are used to provide drivers for new hardware, such as IoT expansion cards that aren't included in the Linux Distribution.
Ko files are ELF files, which stands for "Executable and Linking Format". The directory for each installed kernel is located at /lib/modules/.



Loading/unloading a kernel module

To load a kernel module, use the **insmod** utility. This utility receives as a parameter the path to the `*.ko` file in which the module was compiled and linked. Unloading the module from the kernel is done using the **rmmod** command, which receives the module name as a parameter.

```
$ insmod module.ko
$ rmmod module.ko
```

When loading the kernel module, the routine specified as a parameter of the `module_init` macro will be executed. Similarly, when the module is unloaded the routine specified as a parameter of the `module_exit` will be executed.

A complete example of compiling and loading/unloading a kernel module is presented below:

```
faust:~/lab-01/modul-lin# ls
Kbuild  Makefile  modul.c

faust:~/lab-01/modul-lin# make
make -C /lib/modules/`uname -r`/build M=`pwd`
make[1]: Entering directory `/usr/src/linux-2.6.28.4'
  LD      /root/lab-01/modul-lin/built-in.o
  CC [M]  /root/lab-01/modul-lin/modul.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/lab-01/modul-lin/modul.mod.o
  LD [M]  /root/lab-01/modul-lin/modul.ko
make[1]: Leaving directory `/usr/src/linux-2.6.28.4'

faust:~/lab-01/modul-lin# ls
built-in.o  Kbuild  Makefile  modul.c  Module.markers
modules.order  Module.symvers  modul.ko  modul.mod.c
modul.mod.o  modul.o

faust:~/lab-01/modul-lin# insmod modul.ko

faust:~/lab-01/modul-lin# dmesg | tail -1
Hi

faust:~/lab-01/modul-lin# rmmod modul

faust:~/lab-01/modul-lin# dmesg | tail -2
Hi
Bye
```

Information about modules loaded into the kernel can be found using the **lsmod** command or by inspecting the `/proc/modules`, `/sys/module` directories.

libMyLib.so

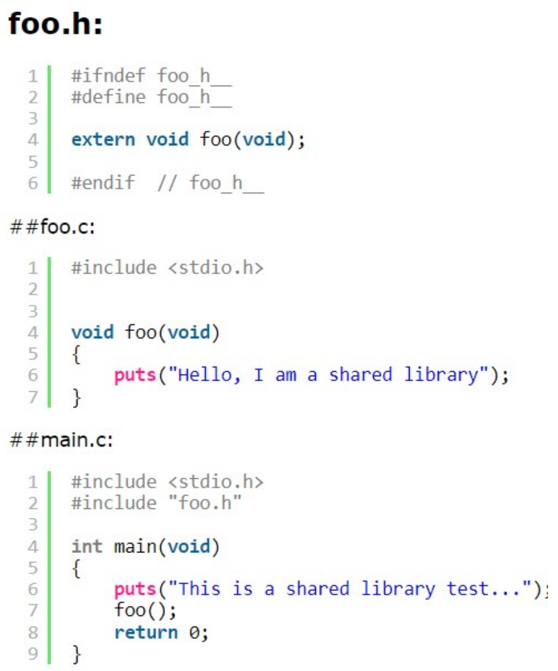
Shared libraries with GCC on Linux

https://www.cprogramming.com/tutorial/shared-libraries-linux-gcc.html

Libraries are an indispensable tool for any programmer. They are pre-existing code that is compiled and ready for you to use. They often provide generic functionality, like linked lists or binary trees that can hold any data, or specific functionality like an interface to a database server such as MySQL.

Most larger software projects will contain several components, some of which you may find use for later on in some other project, or that you just want to separate out for organizational purposes. When you have a reusable or logically distinct set of functions, it is helpful to build a library from it so that you do not have to copy the source code into your current project and recompile it all the time - and so you can keep different modules of your program disjoint and change one without affecting others. Once it is been written and tested, you can safely reuse it over and over again, saving the time and hassle of building it into your project every time.

Building static libraries is fairly simple, and since we rarely get questions on them, I will not cover them. I will stick with shared libraries, which seem to be more confusing for most people.
Before we get started, it might help to get a quick rundown of everything that happens from source code to running program:



foo.h:

```c
#ifndef foo_h__
#define foo_h__

extern void foo(void);

#endif  // foo_h__
```

##foo.c:

```c
#include <stdio.h>


void foo(void)
{
    puts("Hello, I am a shared library");
}
```

##main.c:

```c
#include <stdio.h>
#include "foo.h"

int main(void)
{
    puts("This is a shared library test...");
    foo();
    return 0;
}
```

1. C Preprocessor: This stage processes all the preprocessor directives. Basically, any line that starts with a #, such as #define and #include.
2. Compilation Proper: Once the source file has been preprocessed, the result is then compiled. Since many people refer to the entire build process as compilation, this stage is often referred to as compilation proper. This stage turns a .c file into an .o (object) file.
3. Linking: Here is where all of the object files and any libraries are linked together to make your final program. Note that for static libraries, the actual library is placed in your final program, while for shared libraries, only a reference to the library is placed inside. Now you have a complete program that is ready to run. You launch it from the shell, and the program is handed off to the loader.
4. Loading: This stage happens when your program starts up. Your program is scanned for references to shared libraries. Any references found are resolved and the libraries are mapped into your program.

Steps 3 and 4 are where the magic (and confusion) happens with shared libraries.
foo.h defines the interface to our library, a single function, foo(). foo.c contains the implementation of that function, and main.c is a driver program that uses our library.
For the purposes of this example, everything will happen in /home/username/foo

Step 1: Compiling with Position Independent Code

We need to compile our library source code into position-independent code (PIC):[1]

```
$ gcc -c -Wall -Werror -fpic foo.c
```

Step 2: Creating a shared library from an object file

Now we need to actually turn this object file into a shared library. We will call it libfoo.so:

```
gcc -shared -o libfoo.so foo.o
```

Step 3: Linking with a shared library

As you can see, that was actually pretty easy. We have a shared library. Let us compile our main.c and link it with libfoo. We will call our final program test. Note that the -lfoo option is not looking for foo.o, but libfoo.so. GCC assumes that all libraries start with lib and end with .so or .a (.so is for shared object or shared libraries, and .a is for archive, or statically linked libraries).

```
$ gcc -Wall -o test main.c -lfoo
/usr/bin/ld: cannot find -lfoo
collect2: ld returned 1 exit status
```

Telling GCC where to find the shared library

Uh-oh! The linker does not know where to find libfoo. GCC has a list of places it looks by default, but our directory is not in that list.[2] We need to tell GCC where to find libfoo.so. We will do that with the -L option. In this example, we will use the current directory, /home/username/foo:

```
$ gcc -L/home/username/foo -Wall -o test main.c -lfoo
```

Step 4: Making the library available at runtime

Good, no errors. Now let us run our program:

```
$ ./test
./test: error while loading shared libraries: libfoo.so: cannot open shared object file: No such file or directory
```

Part A3.0: New kernel_module code

- Download the new kernel_module.zip code from canvas.
- Make and insert this new module following the ‘make’ and ‘insmod’ instructions from lab and in the README.

Here are some additional things to keep in mind when using the **insmod** command on the PYNQ-Z2 board:

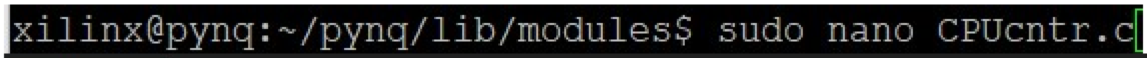
- The insmod command must be run as root.
- The driver module that you want to load must be located in the /lib/modules directory.
- The driver module that you want to load must be compatible with the kernel version that is running on the PYNQ-Z2 board.

To insert the CPUcntr.ko module into the kernel of a PYNQ-Z2 board, you can use the following command:
sudo insmod CPUcntr.ko
https://www.youtube.com/watch?v=SOo1rbnryeo “Making Simple Linux Kernel Module in C”

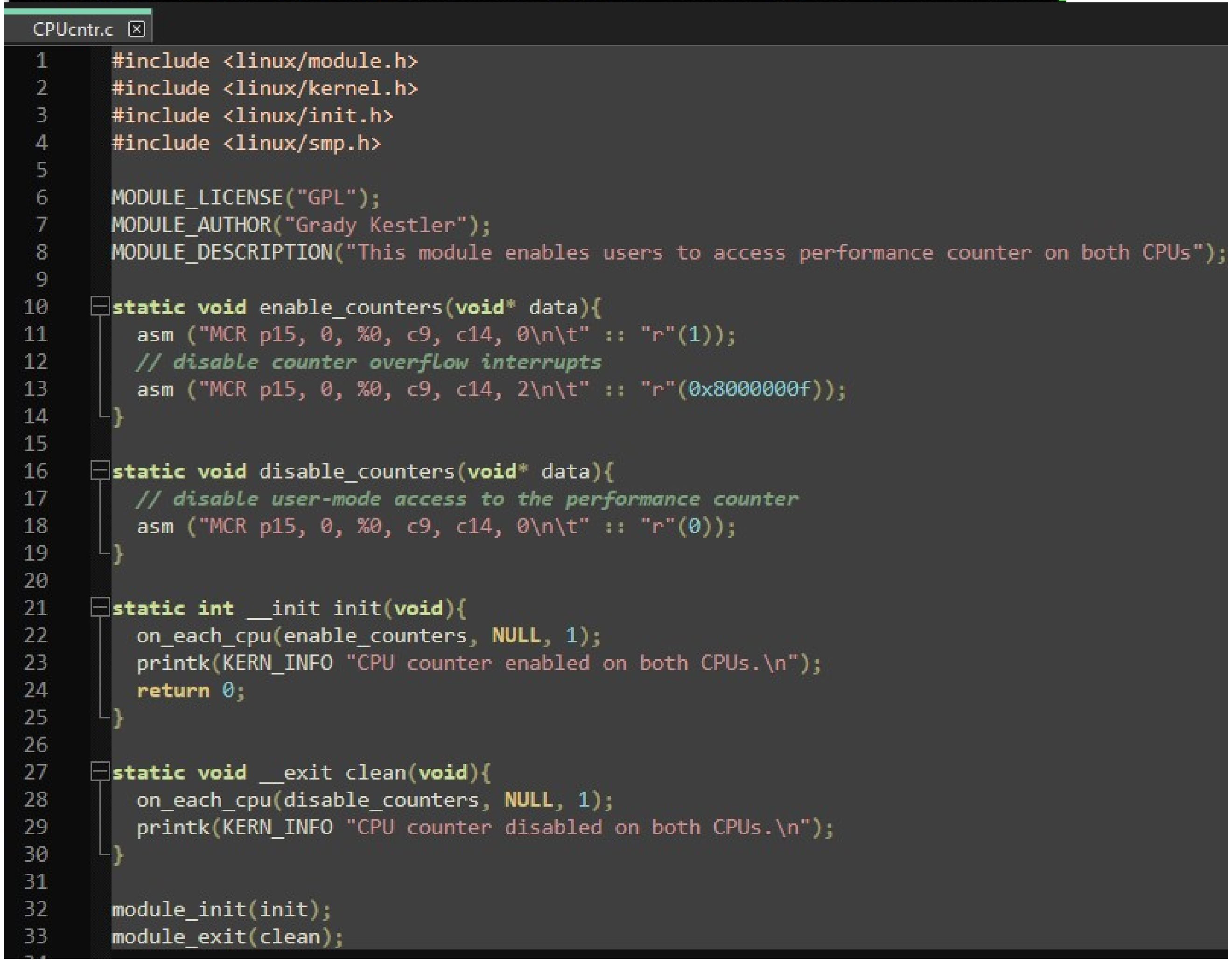
\\192.168.2.99\xilinx\pynq\lib goto “**/home/xilinx/pynq/lib/modules**”

```
xilinx@pynqclean$ mkdir modules
xilinx@pynq:~/pynq/lib$ cd
xilinx@pynq:~/pynq/lib/modules$
```

Create source code file in our modules folder:



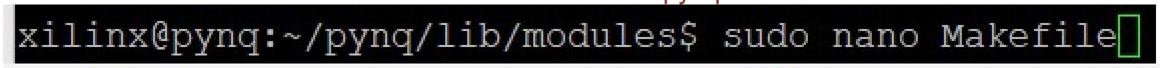
```
xilinx@pynq:~/pynq/lib/modules$ sudo nano CPUcntr.c
```



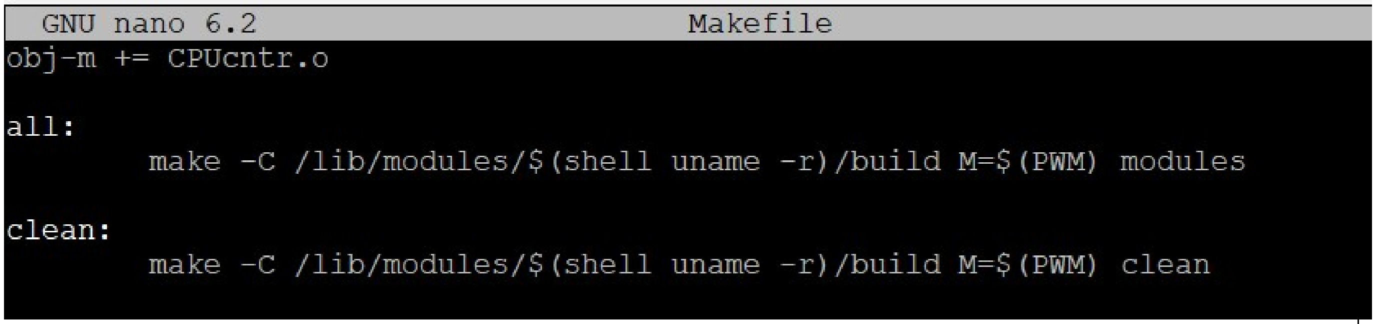
```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/smp.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Grady Kestler");
MODULE_DESCRIPTION("This module enables users to access performance counter on both CPUs");

static void enable_counters(void* data){
  asm ("MCR p15, 0, %0, c9, c14, 0\n\t" :: "r"(1));
  // disable counter overflow interrupts
  asm ("MCR p15, 0, %0, c9, c14, 2\n\t" :: "r"(0x8000000f));
}

static void disable_counters(void* data){
  // disable user-mode access to the performance counter
  asm ("MCR p15, 0, %0, c9, c14, 0\n\t" :: "r"(0));
}

static int __init init(void){
  on_each_cpu(enable_counters, NULL, 1);
  printk(KERN_INFO "CPU counter enabled on both CPUs.\n");
  return 0;
}

static void __exit clean(void){
  on_each_cpu(disable_counters, NULL, 1);
  printk(KERN_INFO "CPU counter disabled on both CPUs.\n");
}

module_init(init);
module_exit(clean);
```

Create the Makefile to create our new Kernel Object file:

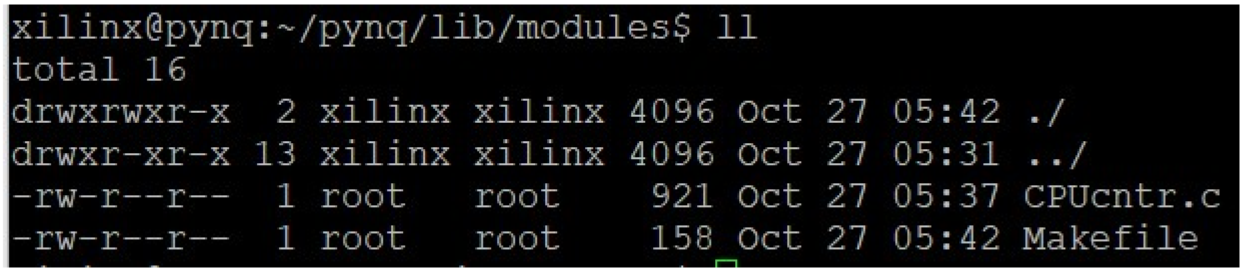
cd /home/xilinx/pynq/lib/modules



```
xilinx@pynq:~/pynq/lib/modules$ sudo nano Makefile
```



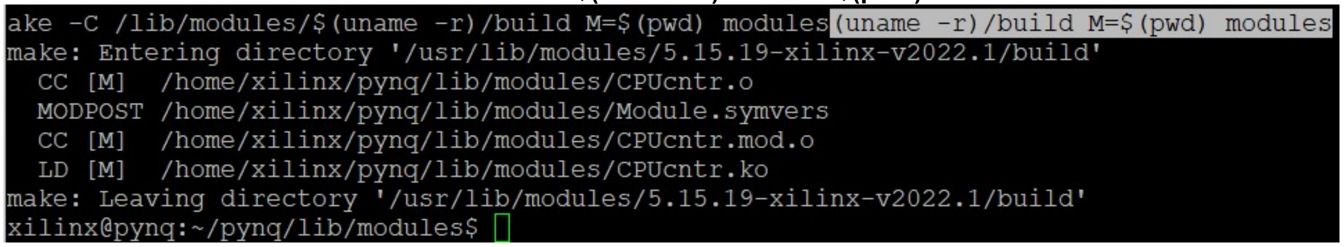
```
  GNU nano 6.2                                Makefile
obj-m += CPUcntr.o

all:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWM) modules

clean:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWM) clean
```



```
xilinx@pynq:~/pynq/lib/modules$ ll
total 16
drwxrwxr-x  2 xilinx xilinx 4096 Oct 27 05:42 ./
drwxr-xr-x 13 xilinx xilinx 4096 Oct 27 05:31 ../
-rw-r--r--  1 root   root    921 Oct 27 05:37 CPUcntr.c
-rw-r--r--  1 root   root    158 Oct 27 05:42 Makefile
```

To compile:

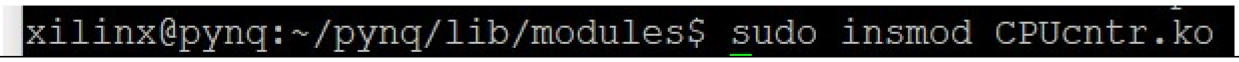
make -C /lib/modules/$(uname -r)/build M=$(pwd) modules



```
ake -C /lib/modules/$(uname -r)/build M=$(pwd) modules(uname -r)/build M=$(pwd) modules
make: Entering directory '/usr/lib/modules/5.15.19-xilinx-v2022.1/build'
  CC [M]  /home/xilinx/pynq/lib/modules/CPUcntr.o
  MODPOST /home/xilinx/pynq/lib/modules/Module.symvers
  CC [M]  /home/xilinx/pynq/lib/modules/CPUcntr.mod.o
  LD [M]  /home/xilinx/pynq/lib/modules/CPUcntr.ko
make: Leaving directory '/usr/lib/modules/5.15.19-xilinx-v2022.1/build'
xilinx@pynq:~/pynq/lib/modules$
```

To insert module onto cpu0:

insmod CPUcntr.ko

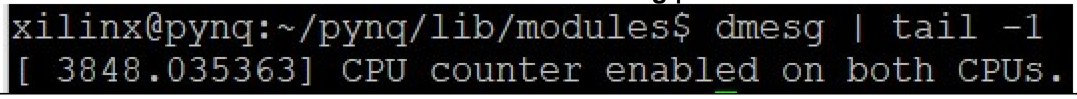


```
xilinx@pynq:~/pynq/lib/modules$ sudo insmod CPUcntr.ko
```

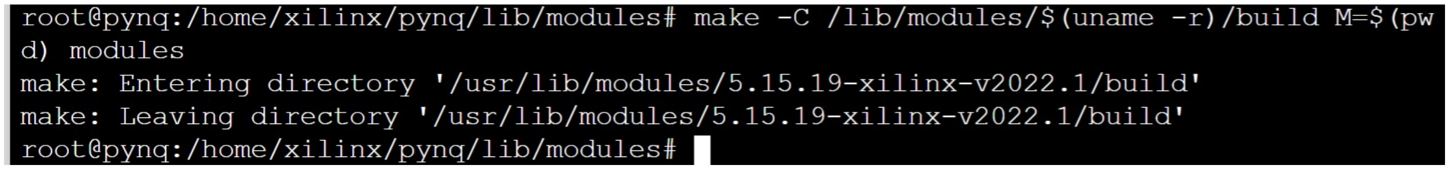
• Check that it is inserted on both CPUs by checking the dmesg | tail output

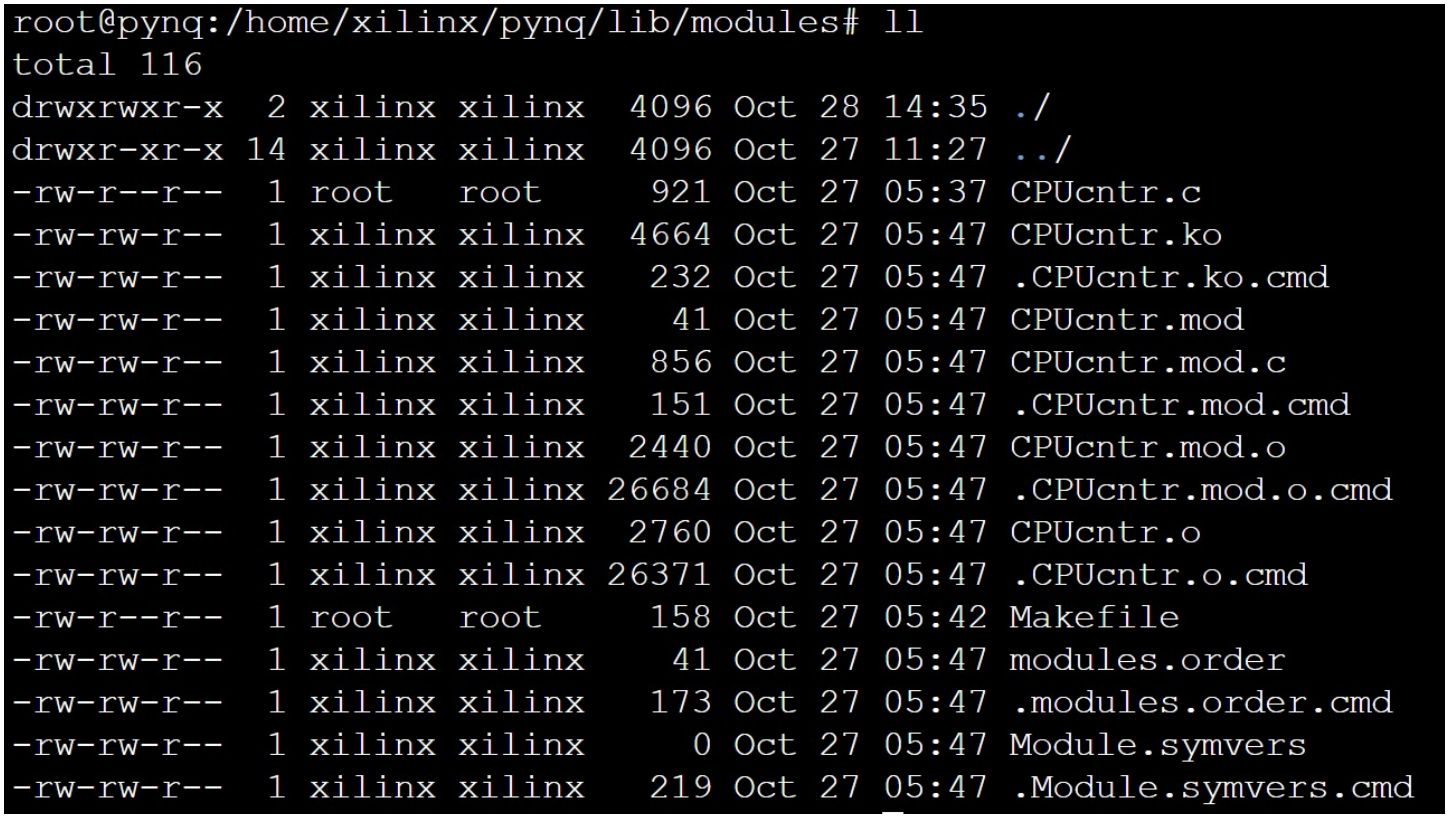
To check module:

dmesg | tail -1



```
xilinx@pynq:~/pynq/lib/modules$ dmesg | tail -1
[ 3848.035363] CPU counter enabled on both CPUs.
```

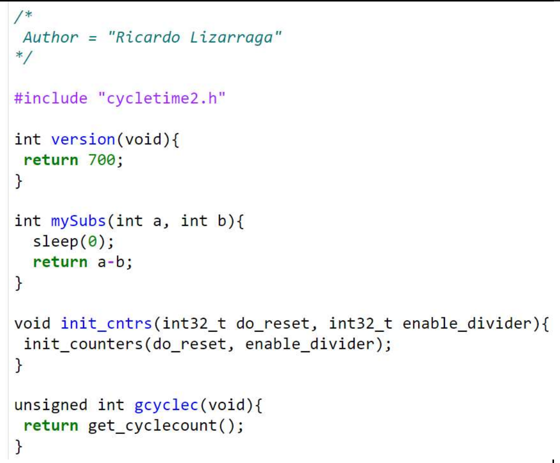

```
root@pynq:/home/xilinx/pynq/lib/modules# make -C /lib/modules/$(uname -r)/build M=$(pw
d) modules
make: Entering directory '/usr/lib/modules/5.15.19-xilinx-v2022.1/build'
make: Leaving directory '/usr/lib/modules/5.15.19-xilinx-v2022.1/build'
root@pynq:/home/xilinx/pynq/lib/modules# 
```


```
root@pynq:/home/xilinx/pynq/lib/modules# ll
total 116
drwxrwxr-x  2 xilinx xilinx  4096 Oct 28 14:35 ./
drwxr-xr-x 14 xilinx xilinx  4096 Oct 27 11:27 ../
-rw-r--r--  1 root   root     921 Oct 27 05:37 CPUcntr.c
-rw-rw-r--  1 xilinx xilinx  4664 Oct 27 05:47 CPUcntr.ko
-rw-rw-r--  1 xilinx xilinx   232 Oct 27 05:47 .CPUcntr.ko.cmd
-rw-rw-r--  1 xilinx xilinx    41 Oct 27 05:47 CPUcntr.mod
-rw-rw-r--  1 xilinx xilinx   856 Oct 27 05:47 CPUcntr.mod.c
-rw-rw-r--  1 xilinx xilinx   151 Oct 27 05:47 .CPUcntr.mod.cmd
-rw-rw-r--  1 xilinx xilinx  2440 Oct 27 05:47 CPUcntr.mod.o
-rw-rw-r--  1 xilinx xilinx 26684 Oct 27 05:47 .CPUcntr.mod.o.cmd
-rw-rw-r--  1 xilinx xilinx  2760 Oct 27 05:47 CPUcntr.o
-rw-rw-r--  1 xilinx xilinx 26371 Oct 27 05:47 .CPUcntr.o.cmd
-rw-r--r--  1 root   root     158 Oct 27 05:42 Makefile
-rw-rw-r--  1 xilinx xilinx    41 Oct 27 05:47 modules.order
-rw-rw-r--  1 xilinx xilinx   173 Oct 27 05:47 .modules.order.cmd
-rw-rw-r--  1 xilinx xilinx     0 Oct 27 05:47 Module.symvers
-rw-rw-r--  1 xilinx xilinx   219 Oct 27 05:47 .Module.symvers.cmd
```

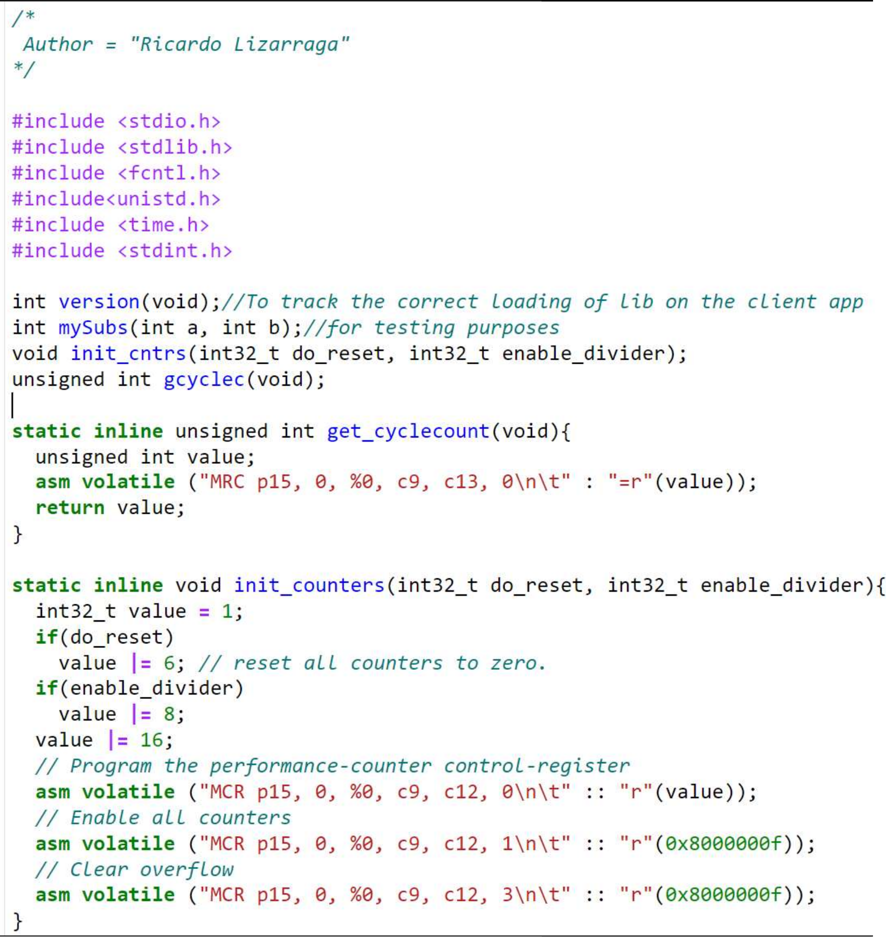
Part A3.1: Access PMU from python

• Create a shared library object with two fuctions by wrapping the **cycletime.h** into a new shared opticed library (see Lab2)

– **One function to initialize the PMU counters**
– **One function to get the cycle count**

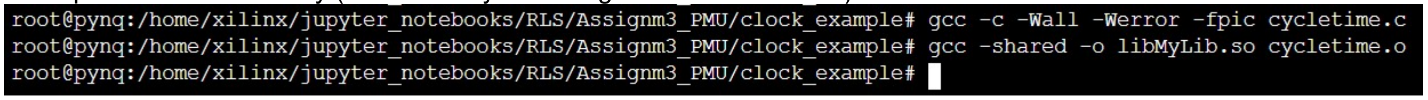


```
/*
 Author = "Ricardo Lizarraga"
*/

#include "cycletime2.h"

int version(void){
 return 700;
}

int mySubs(int a, int b){
  sleep(0);
  return a-b;
}

void init_cntrs(int32_t do_reset, int32_t enable_divider){
 init_counters(do_reset, enable_divider);
}

unsigned int gcyclec(void){
 return get_cyclecount();
}
```



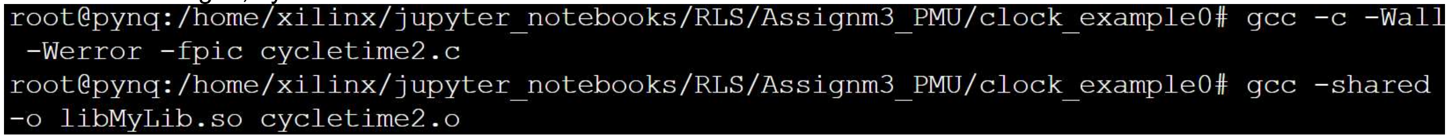
```
/*
 Author = "Ricardo Lizarraga"
*/

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include<unistd.h>
#include <time.h>
#include <stdint.h>

int version(void);//To track the correct loading of lib on the client app
int mySubs(int a, int b);//for testing purposes
void init_cntrs(int32_t do_reset, int32_t enable_divider);
unsigned int gcyclec(void);

static inline unsigned int get_cyclecount(void){
  unsigned int value;
  asm volatile ("MRC p15, 0, %0, c9, c13, 0\n\t" : "=r"(value));
  return value;
}

static inline void init_counters(int32_t do_reset, int32_t enable_divider){
  int32_t value = 1;
  if(do_reset)
    value |= 6; // reset all counters to zero.
  if(enable_divider)
    value |= 8;
  value |= 16;
  // Program the performance-counter control-register
  asm volatile ("MCR p15, 0, %0, c9, c12, 0\n\t" :: "r"(value));
  // Enable all counters
  asm volatile ("MCR p15, 0, %0, c9, c12, 1\n\t" :: "r"(0x8000000f));
  // Clear overflow
  asm volatile ("MCR p15, 0, %0, c9, c12, 3\n\t" :: "r"(0x8000000f));
}
```

• Compile the shared library (see Lab2 if you've forgotten how to do this)



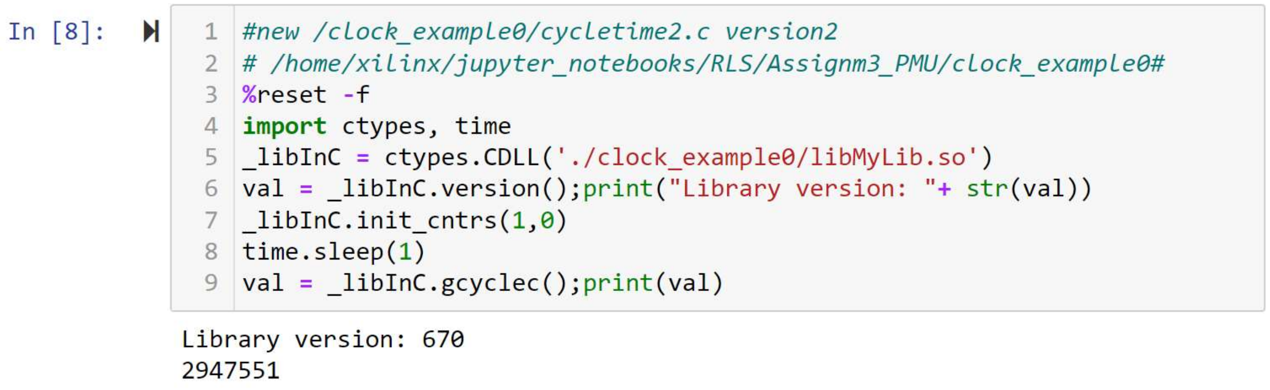
```
root@pynq:/home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example# gcc -c -Wall -Werror -fpic cycletime.c
root@pynq:/home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example# gcc -shared -o libMyLib.so cycletime.o
root@pynq:/home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example# 
```

Because of changes, cycletime2.c



```
root@pynq:/home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example0# gcc -c -Wall
 -Werror -fpic cycletime2.c
root@pynq:/home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example0# gcc -shared
-o libMyLib.so cycletime2.o
```

• Access the shared library functions using the ctypes module, **but don't wrap the function calls in a python function.**



```
In [8]:
#new /clock_example0/cycletime2.c version2
# /home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example0#
%reset -f
import ctypes, time
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))
_libInC.init_cntrs(1,0)
time.sleep(1)
val = _libInC.gcyclec();print(val)
```

```
Library version: 670
2947551
```

In [7]:

```python
#new /clock_example0/cycletime2.c version2
# /home/xilinx/jupyter_notebooks/RLS/Assignm3_PMU/clock_example0#
# https://docs.python.org/3/library/ctypes.html

%reset -f
import ctypes, time
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))
_libInC.init_cntrs(1,1)#reset=1, enable_divider=1
start = ctypes.c_uint(_libInC.gcyclec()).value
time.sleep(1)
stop = ctypes.c_uint(_libInC.gcyclec()).value
print("  start: "+str(start)+"\n"\
      "   stop: "+str(stop)+"\n"\
      "Elapsed: "+str(stop-start))
```

```
Library version: 700
  start: 5914
   stop: 170397
Elapsed: 164483
```

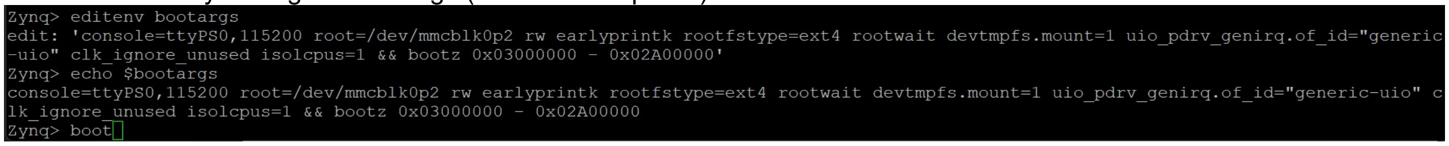
Part A3.2: Comparing and Gathering Data

In this section, we are going to use psutil to monitor CPU usage in percent, and the time module and PMU counting to evaluate the recursive fibonacci sequence timing operations.

```python
# use psutil
import psutil, time

# Create a function to get the CPU usage
def get_cpu_usage():
  """Returns the CPU usage as a percentage."""
  return psutil.cpu_percent()

# Create a loop to continuously monitor the CPU usage
while True:
  # Get the CPU usage
  cpu_usage = get_cpu_usage()

  # Print the CPU usage to the console
  print(f"CPU usage: {cpu_usage}%")

  # Sleep for 1 second
  time.sleep(1)
```

```
CPU usage: 1.1%
CPU usage: 5.5%
CPU usage: 2.0%
```

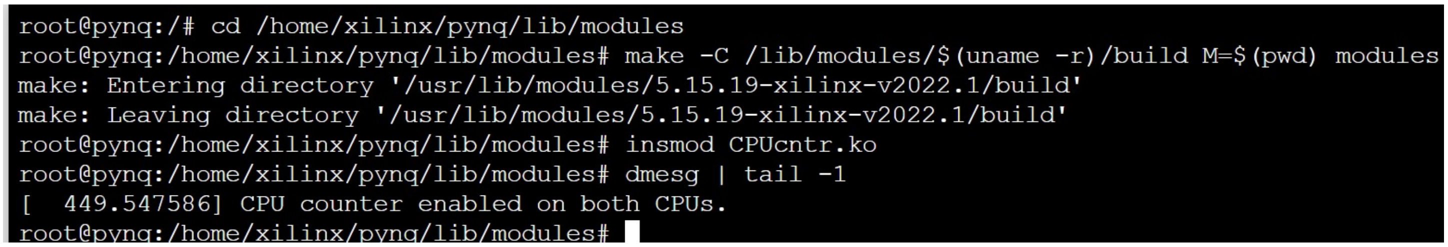
• Isolate CPU 1 by editing the bootargs (see lab work part 1)



```
Zynq> editenv bootargs
edit: 'console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk rootfstype=ext4 rootwait devtmpfs.mount=1 uio_pdrv_genirq.of_id="generic
-uio" clk_ignore_unused isolcpus=1 && bootz 0x03000000 - 0x02A00000'
Zynq> echo $bootargs
console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk rootfstype=ext4 rootwait devtmpfs.mount=1 uio_pdrv_genirq.of_id="generic-uio" c
lk_ignore_unused isolcpus=1 && bootz 0x03000000 - 0x02A00000
Zynq> boot
```

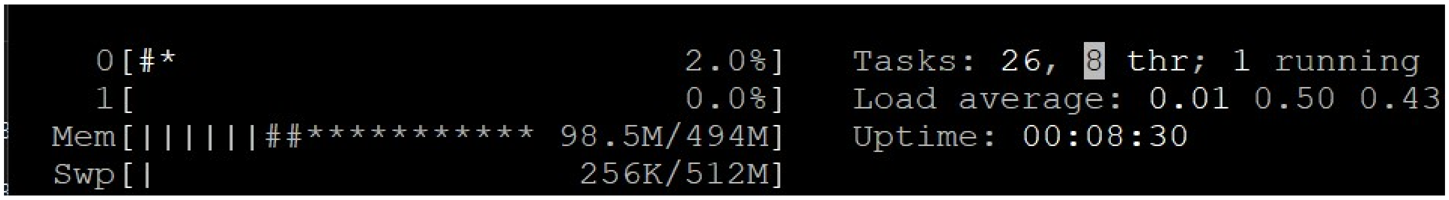
• Insert the CPUcntr kernel object onto both cpus using the instructions from lab

```
cd /home/xilinx/pynq/lib/modules
insmod CPUcntr.ko
dmesg | tail -1
```



```
root@pynq:/# cd /home/xilinx/pynq/lib/modules
root@pynq:/home/xilinx/pynq/lib/modules# make -C /lib/modules/$(uname -r)/build M=$(pwd) modules
make: Entering directory '/usr/lib/modules/5.15.19-xilinx-v2022.1/build'
make: Leaving directory '/usr/lib/modules/5.15.19-xilinx-v2022.1/build'
root@pynq:/home/xilinx/pynq/lib/modules# insmod CPUcntr.ko
root@pynq:/home/xilinx/pynq/lib/modules# dmesg | tail -1
[  449.547586] CPU counter enabled on both CPUs.
root@pynq:/home/xilinx/pynq/lib/modules#
```

To make sure CPU1 is isolated:

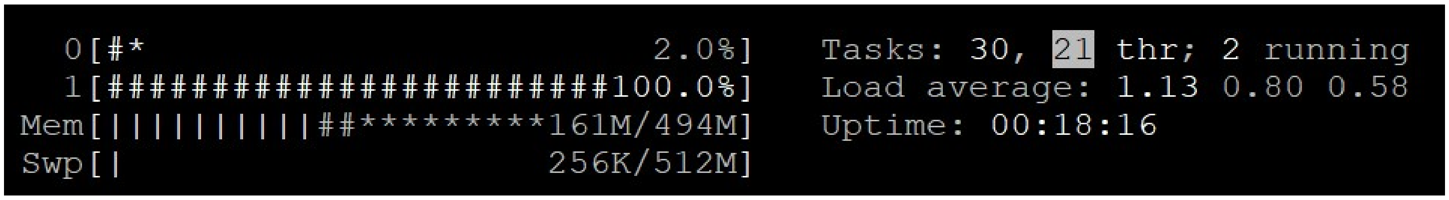


```
    0[#*                           2.0%]   Tasks: 26, 8 thr; 1 running
    1[                             0.0%]   Load average: 0.01 0.50 0.43
  Mem[||||||##*********** 98.5M/494M]   Uptime: 00:08:30
  Swp[|                        256K/512M]
```

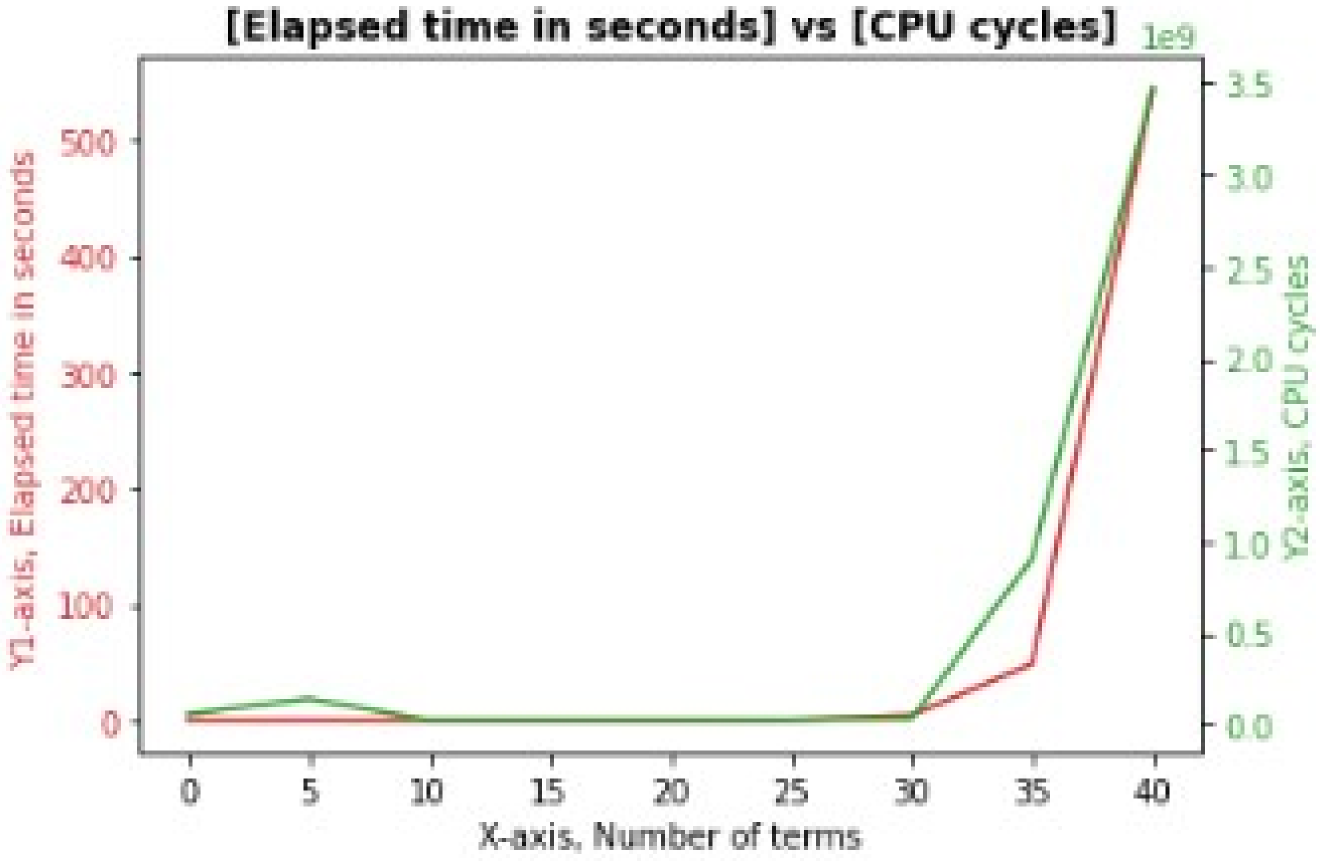
• Write code to do the following using the **recur_fibo** function from lab

- Initialize the cyclecounter
- Get the 'before' time using the python time module
- Get the 'before' cycle count
- Run the recur_fibo function on a CPU 1
- Get the 'after' cycle count
- Get the 'after' time count using the python time module
- Get the cycle count and the amount of time used

CPU1 is 100% Specially when number of terms is above 30

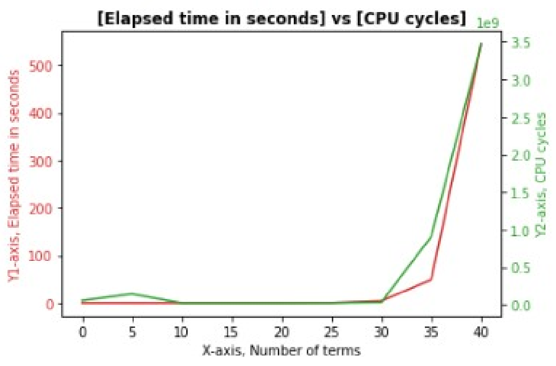


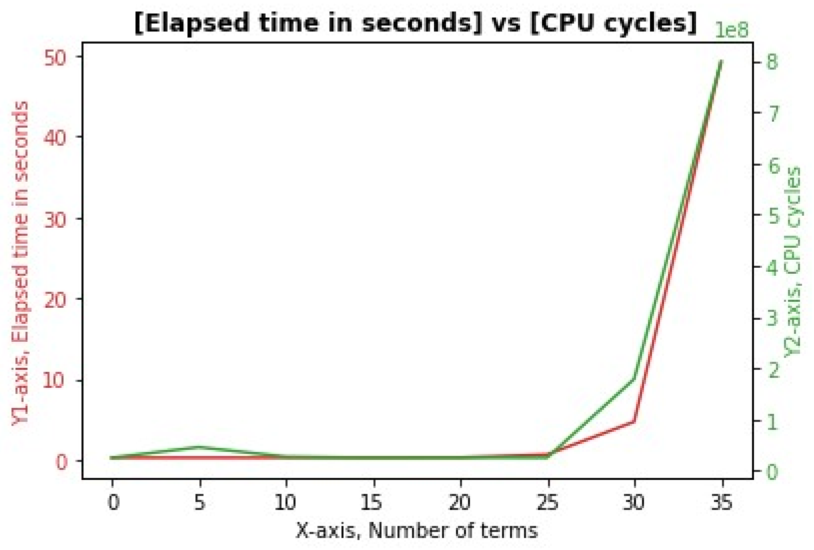
```
    0[#*                           2.0%]   Tasks: 30, 21 thr; 2 running
    1[#########################100.0%]   Load average: 1.13 0.80 0.58
  Mem[||||||||||##*********161M/494M]   Uptime: 00:18:16
  Swp[|                        256K/512M]
```


[Elapsed time in seconds] vs [CPU cycles]
1e9
Y1-axis, Elapsed time in seconds
Y2-axis, CPU cycles
X-axis, Number of terms
0
100
200
300
400
500
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
5
10
15
20
25
30
35
40

```python
# Part A3.2: Comparing and Gathering Data, (running fibonacci sequence)
%reset -f
CPU, TERMSary, ELAPSEDary, CYCLESary = "1", [], [], []

import os, ctypes, time, math;import matplotlib.pyplot as plt
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))
for NumberOfTerms in range(0,45,5):
    cmd = "taskset -c "+CPU+" python3 fib.py "+ str(NumberOfTerms);print (cmd)
    start = time.time()
    _libInC.init_cntrs(1,0)
    os.system(cmd)
    cycles = ctypes.c_uint(_libInC.gcyclec()).value
    end = time.time();elapsed = end - start;elapsed = round(elapsed, 7)
    print(cmd+"-> NumberOfTerms: "+ str(NumberOfTerms)+ " -> Elapsed time = " +str(elapsed)\
          +", cycles = "+str(cycles))
    TERMSary.append(NumberOfTerms);ELAPSEDary.append(elapsed);CYCLESary.append(cycles)

x = TERMSary
dataset_1 = ELAPSEDary
dataset_2 = CYCLESary
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('X-axis, Number of terms')
ax1.set_ylabel('Y1-axis, Elapsed time in seconds', color = color)
ax1.plot(x, dataset_1, color = color)
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx() # Adding Twin Axes to plot using dataset_2
color = 'tab:green'
ax2.set_ylabel('Y2-axis, CPU cycles', color = color)
ax2.plot(x, dataset_2, color = color)
ax2.tick_params(axis ='y', labelcolor = color)
plt.title('[Elapsed time in seconds] vs [CPU cycles]', fontweight ="bold")
plt.show()
```

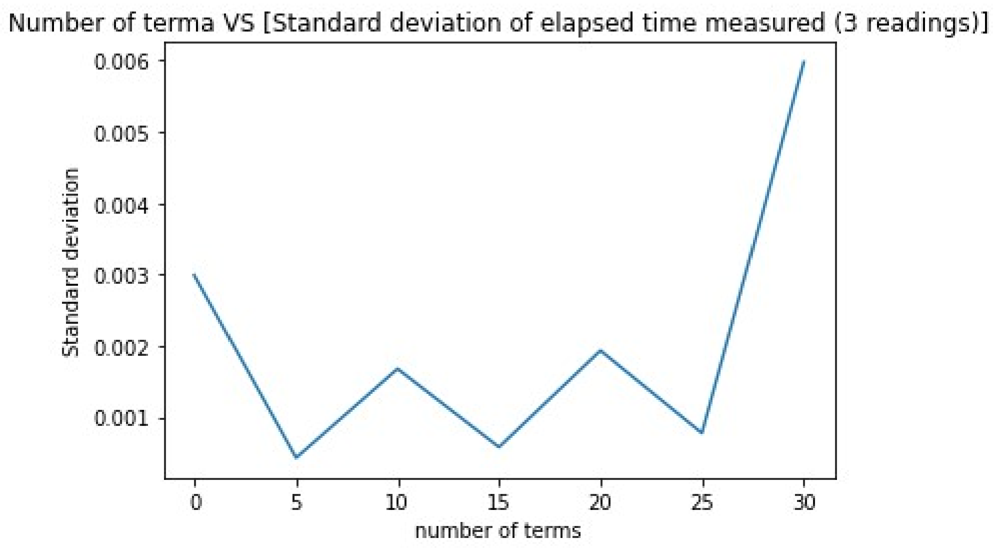
```
Library version: 671
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 4.506111145019531e-05
taskset -c 1 python3 fib.py 0-> NumberOfTerms: 0 -> Elapsed time = 0.289243, cycles = 58872844
taskset -c 1 python3 fib.py 5
time spent: 4.6253204345703125e-05
taskset -c 1 python3 fib.py 5-> NumberOfTerms: 5 -> Elapsed time = 0.3184817, cycles = 145038039
taskset -c 1 python3 fib.py 10
time spent: 0.0003380775451660156
taskset -c 1 python3 fib.py 10-> NumberOfTerms: 10 -> Elapsed time = 0.281321, cycles = 25079257
taskset -c 1 python3 fib.py 15
time spent: 0.0035219192504882812
taskset -c 1 python3 fib.py 15-> NumberOfTerms: 15 -> Elapsed time = 0.2841513, cycles = 25010367
taskset -c 1 python3 fib.py 20
time spent: 0.035887956619262695
taskset -c 1 python3 fib.py 20-> NumberOfTerms: 20 -> Elapsed time = 0.3173056, cycles = 25031988
taskset -c 1 python3 fib.py 25
time spent: 0.395953893661499
taskset -c 1 python3 fib.py 25-> NumberOfTerms: 25 -> Elapsed time = 0.6770241, cycles = 27422230
taskset -c 1 python3 fib.py 30
time spent: 4.435415029525757
taskset -c 1 python3 fib.py 30-> NumberOfTerms: 30 -> Elapsed time = 4.7165933, cycles = 34395945
taskset -c 1 python3 fib.py 35
time spent: 48.906649112701416
taskset -c 1 python3 fib.py 35-> NumberOfTerms: 35 -> Elapsed time = 49.187216, cycles = 908208419
taskset -c 1 python3 fib.py 40
time spent: 543.6587641239166
taskset -c 1 python3 fib.py 40-> NumberOfTerms: 40 -> Elapsed time = 543.9400814, cycles = 3468163212
```





• Vary the number of terms from 1 to 30 **as you see fit** to compare the different execution times

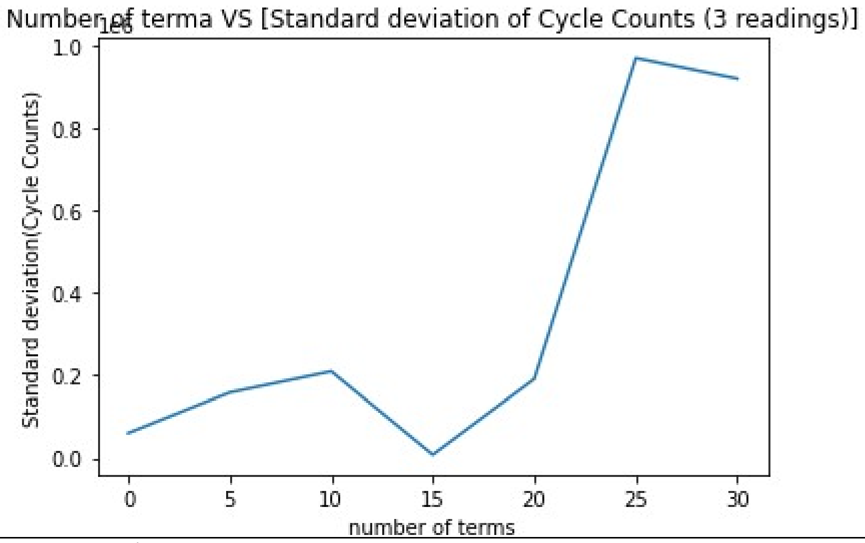
```
Nterms = 1,  CPU1  usage 17%, time spent: 0.28 sec
Nterms = 5,  CPU1  usage 18%, time spent: 0.29 sec
Nterms = 10, CPU1  usage 18%, time spent: 0.298 sec
Nterms = 15, CPU1  usage 22%, time spent: 0.30 sec
Nterms = 20, CPU1  usage 34%, time spent: 0.32 sec
Nterms = 25, CPU1  usage 47%, time spent: 0.69 sec
Nterms = 30, CPU1  usage 97%, time spent: 4.8 sec
```



```
ibrary version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 4.100799560546875e-05
Please enter a positive integer
time spent: 3.9577484130859375e-05
Please enter a positive integer
time spent: 4.38690185546875e-05
3 Readings : [0.2823057174682617, 0.2807629108428955, 0.2877347469329834]
taskset -c 1 python3 fib.py 5
time spent: 4.38690185546875e-05
time spent: 4.291534423828125e-05
time spent: 4.649162292480469e-05
3 Readings : [0.28080177307128906, 0.28180789947509766, 0.2810330390930176]
taskset -c 1 python3 fib.py 10
time spent: 0.00033783912658691406
time spent: 0.0003435611724853516
time spent: 0.0003502368927001953
3 Readings : [0.28055715560913086, 0.2814812660217285, 0.2844858169555664]
taskset -c 1 python3 fib.py 15
time spent: 0.003372669219970703
time spent: 0.003466367721557617
time spent: 0.0033731460571289062
3 Readings : [0.28441858291625977, 0.28578996658325195, 0.2847864627838135]
taskset -c 1 python3 fib.py 20
time spent: 0.03591465950012207
time spent: 0.03592538833618164
time spent: 0.0359797477722168
```

```
3 Readings : [0.31665730476379395, 0.32128024101257324, 0.31807374954223633]
taskset -c 1 python3 fib.py 25
time spent: 0.3961641788482666
time spent: 0.3962666988372803
time spent: 0.3959677219390869
3 Readings : [0.6780669689178467, 0.6773550510406494, 0.6761841773986816]
taskset -c 1 python3 fib.py 30
time spent: 4.457427740097046
time spent: 4.44278883934021
time spent: 4.437021017074585
3 Readings : [4.73832368850708, 4.72719144821167, 4.724515676498413]
all nTerms: [0, 5, 10, 15, 20, 25, 30]
all    avg: [0.2836011250813802, 0.28121423721313477, 0.2821747461954753, 0.2849983374277751, 0.31867043177286786,
0.677202065785726, 4.730010271072388]
all StdDev: [0.00299000393309184, 0.0004302690275945236, 0.001677156533596681, 0.0005795638814882688, 0.0019338919572256263,
0.0007762213354140496, 0.005979109222838678]
```

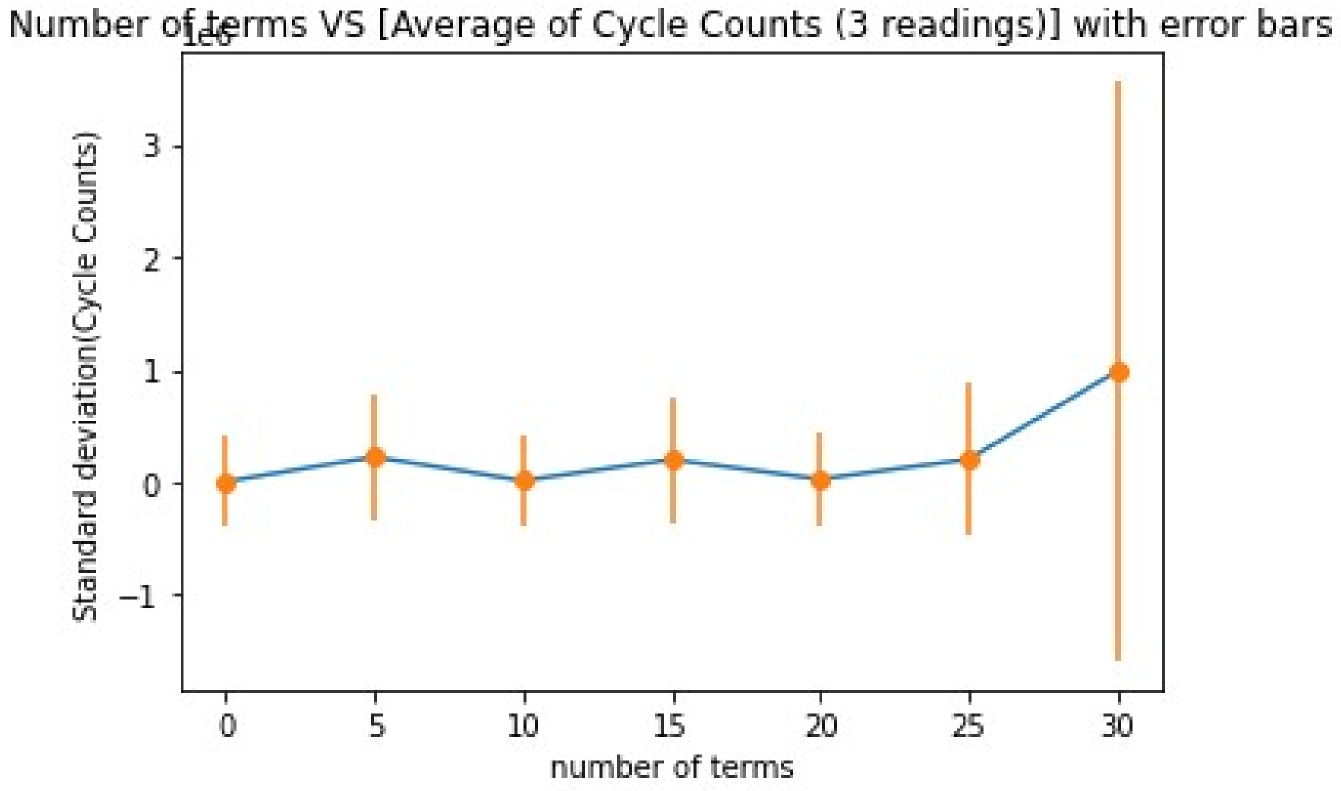
– Take multiple trials for each variation (i.e. get three cyclecounts for n=5, then get three cyclecounts for n=10, etc) and average the different trials.



```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.910064697265625e-05
Please enter a positive integer
time spent: 4.0531158447265625e-05
Please enter a positive integer
time spent: 3.9577484130859375e-05
3 Readings : [537585, 390594, 447519]
taskset -c 1 python3 fib.py 5
time spent: 4.3392181396484375e-05
time spent: 4.458427429199219e-05
time spent: 4.3392181396484375e-05
3 Readings : [724288, 385227, 389008]
taskset -c 1 python3 fib.py 10
time spent: 0.00033855438232421875
time spent: 0.0003442764282226562 5
time spent: 0.0003383159637451172
3 Readings : [387095, 387985, 833363]
taskset -c 1 python3 fib.py 15
time spent: 0.003408908843994l406
time spent: 0.003468751907348633
time spent: 0.0034089088439941406
3 Readings : [405651, 389720, 386964]
taskset -c 1 python3 fib.py 20
time spent: 0.03592491149902344
time spent: 0.0360262393951416
time spent: 0.03585457801818848
3 Readings : [384608, 812256, 430449]
taskset -c 1 python3 fib.py 25
time spent: 0.40308666229248047
time spent: 0.3961191177368164
time spent: 0.3960273265838623
3 Readings : [2646753, 868044, 394107]
taskset -c 1 python3 fib.py 30
time spent: 4.44086766242981
time spent: 4.439512252807617
time spent: 4.4548540115356445
3 Readings : [1754083, 2336424, 3929933]
all nTerms: [0, 5, 10, 15, 20, 25, 30]
all    avg: [458566.0, 499507.6666666667, 536147.6666666666, 394111.6666666667, 542437.6666666666,
1302968.0, 2673480.0]
all StdDev: [60515.09826481322, 158951.193096777, 210163.29175402847, 8236.74872480371,
191706.02333144244, 969698.4633245533, 919704.9766010112]
```

– The error for each ‘n’ will be the standard deviation from the mean which is the standard deviation of all the trials divided by the square root of the number of trials.

• Plot the average results for varying ‘n’ along with error bars of your measurments.

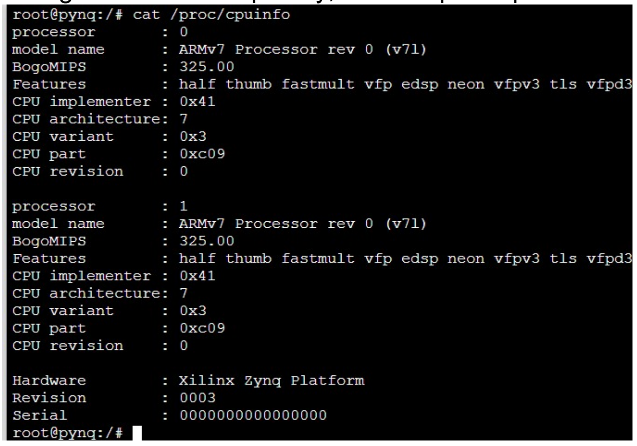


```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.981590270996094e-05
Please enter a positive integer
time spent: 3.9577484130859375e-05
Please enter a positive integer
time spent: 4.100799560546875e-05
3 Readings : [410606, 394479, 397248]
taskset -c 1 python3 fib.py 5
time spent: 4.315376281738281e-05
time spent: 4.363059997558594e-05
time spent: 4.506111145019531e-05
3 Readings : [405537, 873139, 386854]
taskset -c 1 python3 fib.py 10
time spent: 0.0003364086151123047
time spent: 0.0003380775451660156
time spent: 0.00034499168395996094
3 Readings : [386149, 395762, 422180]
taskset -c 1 python3 fib.py 15
time spent: 0.003447294235229492
time spent: 0.003374814987182617
time spent: 0.003436088562011719
3 Readings : [430528, 845592, 407685]
```

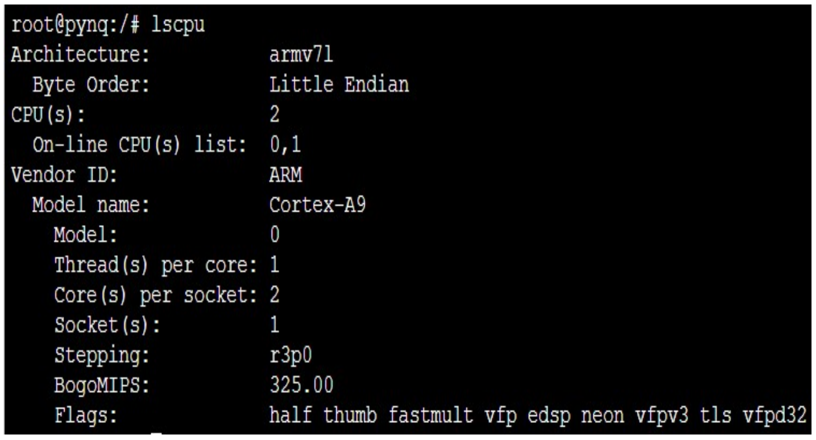
```
taskset -c 1 python3 fib.py 20
time spent: 0.0359346866607666
time spent: 0.03584432601928711
time spent: 0.0359339714050293
3 Readings : [389444, 389000, 448988]
taskset -c 1 python3 fib.py 25
time spent: 0.39617252349853516
time spent: 0.3960864543914795
time spent: 0.3964650630950928
3 Readings : [855102, 392993, 787929]
taskset -c 1 python3 fib.py 30
time spent: 4.458937644958496
time spent: 4.440706968307495
time spent: 4.440298318862915
3 Readings : [3962897, 1923679, 1824311]
all nTerms: [0, 5, 10, 15, 20, 25, 30]
all    avg: [400777.6666666667, 555176.6666666666, 401363.6666666667, 561268.3333333334, 409144.0,
678674.6666666666, 2570295.6666666665]
all StdDev: [7041.01992296255, 224962.66014064546, 15233.564746601134, 201263.36184600406,
28174.545675130237, 203860.350868486, 985553.095155992]
```

• In order to compare the timing module and PMU counting, we need them to be in the same units.

– To get the CPU frequency, run cat /proc/cpuinfo in a new terminal or run lscpu



```
root@pynq:/# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 0 (v7l)
BogoMIPS        : 325.00
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpd3
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x3
CPU part        : 0xc09
CPU revision    : 0

processor       : 1
model name      : ARMv7 Processor rev 0 (v7l)
BogoMIPS        : 325.00
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpd3
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x3
CPU part        : 0xc09
CPU revision    : 0

Hardware        : Xilinx Zynq Platform
Revision        : 0003
Serial          : 0000000000000000
root@pynq:/#
```

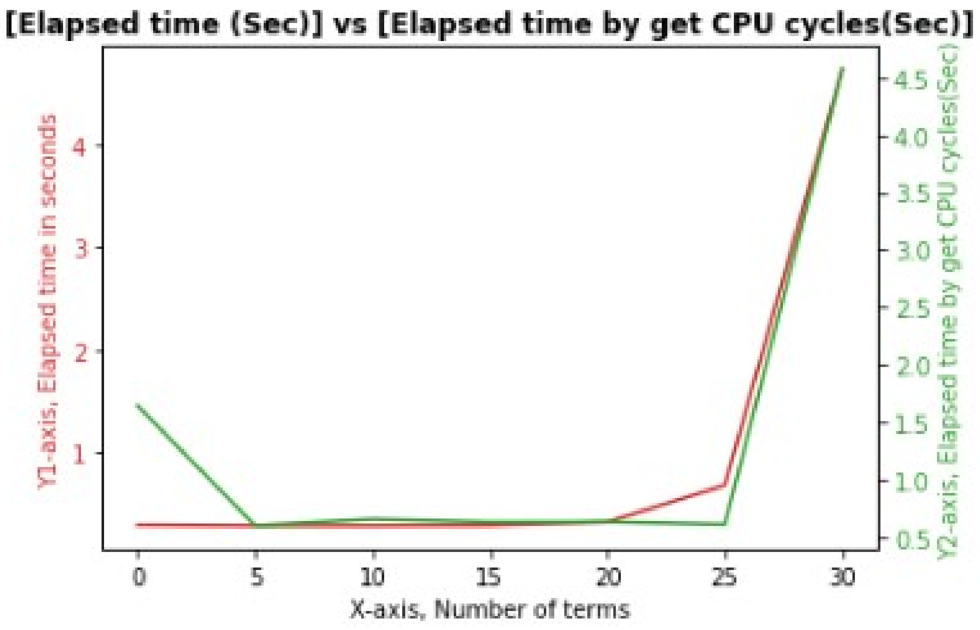


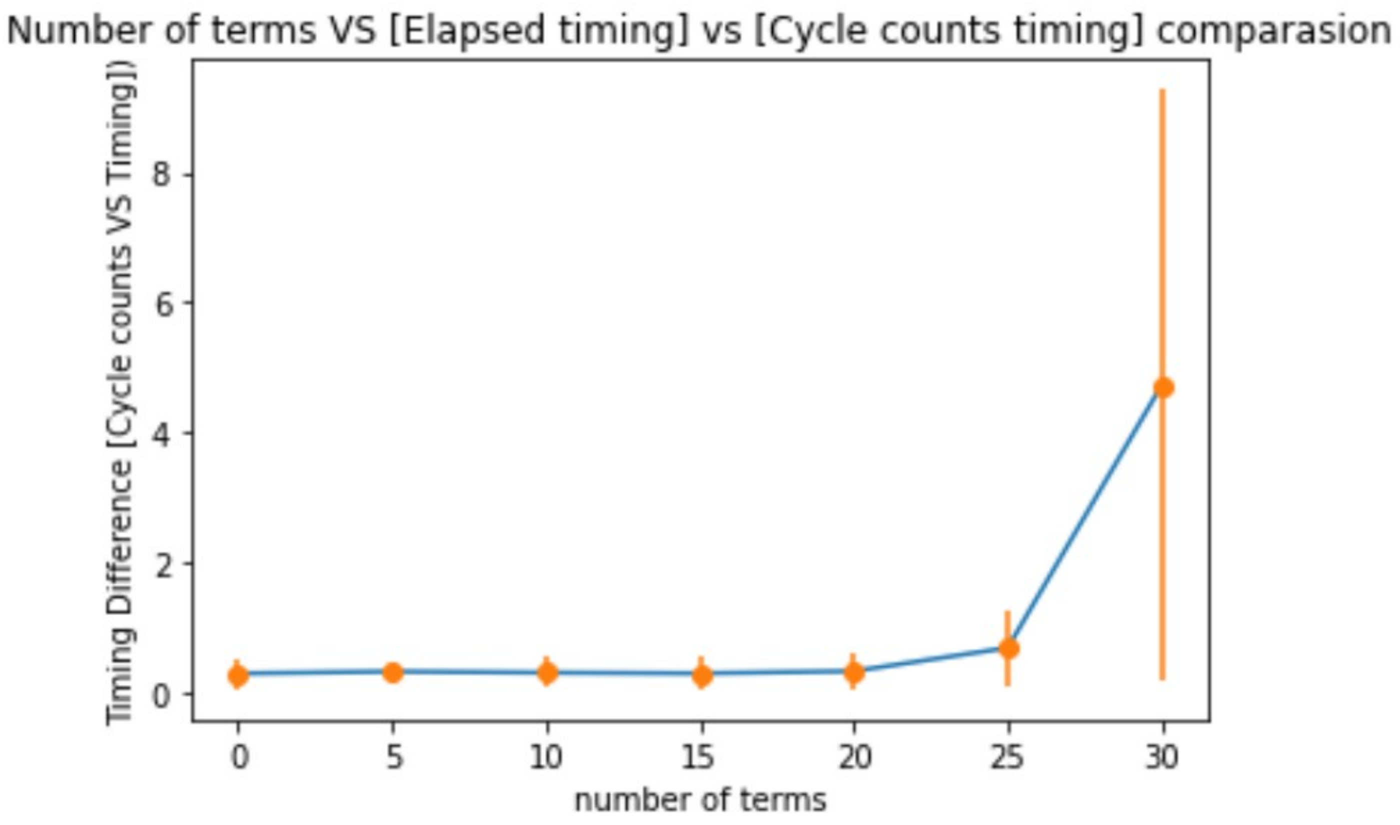
```
root@pynq:/# lscpu
Architecture:          armv7l
  Byte Order:          Little Endian
CPU(s):                2
  On-line CPU(s) list: 0,1
Vendor ID:             ARM
  Model name:          Cortex-A9
    Model:             0
    Thread(s) per core: 1
    Core(s) per socket: 2
    Socket(s):         1
    Stepping:          r3p0
    BogoMIPS:          325.00
    Flags:             half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
```

– Use this frequency to convert the PMU output from clock counts to timing

BogoMips (from "bogus" and MIPS) is a crude measurement of CPU speed made by the Linux kernel when it boots to calibrate an internal busy-loop. An often-quoted definition of the term is "**the number of million times per second a processor can do absolutely nothing**".

– Compare the timing of the PMU counter to the timing module





Deliverables

Each student must submit the following individually
1
• A PDF report detailing your work flow and relative Jupyter notebook cells relating to your progress.
– Your report should detail your work flow throughout the assignment. Be sure to discuss any difficulties or troubles you encountered and your troubleshooting procedure. Also, detail your thought process which led to the design and implementation of the code. For example, describe your top-down design methodology (i.e. how did you split the large task into smaller, more incremental jobs? How were you able to test each of these smaller parts?)
– Please also include a plot with errorbars where the x-axis is the number of terms calculated (n) and the y-axis is the cyclecount
– Please also include a plot with errorbars where the x-axis is the number of terms calculated (n) and the y-axis is the time taken to complete
• Your complete Jupyter notebook, downloaded as a PDF attached at the very end of your report
– You can do this by selecting 'File -> Print Preview' then printing to PDF from the browser.
– Use a PDF stitching tool like pdfjoiner to join your Report and Jupyter Notebook into a single PDF file
Each team should submit the following one per team
• All relevant code (.ipynb, .py, .cpp, .c, etc files) pushed to your team's git repo.

In [ ]:
```
# from lab3
import os
cpu = input('enter CPU number (0/1)?')
str = "taskset -c "+cpu+" python3 fib.py"
print(str)
os.system(str)
```

In [ ]:
```
# use psutil
import psutil, time

# Create a function to get the CPU usage
def get_cpu_usage():
  """Returns the CPU usage as a percentage."""
  return psutil.cpu_percent()

# Create a loop to continuously monitor the CPU usage
while True:
  # Get the CPU usage
  cpu_usage = get_cpu_usage()

  # Print the CPU usage to the console
  print(f"CPU usage: {cpu_usage}%")

  # Sleep for 1 second
  time.sleep(1)
```

In [22]:
```
#Implement ELAPSED TIME
import time
start = time.time();print(start)
#<<<< tratar fibonacci con argumento de entrada que tome valores 1-30
time.sleep(1)
end = time.time();print(end)
print(end - start)
```

```
1667142589.5592985
1667142590.5779364
1.0186378955841064
```

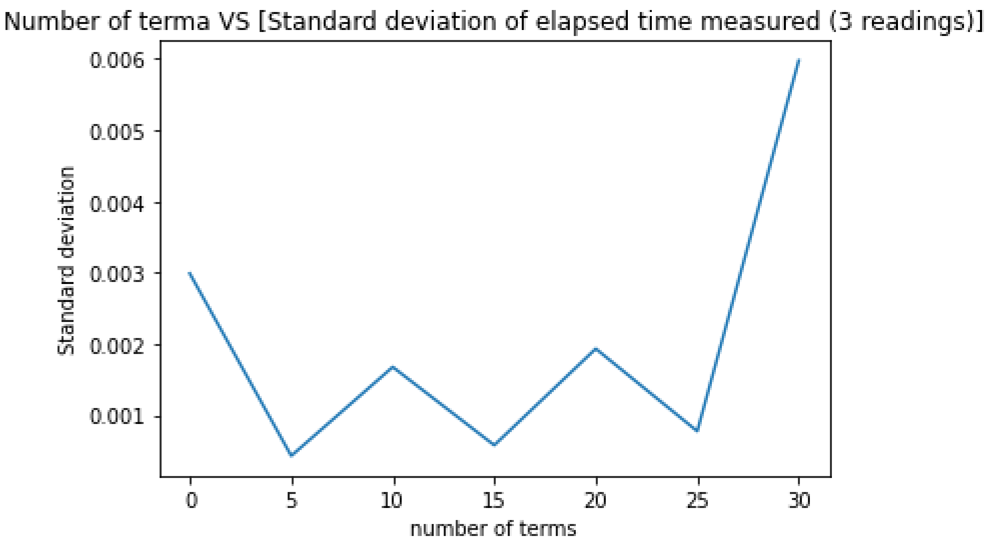
In [1]:
```
#new /clock_example0/cycletime2.c version2, clock_example0
# https://docs.python.org/3/library/ctypes.html

%reset -f
import ctypes, time
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))
_libInC.init_cntrs(1,1)#reset=1, enable_divider=1
start = ctypes.c_uint(_libInC.gcyclec()).value
time.sleep(1)
stop = ctypes.c_uint(_libInC.gcyclec()).value
elapsed = stop-start
print("  start: "+str(start)+"\n"\
      "   stop: "+str(stop)+"\n"\
      "elapsed: "+str(elapsed))
```

```
Library version: 700
  start: 5858
   stop: 900652
elapsed: 894794
```

In [37]:
```
#@@@new /clock_example0/cycletime2.c version2, clock_example0
# https://docs.python.org/3/library/ctypes.html
# add python time and run fibonacci script, standard deviation ELAPSED TIME
%reset -f
import os, ctypes, time, math;import matplotlib.pyplot as plt
N_TERM_Sary, avgEXEC_TIMEary, stddevEXEC_TIMEary = [], [], []
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))

# Standard deviation of list, Using sum() + list comprehension
def Stddev(test_list):
    mean = sum(test_list) / len(test_list)
    variance = sum([((x - mean) ** 2) for x in test_list]) / len(test_list)
    res = variance ** 0.5
    return res

_libInC.init_cntrs(1,1)#reset=1, enable_divider=1
for NumberOfTerms in range(0,35,5):
    N_TERM_Sary.append(NumberOfTerms)
    cmd = "taskset -c 1 python3 fib.py "+ str(NumberOfTerms);print (cmd)
    EXEC_TIMEary = []
    for ith in range(3):
        start1 = ctypes.c_uint(_libInC.gcyclec()).value
        start2 = time.time()
        os.system(cmd)
        stop1 = ctypes.c_uint(_libInC.gcyclec()).value
        stop2 = time.time()
        elapsed1 = stop1-start1
        elapsed2 = stop2-start2
        '''
        print("Cycles elapsed (cycles), getCycles():")
        print("  start1: "+str(start1)+"\n"\
              "   stop1: "+str(stop1)+"\n"\
              "elapsed1: "+str(elapsed1))
        print("Time elapsed (sec), using Python time:")
        print("  start2: "+str(start2)+"\n"\
              "   stop2: "+str(stop2)+"\n"\
              "elapsed2: "+str(elapsed2))
        '''
        EXEC_TIMEary.append(elapsed2)

    sdev = Stddev(EXEC_TIMEary)
    avg = sum(EXEC_TIMEary) / len(EXEC_TIMEary)
    print("3 Readings : " + str(EXEC_TIMEary))
    stddevEXEC_TIMEary.append(sdev)
    avgEXEC_TIMEary.append(avg)
print("all nTerms: " + str(N_TERM_Sary))
print("all    avg: " + str(avgEXEC_TIMEary))
print("all StdDev: " + str(stddevEXEC_TIMEary))

import matplotlib.pyplot as plt
x = N_TERM_Sary
y = stddevEXEC_TIMEary
plt.title("Number of terma VS [Standard deviation of elapsed time measured (3 readings
plt.xlabel("number of terms")
plt.ylabel("Standard deviation")
plt.plot(x, y)
```

```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 4.100799560546875e-05
Please enter a positive integer
time spent: 3.9577484130859375e-05
Please enter a positive integer
time spent: 4.38690185546875e-05
3 Readings : [0.2823057174682617, 0.2807629108428955, 0.2877347469329834]
taskset -c 1 python3 fib.py 5
time spent: 4.38690185546875e-05
time spent: 4.291534423828125e-05
time spent: 4.649162292480469e-05
3 Readings : [0.28080177307128906, 0.28180789947509766, 0.2810330390930176]
taskset -c 1 python3 fib.py 10
time spent: 0.00033783912658691406
time spent: 0.00034356117248535156
time spent: 0.0003502368927001953
3 Readings : [0.28055715560913086, 0.2814812660217285, 0.2844858169555664]
taskset -c 1 python3 fib.py 15
time spent: 0.003372669219970703
time spent: 0.003466367721557617
time spent: 0.0033731460571289062
3 Readings : [0.28441858291625977, 0.28578996658325195, 0.2847864627838135]
taskset -c 1 python3 fib.py 20
time spent: 0.03591465950012207
time spent: 0.03592538833618164
time spent: 0.0359797477722168
3 Readings : [0.31665730476379395, 0.32128024101257324, 0.31807374954223633]
taskset -c 1 python3 fib.py 25
time spent: 0.3961641788482666
time spent: 0.3962666988372803
time spent: 0.3959677219390869
3 Readings : [0.6780669689178467, 0.6773550510406494, 0.6761841773986816]
taskset -c 1 python3 fib.py 30
time spent: 4.457427740097046
time spent: 4.44278883934021
time spent: 4.437021017074585
3 Readings : [4.73832368850708, 4.72719144821167, 4.724515676498413]
all nTerms: [0, 5, 10, 15, 20, 25, 30]
all    avg: [0.2836011250813802, 0.28121423721313477, 0.2821747461954753, 0.284998337
4277751, 0.31867043177286786, 0.677202065785726, 4.730010271072388]
all StdDev: [0.00299000393309184, 0.0004302690275945236, 0.001677156533596681, 0.0005
795638814882688, 0.0019338919572256263, 0.0007762213354140496, 0.005979109222838678]
```

```
Out[37]: [<matplotlib.lines.Line2D at 0xacae2250>]
```

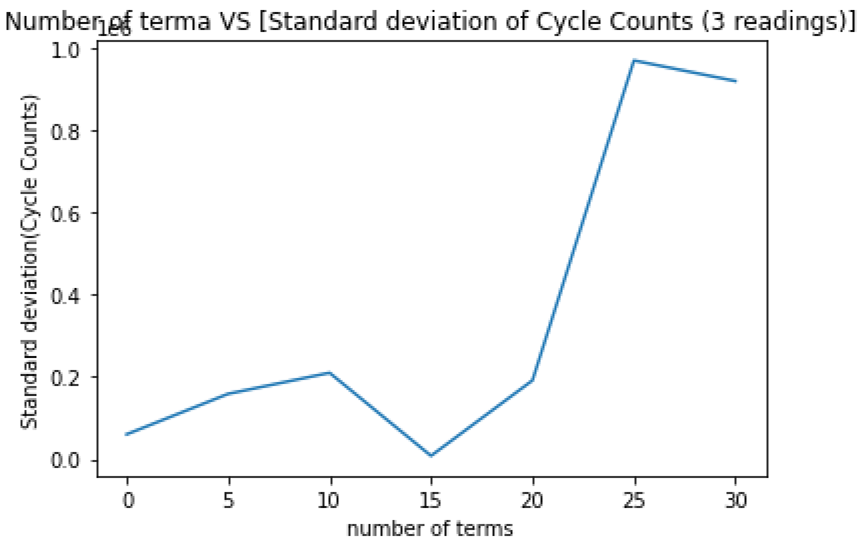


In [38]:
```
#@@@new /clock_example0/cycletime2.c version2, clock_example0
# https://docs.python.org/3/library/ctypes.html
# add python time and run fibonacci script, standard deviation ELAPSED cycleCounts
%reset -f
import os, ctypes, time, math;import matplotlib.pyplot as plt
N_TERM_Sary, avgEXEC_TIMEary, stddevEXEC_TIMEary = [], [], []
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))

# Standard deviation of list, Using sum() + list comprehension
def Stddev(test_list):
    mean = sum(test_list) / len(test_list)
    variance = sum([((x - mean) ** 2) for x in test_list]) / len(test_list)
    res = variance ** 0.5
    return res

_libInC.init_cntrs(1,1)#reset=1, enable_divider=1
for NumberOfTerms in range(0,35,5):
    N_TERM_Sary.append(NumberOfTerms)
    cmd = "taskset -c 1 python3 fib.py "+ str(NumberOfTerms);print (cmd)
    EXEC_TIMEary = []
    for ith in range(3):
        start1 = ctypes.c_uint(_libInC.gcyclec()).value
        start2 = time.time()
        os.system(cmd)
        stop1 = ctypes.c_uint(_libInC.gcyclec()).value
        stop2 = time.time()
        elapsed1 = stop1-start1
        elapsed2 = stop2-start2
        EXEC_TIMEary.append(elapsed1)

    sdev = Stddev(EXEC_TIMEary)
    avg = sum(EXEC_TIMEary) / len(EXEC_TIMEary)
    print("3 Readings : " + str(EXEC_TIMEary))
    stddevEXEC_TIMEary.append(sdev)
    avgEXEC_TIMEary.append(avg)
print("all nTerms: " + str(N_TERM_Sary))
print("all    avg: " + str(avgEXEC_TIMEary))
print("all StdDev: " + str(stddevEXEC_TIMEary))

import matplotlib.pyplot as plt
x = N_TERM_Sary
```

```
y = stddevEXEC_TIMEary
plt.title("Number of terma VS [Standard deviation of Cycle Counts (3 readings)]")
plt.xlabel("number of terms")
plt.ylabel("Standard deviation(Cycle Counts)")
plt.plot(x, y)
```

```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.910064697265625e-05
Please enter a positive integer
time spent: 4.0531158447265625e-05
Please enter a positive integer
time spent: 3.9577484130859375e-05
3 Readings : [537585, 390594, 447519]
taskset -c 1 python3 fib.py 5
time spent: 4.3392181396484375e-05
time spent: 4.458427429199219e-05
time spent: 4.3392181396484375e-05
3 Readings : [724288, 385227, 389008]
taskset -c 1 python3 fib.py 10
time spent: 0.00033855438232421875
time spent: 0.00034427642822265625
time spent: 0.0003383159637451172
3 Readings : [387095, 387985, 833363]
taskset -c 1 python3 fib.py 15
time spent: 0.0034089088439941406
time spent: 0.003468751907348633
time spent: 0.0034089088439941406
3 Readings : [405651, 389720, 386964]
taskset -c 1 python3 fib.py 20
time spent: 0.03592491149902344
time spent: 0.0360262393951416
time spent: 0.03585457801818848
3 Readings : [384608, 812256, 430449]
taskset -c 1 python3 fib.py 25
time spent: 0.40308666229248047
time spent: 0.3961191177368164
time spent: 0.3960273265838623
3 Readings : [2646753, 868044, 394107]
taskset -c 1 python3 fib.py 30
time spent: 4.44086766242981
time spent: 4.439512252807617
time spent: 4.4548540115356445
3 Readings : [1754083, 2336424, 3929933]
all nTerms: [0, 5, 10, 15, 20, 25, 30]
all    avg: [458566.0, 499507.6666666667, 536147.6666666666, 394111.6666666667, 54243
7.6666666666, 1302968.0, 2673480.0]
all StdDev: [60515.09826481322, 158951.193096777, 210163.29175402847, 8236.7487248037
1, 191706.02333144244, 969698.4633245533, 919704.9766010112]
```

```
Out[38]: [<matplotlib.lines.Line2D at 0xac9e7d78>]
```



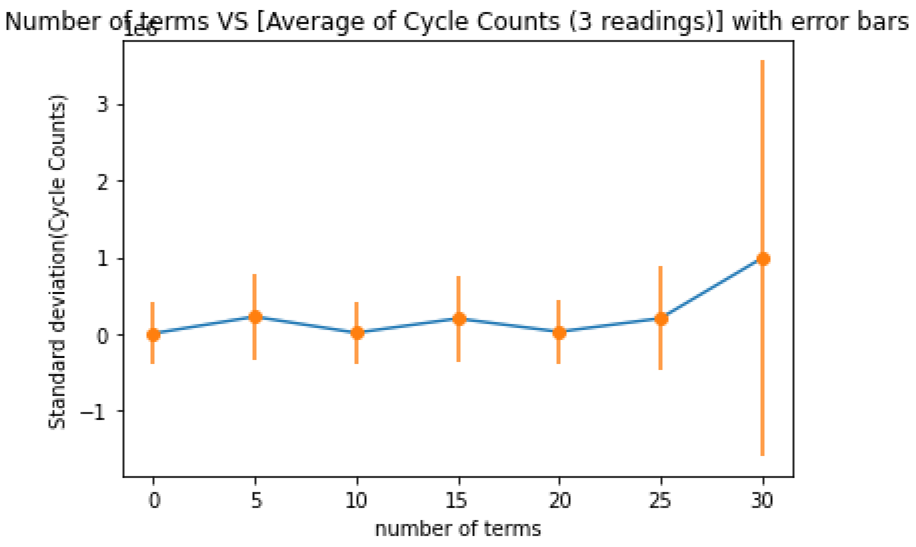
In [2]:
```
#@@@new /clock_example0/cycletime2.c version2, clock_example0
# https://docs.python.org/3/library/ctypes.html
# add python time and run fibonacci script, std deviation ELAPSED cycleCounts plus err
%reset -f
import os, ctypes, time, math;import matplotlib.pyplot as plt
N_TERM_Sary, avgEXEC_TIMEary, stddevEXEC_TIMEary = [], [], []
_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))

# Standard deviation of list, Using sum() + list comprehension
def Stddev(test_list):
    mean = sum(test_list) / len(test_list)
    variance = sum([((x - mean) ** 2) for x in test_list]) / len(test_list)
    res = variance ** 0.5
    return res

_libInC.init_cntrs(1,1)#reset=1, enable_divider=1
for NumberOfTerms in range(0,35,5):
    N_TERM_Sary.append(NumberOfTerms)
    cmd = "taskset -c 1 python3 fib.py "+ str(NumberOfTerms);print (cmd)
    EXEC_TIMEary = []
    for ith in range(3):
        start1 = ctypes.c_uint(_libInC.gcyclec()).value
        start2 = time.time()
        os.system(cmd)
        stop1 = ctypes.c_uint(_libInC.gcyclec()).value
        stop2 = time.time()
        elapsed1 = stop1-start1
        elapsed2 = stop2-start2
        EXEC_TIMEary.append(elapsed1)

    sdev = Stddev(EXEC_TIMEary)
    avg = sum(EXEC_TIMEary) / len(EXEC_TIMEary)
    print("3 Readings : " + str(EXEC_TIMEary))
    stddevEXEC_TIMEary.append(sdev)
    avgEXEC_TIMEary.append(avg)
print("all nTerms: " + str(N_TERM_Sary))
print("all    avg: " + str(avgEXEC_TIMEary))
print("all StdDev: " + str(stddevEXEC_TIMEary))

import matplotlib.pyplot as plt
x = N_TERM_Sary
```

```
y = stddevEXEC_TIMEary
plt.title("Number of terms VS [Average of Cycle Counts (3 readings)] with error bars")
plt.xlabel("number of terms")
plt.ylabel("Standard deviation(Cycle Counts)")

y_error = avgEXEC_TIMEary
plt.plot(x, y)
plt.errorbar(x, y,
             yerr = y_error,
             fmt ='o')
```

```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.981590270996094e-05
Please enter a positive integer
time spent: 3.9577484130859375e-05
Please enter a positive integer
time spent: 4.100799560546875e-05
3 Readings : [410606, 394479, 397248]
taskset -c 1 python3 fib.py 5
time spent: 4.315376281738281e-05
time spent: 4.363059997558594e-05
time spent: 4.506111145019531e-05
3 Readings : [405537, 873139, 386854]
taskset -c 1 python3 fib.py 10
time spent: 0.0003364086151123047
time spent: 0.0003380775451660156
time spent: 0.00034499168395996094
3 Readings : [386149, 395762, 422180]
taskset -c 1 python3 fib.py 15
time spent: 0.003447294235229492
time spent: 0.003374814987182617
time spent: 0.0034360885620117188
3 Readings : [430528, 845592, 407685]
taskset -c 1 python3 fib.py 20
time spent: 0.0359346866607666
time spent: 0.03584432601928711
time spent: 0.0359339714050293
3 Readings : [389444, 389000, 448988]
taskset -c 1 python3 fib.py 25
time spent: 0.39617252349853516
time spent: 0.3960864543914795
time spent: 0.3964650630950928
3 Readings : [855102, 392993, 787929]
taskset -c 1 python3 fib.py 30
time spent: 4.458937644958496
time spent: 4.440706968307495
time spent: 4.440298318862915
3 Readings : [3962897, 1923679, 1824311]
all nTerms: [0, 5, 10, 15, 20, 25, 30]
all    avg: [400777.6666666667, 555176.6666666666, 401363.6666666667, 561268.33333333
34, 409144.0, 678674.6666666666, 2570295.6666666665]
all StdDev: [7041.01992296255, 224962.66014064546, 15233.564746601134, 201263.3618460
0406, 28174.545675130237, 203860.350868486, 985553.095155992]
```

Out[2]:

```
<ErrorbarContainer object of 3 artists>
```



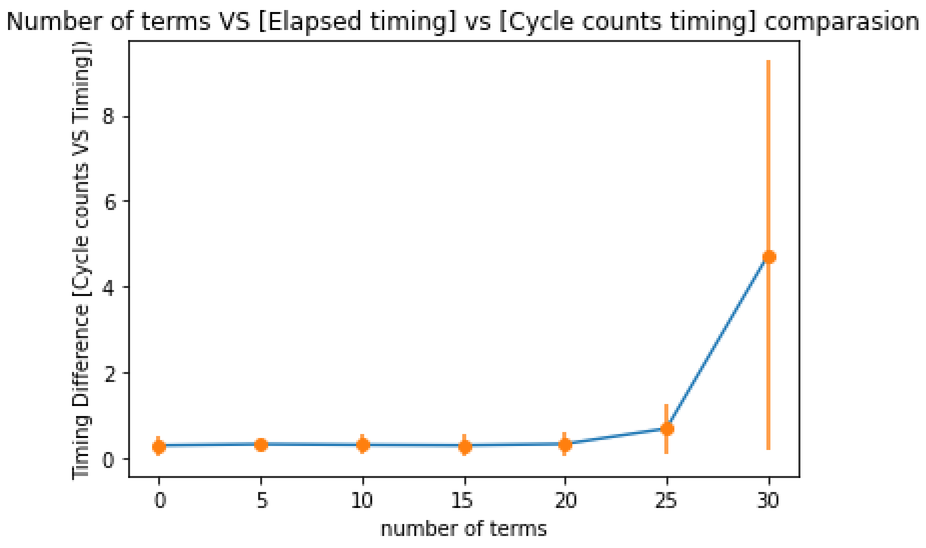
```
In [2]: #@@@new /clock_example0/cycletime2.c version2, clock_example0
        # https://docs.python.org/3/library/ctypes.html
        # add python time and run fibonacci script, std deviation ELAPSED cycleCounts plus err
        #@@ convert Cycle Counts to Timing (sec), for comparasion
        %reset -f
        import os, ctypes, time, math;import matplotlib.pyplot as plt
        N_TERM_Sary, avgEXEC_TIMEary, stddevEXEC_TIMEary = [], [], []
        _libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
        val = _libInC.version();print("Library version: "+ str(val))

        # Standard deviation of list, Using sum() + list comprehension
        def Stddev(test_list):
            mean = sum(test_list) / len(test_list)
            variance = sum([((x - mean) ** 2) for x in test_list]) / len(test_list)
            res = variance ** 0.5
            return res

        def Cycles2Seconds(cycles):
            JustOneCycle = 1/650000000 #seconds
            return (cycles * JustOneCycle)

        _libInC.init_cntrs(1,0)#reset=1, enable_divider=1
        ELAPSED1_DIFFary = []
        ELAPSED2_DIFFary = []
        ELAPSED__DIFFary = []
        for NumberOfTerms in range(0,35,5):
            N_TERM_Sary.append(NumberOfTerms)
            cmd = "taskset -c 1 python3 fib.py "+ str(NumberOfTerms);print (cmd)
            EXEC_TIMEary = []
            start1 = ctypes.c_uint(_libInC.gcyclec()).value
            start2 = time.time()
            os.system(cmd)
            stop1 = ctypes.c_uint(_libInC.gcyclec()).value
            stop2 = time.time()
            elapsed1 = stop1-start1
            elapsed1 = Cycles2Seconds(elapsed1)
            elapsed2 = stop2-start2
            diff = elapsed2-elapsed1
            ELAPSED1_DIFFary.append(elapsed1)
            ELAPSED2_DIFFary.append(elapsed2)
            ELAPSED__DIFFary.append(diff)
```

```
print("all     nTerms: " + str(N_TERM_Sary))
print("all   elapsed1: " + str(ELAPSED1_DIFFary))
print("all   elapsed2: " + str(ELAPSED2_DIFFary))
print("all Difference: " + str(ELAPSED__DIFFary))

import matplotlib.pyplot as plt
x = N_TERM_Sary
y = ELAPSED2_DIFFary
plt.title("Number of terms VS [Elapsed timing] vs [Cycle counts timing] comparasion")
plt.xlabel("number of terms")
plt.ylabel("Timing Difference [Cycle counts VS Timing])")

y_error = ELAPSED__DIFFary
plt.plot(x, y)
plt.errorbar(x, y,
             yerr = y_error,
             fmt ='o')
```

```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.8623809814453125e-05
taskset -c 1 python3 fib.py 5
time spent: 4.5299530029296875e-05
taskset -c 1 python3 fib.py 10
time spent: 0.000339508056640625
taskset -c 1 python3 fib.py 15
time spent: 0.003369569778442383
taskset -c 1 python3 fib.py 20
time spent: 0.035964250564575195
taskset -c 1 python3 fib.py 25
time spent: 0.39615368843078613
taskset -c 1 python3 fib.py 30
time spent: 4.441561698913574
all     nTerms: [0, 5, 10, 15, 20, 25, 30]
all   elapsed1: [0.04315480307692308, 0.2914146476923077, 0.06167966769230769, 0.0385
7442769230769, 0.03841540153846154, 0.08552718153846153, 0.16923483076923077]
all   elapsed2: [0.2822880744934082, 0.3119163513183594, 0.2949042320251465, 0.284116
9834136963, 0.317455530166626, 0.6771078109741211, 4.721894979476929]
all Difference: [0.23913327141648513, 0.020501703626051693, 0.23322456433283878, 0.24
55425557213886, 0.2790401286281644, 0.5915806294356596, 4.552660148707698]
```

Out[2]:

```
<ErrorbarContainer object of 3 artists>
```

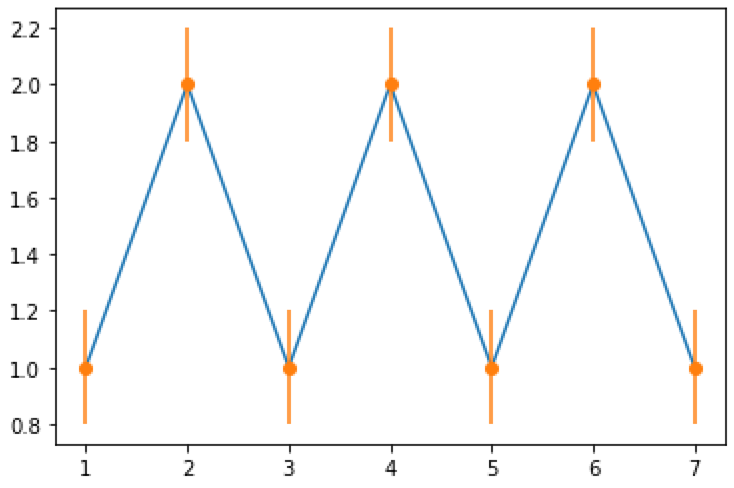


In [5]:
```
def Cycles2Seconds(cycles):
    JustOneCycle = 1/650000000 #seconds
    return (cycles * JustOneCycle)

cycles = 650000000
TimeElapsedInSeconds = Cycles2Seconds(cycles)
print(TimeElapsedInSeconds)
```

```
1.0
```

In [30]:
```
 # importing matplotlib
import matplotlib.pyplot as plt
# making a simple plot
x =[1, 2, 3, 4, 5, 6, 7]
y =[1, 2, 1, 2, 1, 2, 1]

# creating error
y_error = 0.2

# plotting graph
plt.plot(x, y)

plt.errorbar(x, y,
             yerr = y_error,
             fmt ='o')
```

Out[30]:
```
<ErrorbarContainer object of 3 artists>
```



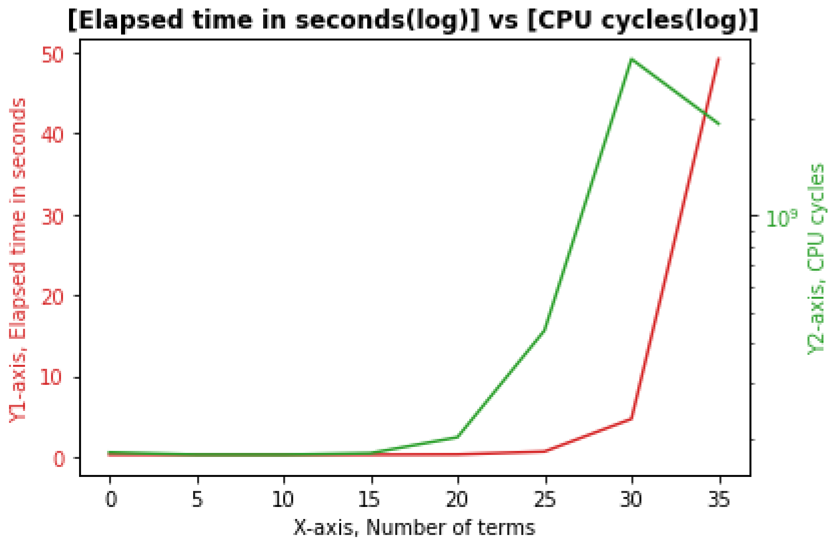
In [ ]:
```
#Cycles To Seconds Formula
f = 3e9#650000000
cycles = 100

secs=(1/( *f))*cycles

print(secs)
```

In [ ]:
```
f = 650000000
JustOneCycle = 1/f #secons
print(JustOneCycle)
print("----------")
cycles = 3000000000
TimeElapsedInSeconds = cycles * JustOneCycle
print(TimeElapsedInSeconds)
```

In [ ]:
```
# Part A3.2: Comparing and Gathering Data, ERROR BAR (running fibonacci sequence)
%reset -f
CPU, TERMSary, ELAPSEDary, ELAPSEDCYCLESary = "1", [], [], []

import os, ctypes, time, math;import matplotlib.pyplot as plt
f = 650000
JustOneCycle = 1/f #secons

def Average(lst):
    return sum(lst) / len(lst)

_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))
_libInC.init_cntrs(1,1)
for NumberOfTerms in range(0,35,5):
    cmd = "taskset -c "+CPU+" python3 fib.py "+ str(NumberOfTerms);print (cmd)
    start = time.time()
    #_libInC.init_cntrs(1,1)
    StartCycles = ctypes.c_uint(_libInC.gcyclec()).value
    os.system(cmd)
    lst = []
    #for n in range(4):
    #    cycles = ctypes.c_uint(_libInC.gcyclec()).value
    #    lst.append(cycles)
    #ave = Average(lst)
    StopCycles = ctypes.c_uint(_libInC.gcyclec()).value
    ElapsedTimeCycles = StopCycles - StartCycles
    TimeElapsedInSeconds = ElapsedTimeCycles * JustOneCycle
    #TimeElapsedInSeconds = round(TimeElapsedInSeconds, 9)
    end = time.time();elapsed = end - start;elapsed = round(elapsed, 9)
    print(cmd+"-> NumberOfTerms: "+ str(NumberOfTerms)+ ":\nElapsed time = " +str(elap
          +", Elapsed Seconds by get cycles = "+str(TimeElapsedInSeconds))
    TERMSary.append(NumberOfTerms);ELAPSEDary.append(elapsed);ELAPSEDCYCLESary.append(
    lst = []

x = TERMSary
dataset_1 = ELAPSEDary
dataset_2 = ELAPSEDCYCLESary
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('X-axis, Number of terms')
ax1.set_ylabel('Y1-axis, Elapsed time in seconds', color = color)
ax1.plot(x, dataset_1, color = color)
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx() # Adding Twin Axes to plot using dataset_2
color = 'tab:green'
ax2.set_ylabel('Y2-axis, Elapsed time by get CPU cycles(Sec)', color = color)
ax2.plot(x, dataset_2, color = color)
ax2.tick_params(axis ='y', labelcolor = color)
plt.title('[Elapsed time (Sec)] vs [Elapsed time by get CPU cycles(Sec)]', fontweight
#plt.yscale("log")
plt.show()
```

In [3]:
```
# Part A3.2: Comparing and Gathering Data, (running fibonacci sequence)
%reset -f
CPU, TERMSary, ELAPSEDary, CYCLESary = "1", [], [], []

import os, ctypes, time, math;import matplotlib.pyplot as plt

def Average(lst):
    return sum(lst) / len(lst)

_libInC = ctypes.CDLL('./clock_example0/libMyLib.so')
val = _libInC.version();print("Library version: "+ str(val))
for NumberOfTerms in range(0,40,5):
    cmd = "taskset -c "+CPU+" python3 fib.py "+ str(NumberOfTerms);print (cmd)
    start = time.time()
    _libInC.init_cntrs(1,0)
    os.system(cmd)
    lst = []
    for n in range(4):
        cycles = ctypes.c_uint(_libInC.gcyclec()).value
        lst.append(cycles)
    ave = Average(lst)
    end = time.time();elapsed = end - start;elapsed = round(elapsed, 7)
    print(cmd+"-> NumberOfTerms: "+ str(NumberOfTerms)+ " -> Elapsed time = " +str(ela
          +", cycles = "+str(ave))
    TERMSary.append(NumberOfTerms);ELAPSEDary.append(elapsed);CYCLESary.append(ave)
    lst = []

x = TERMSary
dataset_1 = ELAPSEDary
dataset_2 = CYCLESary
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('X-axis, Number of terms')
ax1.set_ylabel('Y1-axis, Elapsed time in seconds', color = color)
ax1.plot(x, dataset_1, color = color)
ax1.tick_params(axis ='y', labelcolor = color)
ax2 = ax1.twinx() # Adding Twin Axes to plot using dataset_2
color = 'tab:green'
ax2.set_ylabel('Y2-axis, CPU cycles', color = color)
ax2.plot(x, dataset_2, color = color)
ax2.tick_params(axis ='y', labelcolor = color)
plt.title('[Elapsed time in seconds(log)] vs [CPU cycles(log)]', fontweight ="bold")
plt.yscale("log")
plt.show()
```

```
Library version: 700
taskset -c 1 python3 fib.py 0
Please enter a positive integer
time spent: 3.8623809814453125e-05
taskset -c 1 python3 fib.py 0-> NumberOfTerms: 0 -> Elapsed time = 0.2854857, cycles
= 182654923.25
taskset -c 1 python3 fib.py 5
time spent: 4.38690185546875e-05
taskset -c 1 python3 fib.py 5-> NumberOfTerms: 5 -> Elapsed time = 0.2801116, cycles
= 179635468.25
taskset -c 1 python3 fib.py 10
time spent: 0.00034427642822265625
taskset -c 1 python3 fib.py 10-> NumberOfTerms: 10 -> Elapsed time = 0.2799449, cycle
s = 179563247.25
taskset -c 1 python3 fib.py 15
time spent: 0.0034863948822021484
taskset -c 1 python3 fib.py 15-> NumberOfTerms: 15 -> Elapsed time = 0.283051, cycles
= 181602819.0
taskset -c 1 python3 fib.py 20
time spent: 0.03589892387390137
taskset -c 1 python3 fib.py 20-> NumberOfTerms: 20 -> Elapsed time = 0.3169413, cycle
s = 203540381.0
taskset -c 1 python3 fib.py 25
time spent: 0.39631080627441406
taskset -c 1 python3 fib.py 25-> NumberOfTerms: 25 -> Elapsed time = 0.6766727, cycle
s = 437420747.0
taskset -c 1 python3 fib.py 30
time spent: 4.435394048690796
taskset -c 1 python3 fib.py 30-> NumberOfTerms: 30 -> Elapsed time = 4.7155745, cycle
s = 3062687988.5
taskset -c 1 python3 fib.py 35
time spent: 48.941927671432495
taskset -c 1 python3 fib.py 35-> NumberOfTerms: 35 -> Elapsed time = 49.2225311, cycl
es = 1927318739.5
```


[Elapsed time in seconds(log)] vs [CPU cycles(log)]
Y1-axis, Elapsed time in seconds
Y2-axis, CPU cycles
X-axis, Number of terms
10^9