1. Names of the students: Priyanshu Jain, Sayali Patil, Shadi Davari

2. **Output:**

root@raspberrypi_Sayali_Patil:/home/pi# cat /proc/sys/kernel/sched_autogroup_enabled
1
root@raspberrypi_Sayali_Patil:/home/pi# echo 0 > /proc/sys/kernel/sched_autogroup_enabled
root@raspberrypi_Sayali_Patil:/home/pi# cat /proc/sys/kernel/sched_autogroup_enabled
0

3. Refer to the attached screenshot.

4.
The empty (while) loop constantly keeps checking for true condition. As soon as any application gets the assigned processor's attention, it checks the loop condition to be true, just like any other condition and checks whether the next instruction it true or not. And this goes on continuously. Therefore, when we run the infinite loop on all the processors, the loop process taken up almost all the CPU time or at least 4/5th of the CPU time keeping the CPU busy. In this case, when we try to open a browser or a text editor, it doesn't open immediately or takes time to browse something in case of a browser or takes time to save the written text in case of a text editor as the cpu time window available for these tasks is very little.

When we terminate the infinite loop on all the cores, we can see a drastic change in the speed of browser process loading or in case of a text editor.

5. Output:

Without Background:
pi@raspberrypi_Sayali_Patil:~/userspace_programs/test_programs $ time ./dense_mm 300
Generating matrices...
Multiplying matrices...
Multiplication done!

real 0m2.407s
user 0m2.385s
sys 0m0.020s

With Background: (infinite loop running on all the 4 cores)
pi@raspberrypi_Sayali_Patil:~/userspace_programs/test_programs $ time ./dense_mm 300
Generating matrices...
Multiplying matrices...
Multiplication done!

real 0m5.435s
user 0m2.193s
sys 0m0.001s

6.
Since, we know that real time is the actual wall clock time, i.e., time from start to finish of the call. Real time tells us about the total elapsed time including time slices used by other process and timethe process spends blocked. Therefore, as we go on running background processes on increasing number of cpu cores, the dedicated cpu time available for any process cuts down into slices increasing the total real time values. And that's what we are being able to see in this case as well.

On the other hnad, user time doesn't increase drastically as it only denotes the time spent in user-mode within the process. Thus, user time remains fairly the same.

7. **Output:**
pi@raspberrypi_Sayali_Patil:~/userspace_programs/test_programs $ time ./dense_mm 300
Generating matrices...
Multiplying matrices...
Multiplication done!

real 0m7.677s
user 0m2.211s
sys 0m0.000s

The output supports our prediction as you can see in this output as well the real time has increased by almost thrice the actual running time, but the user time is still fairly the same.

8. **Proportion of time (user divided by real) recieved by the task:** 0.3900

9. **Proportion of time (user divided by real) recieved by the task:**
i. For nice priority -10 = 0.4148
ii. For nice priority -5 = 0.4049
iii. For nice priority 0 = 0.4132
iv. For nice priority 5 = 0.3764
v. For nice priority 10 = 0.3867
vi. For nice priority 19 = 0.3610

Enrichment Exercise: Refer to the attached screenshot