Assignment 3

In [1]:
```
import subprocess
import matplotlib.pyplot as plot
from pprint import pprint
from operator import itemgetter

times = []
cyc_cnts = []

NUM_TRIALS = 5
CPU_FREQ_HZ = 650000000 # CPU frequency is 650 MHz

# Run the Fibonacci program and get the time and cycle count stats
for nterms in range(6, 28, 3):
    trial_times = []
    trial_cyc_cnts = []
    print(f"\nRunning fib.py for number of terms: {nterms}")

    for trial in range(0, NUM_TRIALS):
        res = subprocess.run(
                  ["taskset", "-c", "1", "python3", "fib.py", str(nterms)],
                  capture_output = True, text = True)

        trial_times.append(float(res.stdout.split(",")[0].strip()))
        trial_cyc_cnts.append(int(res.stdout.split(",")[1].strip()))

    print("\nTimes:\n")
    pprint(trial_times)
    print("\nCycle counts:\n")
    pprint(trial_cyc_cnts)

    time_avg = sum(trial_times) / NUM_TRIALS

    err_squared_sum = 0
    # Calculate the standard deviation
    for t in trial_times:
        err_squared_sum += (t - time_avg) ** 2
    time_stddev = (err_squared_sum / NUM_TRIALS) ** 0.5

    cyc_cnt_avg = sum(trial_cyc_cnts) / NUM_TRIALS

    err_squared_sum = 0
    # Calculate the standard deviation
    for cc in trial_cyc_cnts:
        err_squared_sum += (cc - cyc_cnt_avg) ** 2
    cyc_cnt_stddev = (err_squared_sum / NUM_TRIALS) ** 0.5

    times.append((time_avg, time_stddev))
```

```
    cyc_cnts.append((cyc_cnt_avg / CPU_FREQ_HZ, cyc_cnt_stddev / CPU_FREQ_HZ

print("\nOverall data")
pprint(times)
print("\n")
pprint(cyc_cnts)
```

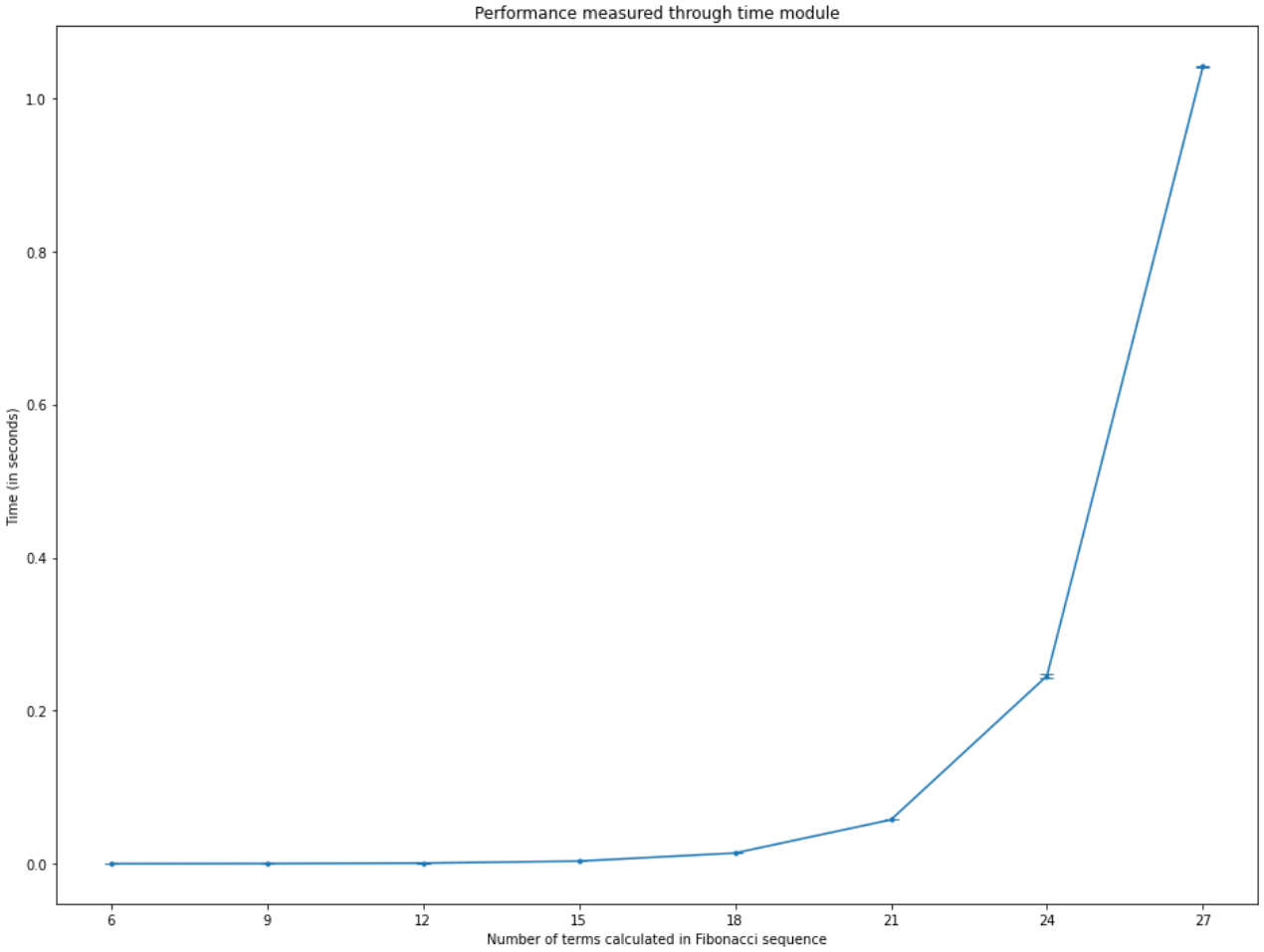
```
Running fib.py for number of terms: 6

Times:

[0.00013256072998046875,
 0.0001461505889892578,
 0.00013375282287597656,
 0.00013566017150878906,
 0.00013208389282226562]

Cycle counts:

[46012, 47591, 46474, 47813, 45621]

Running fib.py for number of terms: 9

Times:

[0.00028395652770996094,
 0.00027847290039062,
 0.00028109550476074219,
 0.00027871131896972656,
 0.00027990341186523438]

Cycle counts:

[144431, 142426, 141977, 142161, 142490]

Running fib.py for number of terms: 12

Times:

[0.00092506408691406250,
 0.00095343589782714844,
 0.00090646743774414062,
 0.00090622901916503906,
 0.00092506408691406250]

Cycle counts:

[560808, 574150, 548778, 550037, 561809]

Running fib.py for number of terms: 15

Times:
```

```
[0.0035181045532226562,
 0.0035085678100585938,
 0.0035195350646972656,
 0.0035791397094726562,
 0.0035033226013183594]

Cycle counts:

[2249511, 2242374, 2250878, 2285050, 2240091]

Running fib.py for number of terms: 18

Times:

[0.013858556747436523,
 0.013887882232666016,
 0.013792037963867188,
 0.015005350112915039,
 0.01383662223815918]

Cycle counts:

[8969783, 8990643, 8928239, 9719915, 8955314]

Running fib.py for number of terms: 21

Times:

[0.05755305290222168,
 0.057578086853027344,
 0.057793617248535156,
 0.05755758285522461,
 0.057566165924072266]

Cycle counts:

[37373555, 37389956, 37519077, 37376241, 37381598]

Running fib.py for number of terms: 24

Times:

[0.2443220615386963,
 0.24437332153320312,
 0.24412918090820312,
 0.24428510665893555,
 0.25002527236938477]

Cycle counts:

[158774559, 158807067, 158646748, 158750308, 162470084]

Running fib.py for number of terms: 27
```

```
Times:

[1.0417380332946777,
 1.0420022010803223,
 1.0409727096557617,
 1.0410993099212646,
 1.0410728454589844]

Cycle counts:

[677091143, 677266002, 676597495, 676682409, 676664662]

Overall data
[(0.00013604164123535157, 5.20299046686338e-06),
 (0.0002804279327392578, 1.9970313636180804e-06),
 (0.0009232521057128906, 1.7257684737996346e-05),
 (0.0035257339477539062, 2.7370619025706397e-05),
 (0.01407608985900879, 0.00046568178089822425),
 (0.05760970115661621, 9.235285766742223e-05),
 (0.24542698860168458, 0.0023005864858534316),
 (1.041377019882202, 0.0004133506685172427)]


[(7.184953846153846e-05, 1.327236181565589e-06),
 (0.00021953384615384615, 1.363730005032968e-06),
 (0.0008601790769230769, 1.4191234117410224e-05),
 (0.0034670473846153843, 2.5013568093757938e-05),
 (0.014019659692307694, 0.00046807141458501960),
 (0.05755090061538461, 8.581197861018054e-05),
 (0.24536885107692305, 0.002294049089701676),
 (1.0413236033846154, 0.00041101317878570104)]
```

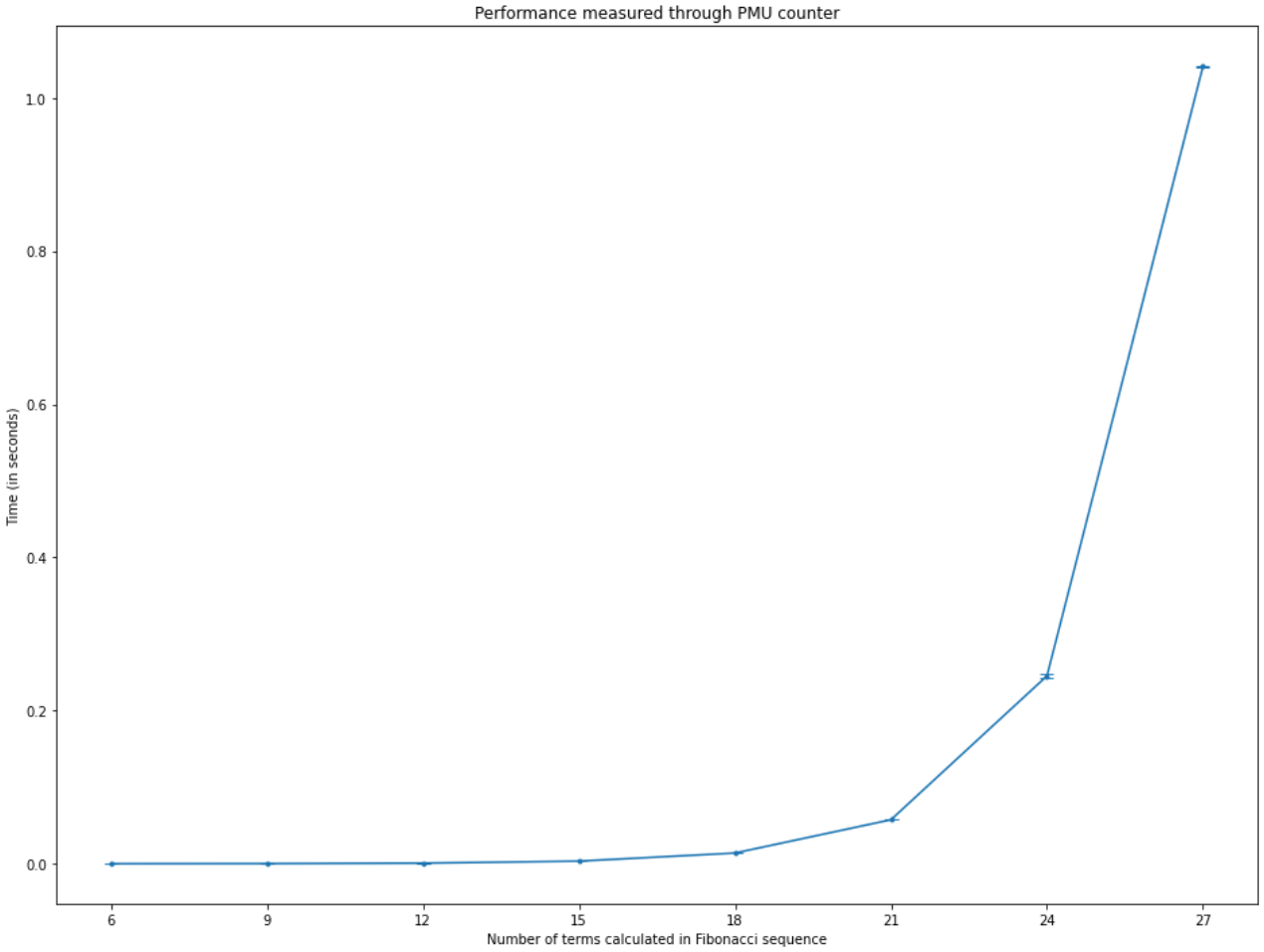
In [8]:
```
plot.figure(figsize=(16, 12))
plot.errorbar(
    range(6, 28, 3), # X data
    list(map(itemgetter(0), times)), # Y data
    list(map(itemgetter(1), times)), # Standard deviations
    marker = '.', capsize = 5.0)

plot.xticks(range(6, 28, 3))
plot.xlabel("Number of terms calculated in Fibonacci sequence")
plot.ylabel("Time (in seconds)")

plot.title("Performance measured through time module")
```

Out[8]:
```
Text(0.5, 1.0, 'Performance measured through time module')
```



In [9]:
```
plot.figure(figsize=(16, 12))
plot.errorbar(
    range(6, 28, 3), # X data
    list(map(itemgetter(0), cyc_cnts)), # Y data
    list(map(itemgetter(1), cyc_cnts)), # Standard deviations
    marker = '.', capsize = 5.0)

plot.xticks(range(6, 28, 3))
plot.xlabel("Number of terms calculated in Fibonacci sequence")
plot.ylabel("Time (in seconds)")

plot.title("Performance measured through PMU counter")
```

Out[9]:
```
Text(0.5, 1.0, 'Performance measured through PMU counter')
```


Performance measured through PMU counter
Time (in seconds)
0.0
0.2
0.4
0.6
0.8
1.0
6
9
12
15
18
21
24
27
Number of terms calculated in Fibonacci sequence

In [ ]: