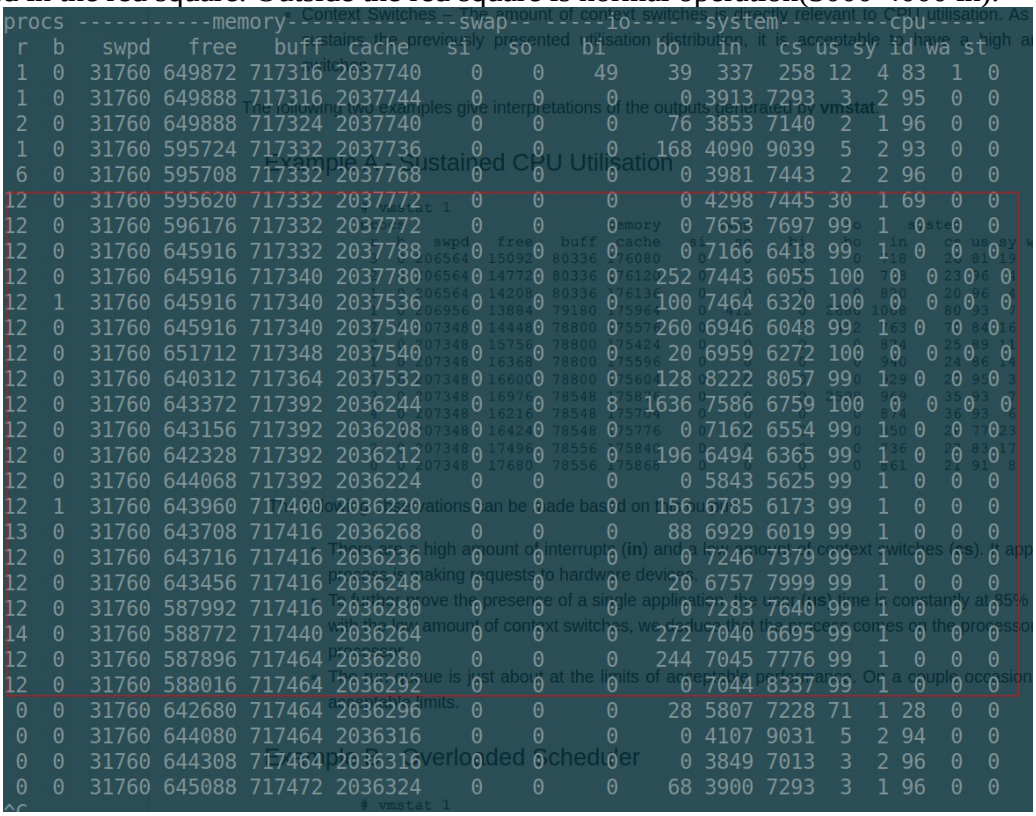
EP-Lab01-2021

Olaru Gabriel Iulian, 342C2

EX1

The impact of creating workers with stress will result in a greater number of interrupts(6000-7000), as depicted in the red square. Outside the red square is normal operation(3000-4000 in).

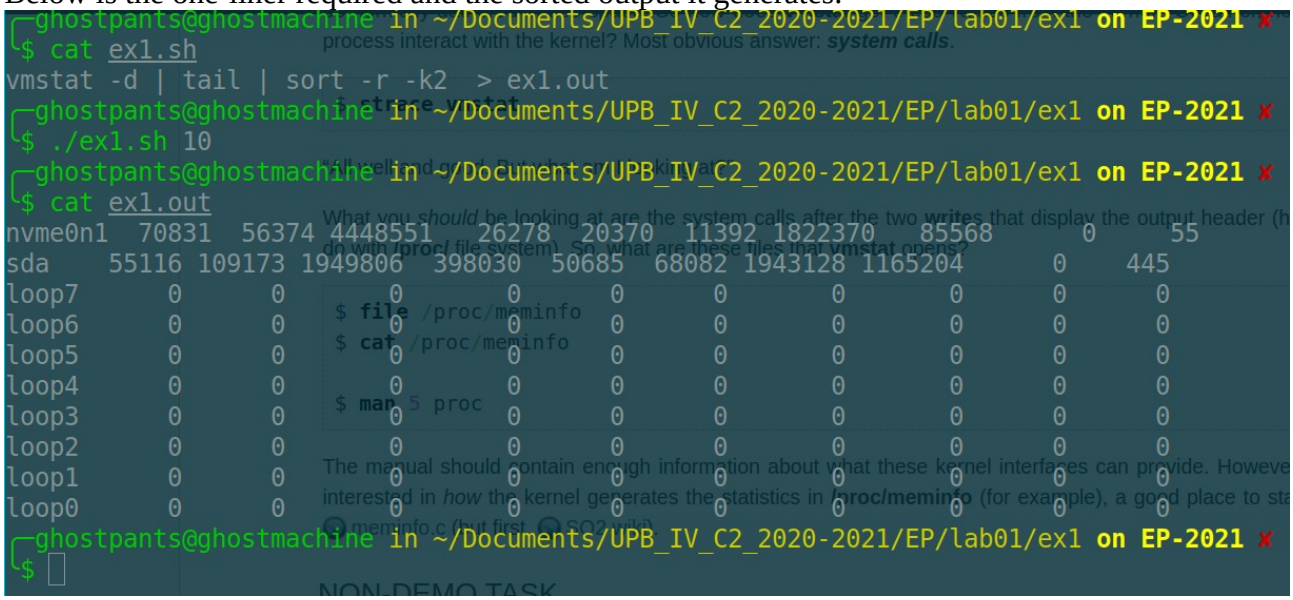


```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0  31760 649872 717316 2037740    0    0    49    39  337  258 12  4 83  1  0
 1  0  31760 649888 717316 2037744    0    0     0     0 3913 7293  3  2 95  0  0
 2  0  31760 649888 717324 2037740    0    0     0    76 3853 7140  2  1 96  0  0
 1  0  31760 595724 717332 2037736    0    0     0   168 4090 9039  5  2 93  0  0
 6  0  31760 595708 717332 2037768    0    0     0     0 3981 7443  2  2 96  0  0
12  0  31760 595620 717332 2037772    0    0     0     0 4298 7445 30  1 69  0  0
12  0  31760 596176 717332 2037772    0    0     0     0 7653 7697 99  1  0  0  0
12  0  31760 645916 717332 2037788    0    0     0     0 7166 6418 99  1  0  0  0
12  0  31760 645916 717340 2037780    0    0     0   252 7443 6055 100  0  0  0  0
12  1  31760 645916 717340 2037536    0    0     0   100 7464 6320 100  0  0  0  0
12  0  31760 645916 717340 2037540    0    0     0   260 6946 6048 99  1  0  0  0
12  0  31760 651712 717348 2037540    0    0     0    20 6959 6272 100  0  0  0  0
12  0  31760 640312 717364 2037532    0    0     0   128 8222 8057 99  1  0  0  0
12  0  31760 643372 717392 2036244    0    0     8  1636 7586 6759 100  0  0  0  0
12  0  31760 643156 717392 2036208    0    0     0     0 7162 6554 99  1  0  0  0
12  0  31760 642328 717392 2036212    0    0     0   196 6494 6365 99  1  0  0  0
12  0  31760 644068 717392 2036224    0    0     0     0 5843 5625 99  1  0  0  0
12  1  31760 643960 717400 2036220    0    0     0   156 6785 6173 99  1  0  0  0
13  0  31760 643708 717416 2036268    0    0     0    88 6929 6019 99  1  0  0  0
12  0  31760 643716 717416 2036236    0    0     0     0 7246 7379 99  1  0  0  0
12  0  31760 643456 717416 2036248    0    0     0    20 6757 7999 99  1  0  0  0
12  0  31760 587992 717416 2036280    0    0     0     0 7283 7640 99  1  0  0  0
14  0  31760 588772 717440 2036264    0    0     0   272 7040 6695 99  1  0  0  0
12  0  31760 587896 717464 2036280    0    0     0   244 7045 7776 99  1  0  0  0
12  0  31760 588016 717464 2036292    0    0     0     0 7044 8337 99  1  0  0  0
 0  0  31760 642680 717464 2036296    0    0     0    28 5807 7228 71  1 28  0  0
 0  0  31760 644080 717464 2036316    0    0     0     0 4107 9031  5  2 94  0  0
 0  0  31760 644308 717464 2036316    0    0     0     0 3849 7013  3  2 96  0  0
 0  0  31760 645088 717472 2036324    0    0     0    68 3900 7293  3  1 96  0  0
^C
```

Stress command used, passing the number of cores indirectly:

sudo stress --cpu $(nproc) --timeout $1

Below is the one-liner required and the sorted output it generates.



```
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex1 on EP-2021 ✘
$ cat ex1.sh
vmstat -d | tail | sort -r -k2  > ex1.out
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex1 on EP-2021 ✘
$ ./ex1.sh 10
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex1 on EP-2021 ✘
$ cat ex1.out
nvme0n1  70831  56374 4448551   26278  20370   11392 1822370   85568       0      55
sda    55116 109173 1949806  398030  50685   68082 1943128 1165204       0     445
loop7      0      0       0       0      0       0       0       0       0       0
loop6      0      0       0       0      0       0       0       0       0       0
loop5      0      0       0       0      0       0       0       0       0       0
loop4      0      0       0       0      0       0       0       0       0       0
loop3      0      0       0       0      0       0       0       0       0       0
loop2      0      0       0       0      0       0       0       0       0       0
loop1      0      0       0       0      0       0       0       0       0       0
loop0      0      0       0       0      0       0       0       0       0       0
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex1 on EP-2021 ✘
$
```

EX2.

The python script crashes because python enforces a max size of the recursion stack: no more than 1000 recursive calls. We can fix this by artificially increasing the limit when we get close to it, as shown in the screenshot below.

```python
#!/usr/bin/python3

import sys

def factorial(n, level):
    recursion_limit = sys.getrecursionlimit()
    if level ==  recursion_limit - 100:
            sys.setrecursionlimit(recursion_limit + 100)

    if n == 0:
        return 1
    else:
        return n * factorial(n - 1, level + 1)

if __name__ == '__main__':
    print()

    big_number = int(sys.argv[1])

    for i in range(1, big_number + 1):
        print('factorial(%d) = %d' % (i, factorial(i, 0)))

```

Creating workers then setting their affinity bonds them to speciffic CPUs. This can be seen when monitoring the activity.

sudo stress --cpu $((`nproc` - 1)) --timeout 2

sudo taskset -cp 1-<number of cores> <pid of workers from prev command>

```
02:49:51 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
02:49:52 PM  all   90,67    8,33    1,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    0   97,00    0,00    3,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    1   92,00    7,00    1,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    2   72,00   20,00    8,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    3   78,00   22,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    4  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    5  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    6  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    7   93,00    7,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    8   74,75   25,25    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM    9   81,00   19,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM   10  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:52 PM   11  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00

02:49:52 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
02:49:53 PM  all   92,24    7,17    0,58    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    0   93,07    3,96    2,97    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    1   88,12    7,92    3,96    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    2   86,00   14,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    3   70,00   30,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    4  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    5  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    6   94,00    6,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    7   98,02    1,98    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    8   80,20   19,80    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM    9   97,00    3,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM   10  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
02:49:53 PM   11  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
```

Below is the script generating N/2 workers and binding them to odd cores. First the list of cores is created and then the workers are started and their PID is extracted into a list. Finally, the pid is binded to the cores from the first list.

```
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex2 on EP-2021 ✗
$ cat ex2.sh
#!/bin/bash

#Generate list of cores
target_cores=($(for (( i = 1; i < $((`nproc` - 1)); i=i+2 )); do echo $i ; done | paste -s -d ' '))

# Start workers
sudo stress -v --cpu $(((`nproc` - 1) / 2)) --timeout 180 > ./stress.txt 2>&1 &

# Get workers PIDs
worker_pids=($(cat stress.txt | grep forked | awk '{print $8}' | sed 's/.$//' | sed 's/^.//g' | paste -s -d ' '))

echo "${target_cores[*]}"
echo
echo "${worker_pids[*]}"
echo

counter=0
for i in "${target_cores[@]}"
do
        sudo taskset -cp $i ${worker_pids[$counter]} 2>&1
        counter=$((counter+1))
        echo
done
```

```
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex2 on EP-2021 ✗
$ ./ex2.sh
1 3 5 7 9

5467 5468 5469 5470 5471

pid 5467's current affinity list: 0-11
pid 5467's new affinity list: 1

pid 5468's current affinity list: 0-11
pid 5468's new affinity list: 3

pid 5469's current affinity list: 0-11
pid 5469's new affinity list: 5

pid 5470's current affinity list: 0-11
pid 5470's new affinity list: 7

pid 5471's current affinity list: 0-11
pid 5471's new affinity list: 9
```

EX3

Below is the script that measures the compression rates and time and outputs the data as a list. The 0% compression rate is due to an empty file being present in the directory (by mistake). It also gets data about the size of the compressed files (used for the graphs).

```
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex3 on EP-2021 x
$ ./ex3.sh
0% 56% 58% 60% 61% 63% 63% 63% 63% 63% 0% 39% 41% 41% 41% 41% 41% 41% 41% 0% 0% 0% 0% 0% 0% 0% 0% 0% 77% 79% 80% 82% 83% 84% 84% 84% 84%

0m0,022s 0m0,110s 0m0,111s 0m0,156s 0m0,158s 0m0,232s 0m0,357s 0m0,428s 0m0,519s 0m0,506s 0m0,012s 0m0,090s 0m0,094s 0m0,101s 0m0,115s 0m0,144s 0m0,152s 0m0,153s 0m0,170s 0m0,164s 0m0,001s 0m0,001s 0m0,001s 0m0,001s 0m0,001s 0m0,001s 0m0,001s 0m0,001s 0m0,001s 0m0,005s 0m0,017s 0m0,020s 0m0,040s 0m0,027s 0m0,051s 0m0,069s 0m0,094s 0m0,199s 0m0,203s

6488665 2848209 2724469 2609121 2522249 2428104 2385236 2376868 2372126 2372105 4264316 2582921 2535089 2505860 2529535 2516390 2512654 2512067 2511624 2511497 0 0 0 0 0 0 0 0 0 0 1440054 331089 310084 285593 255405 240715 231578 229028 227360 227226
```

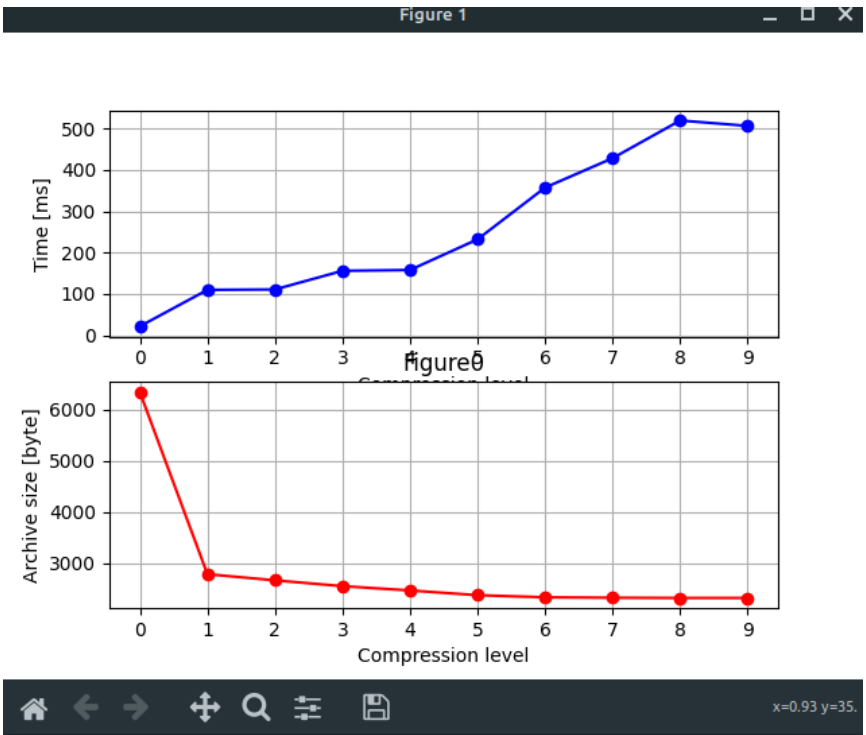
```
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex3 on EP-2021 x
$ cat ex3.sh
#!/bin/bash

script_name="ex3.sh"
directory_name="part2"
zip_times=()
zip_ratios=()
zip_sizes=()

for file in *
do
        if [[ ""$file"" != "$script_name" ]]; then
                for i in {0..9}
                do
                        #echo "zipping $file with -$i..."
                        zip_times+=($({ time zip -$i $file.zip $file >/dev/null; } |& grep real | awk '{print $2}'))
                        zip_ratios+=($(unzip -v $file.zip | tail -1 | awk '{print $3}'))
                        zip_sizes+=($(unzip -v $file.zip | tail -1 | awk '{print $2}'))
                        rm $file.zip
                        #echo
                done
        fi
done

echo "${zip_ratios[*]}"
echo
echo "${zip_times[*]}"
echo
echo "${zip_sizes[*]}"
```

The graphs below are recreated from the data gathered above, and they depict how the size of the file decreases with the increase of the level of compression. Please note that the time of compression also increases



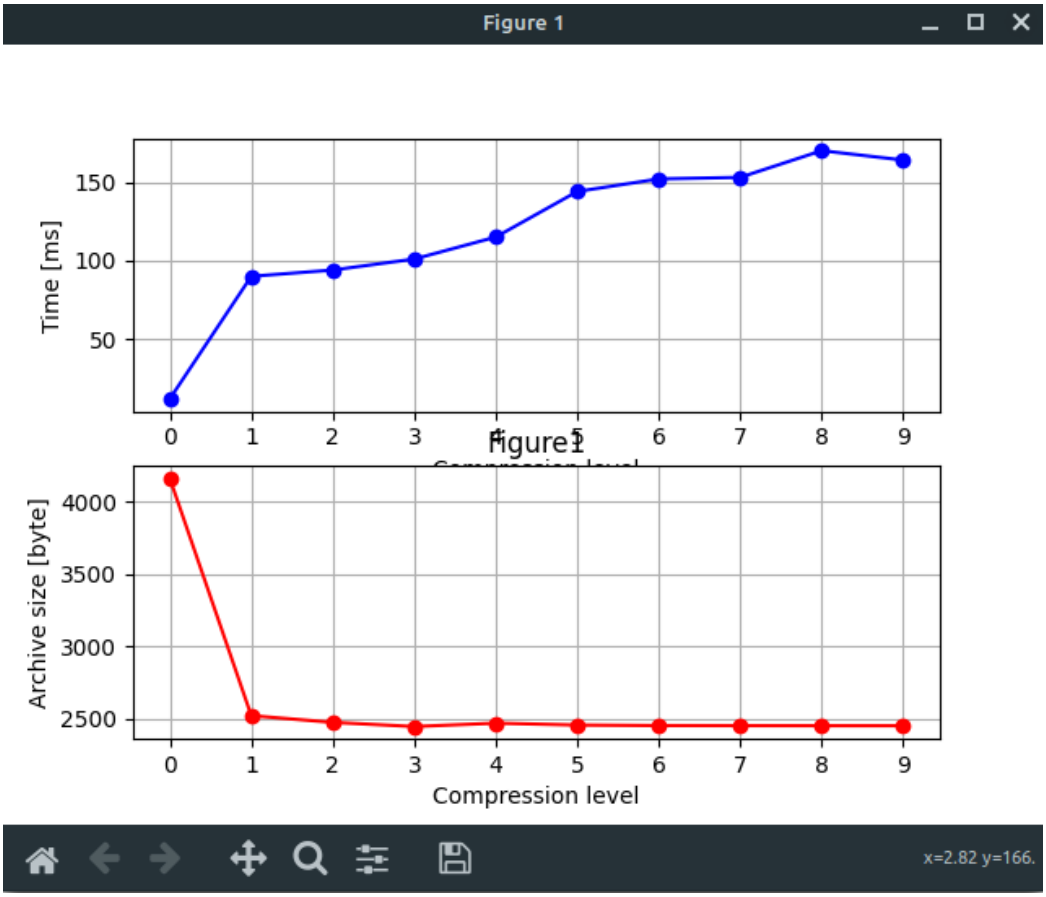


Figure1

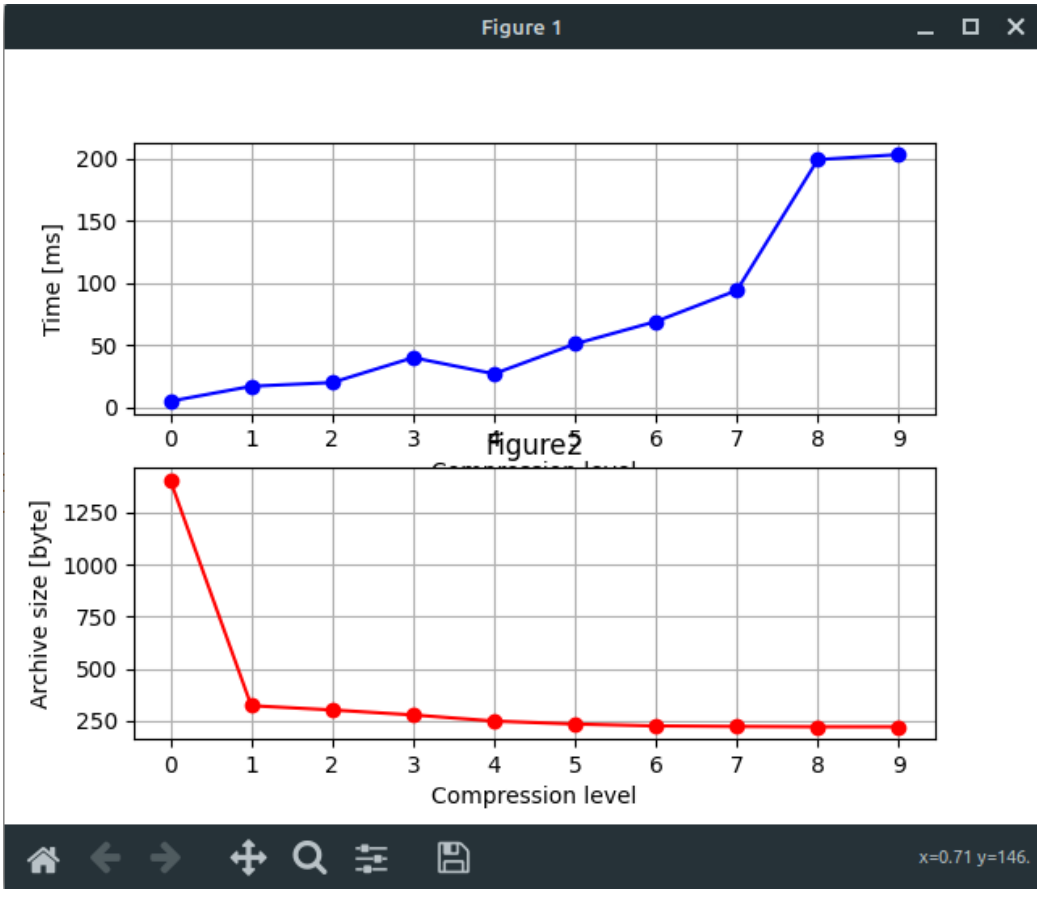


Figure2

EX4.

The gist of it would be to extract the values from the registers and the pass themto the macro. Didn’t manage to get it done in time.

```
/* TODO: count L2 cache misses for the next block using RDPMC */
register int eax2 asm("eax");
register int edx2 asm("ebx");
register int ecx2 asm("ecx");

rdpmc(ecx2, eax2, edx2);
```

```
ghostpants@ghostmachine in ~/Documents/UPB_IV_C2_2020-2021/EP/lab01/ex4/hw_counter on EP-2021 ✘
$ taskset 0x01 ./mat_mul 1024
Multiplication 1 finished in  11.27 s
Multiplication 2 finished in   5.39 s
```

EX5.