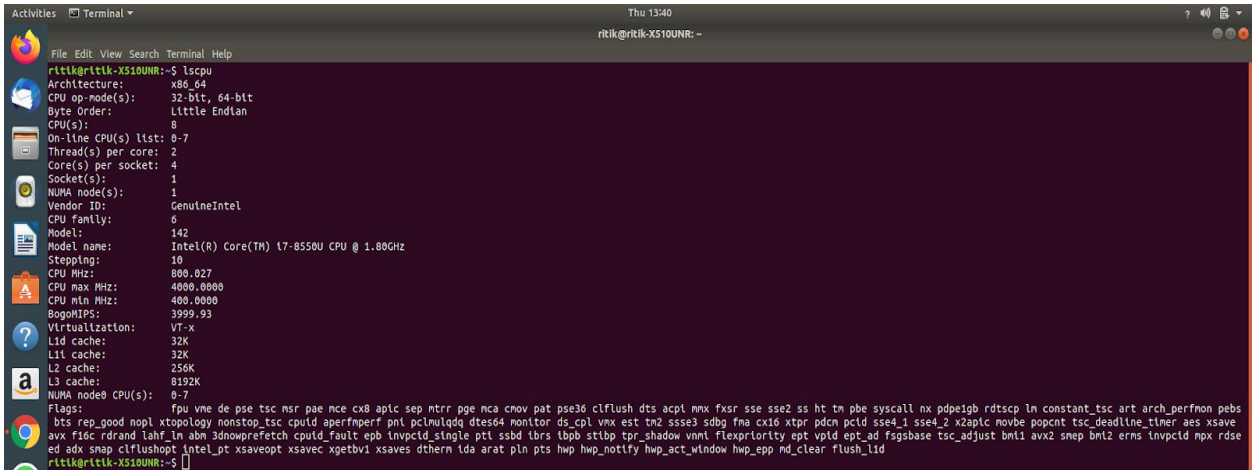
DEPARTMENT OF INFORMATION TECHNOLOGY, NITK SURATHKAL

Parallel Programming

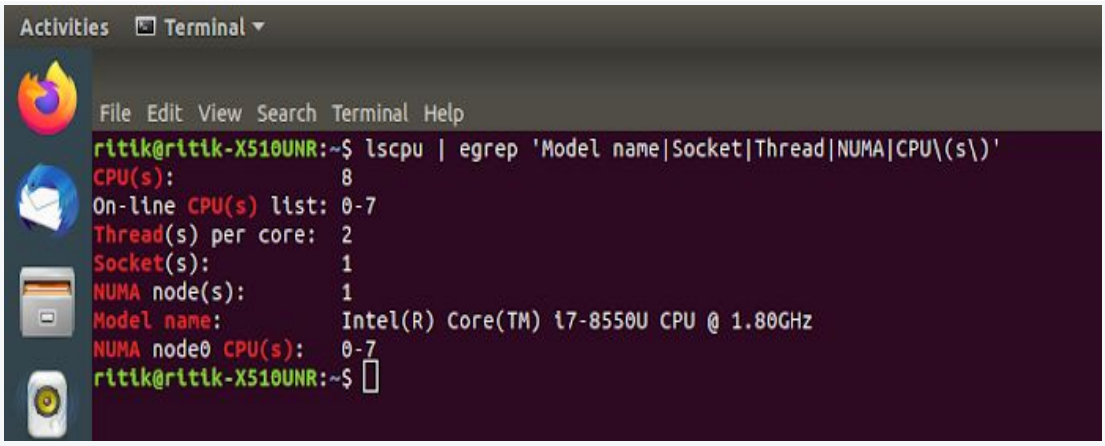
LAB 1 -3rd August 2020

Finding number of CPU s in system

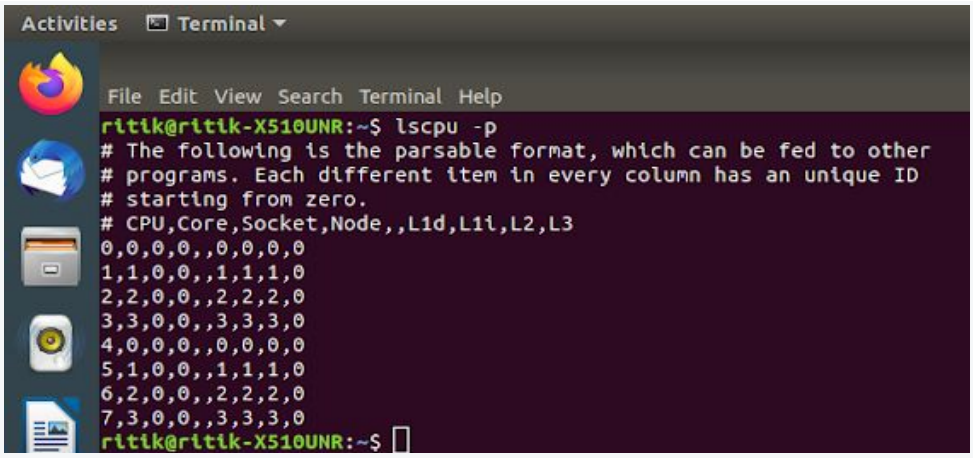
lscpu command



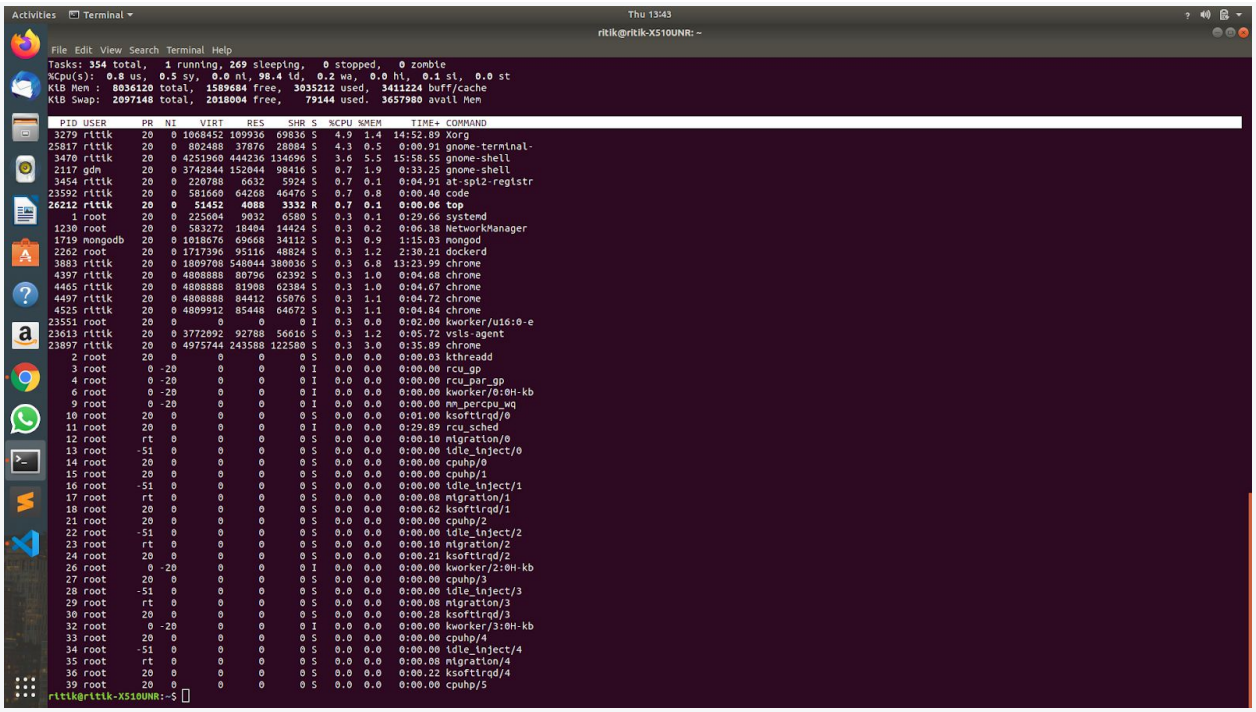

```
lscpu | egrep 'Model name|Socket|Thread|NUMA|CPU\(s\)'
```

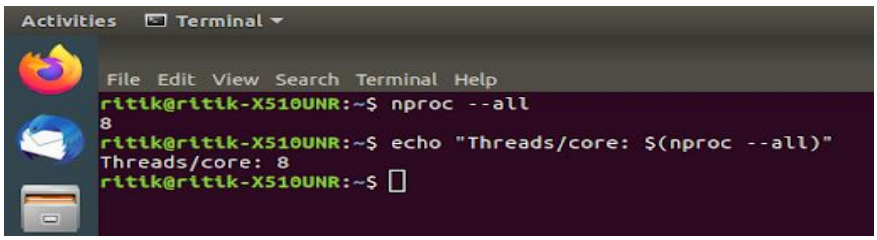


`lscpu -p`


Activities
Terminal
File Edit View Search Terminal Help
ritik@ritik-X510UNR:~$ lscpu -p
The following is the parsable format, which can be fed to other
programs. Each different item in every column has an unique ID
starting from zero.
CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0,0,0,0,,0,0,0,0
1,1,0,0,,1,1,1,0
2,2,0,0,,2,2,2,0
3,3,0,0,,3,3,3,0
4,0,0,0,,0,0,0,0
5,1,0,0,,1,1,1,0
6,2,0,0,,2,2,2,0
7,3,0,0,,3,3,3,0
ritik@ritik-X510UNR:~$

`Top`

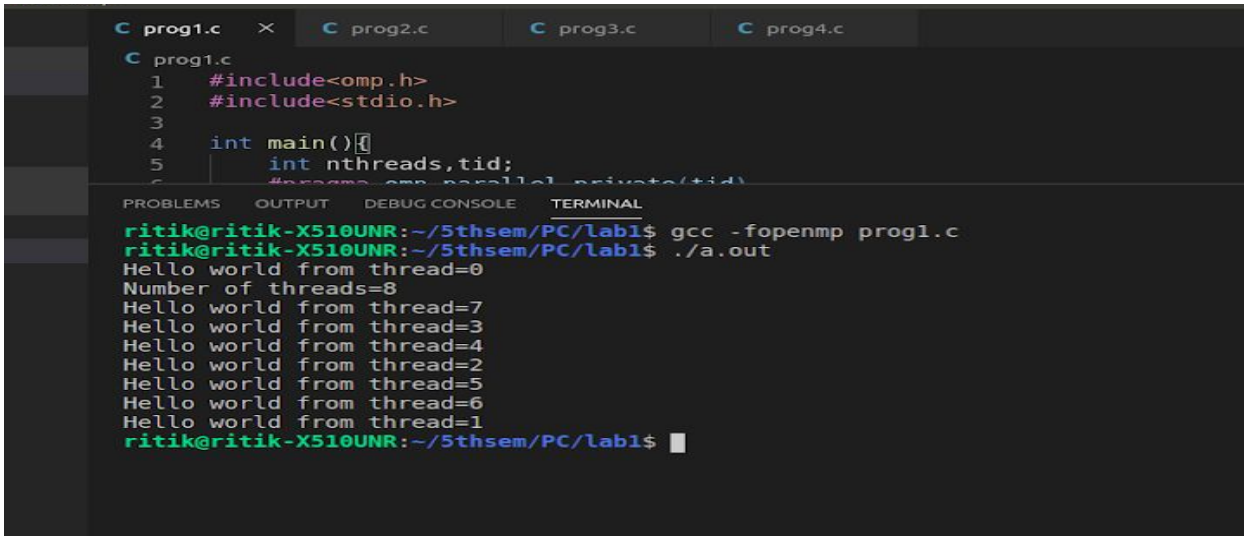

Activities
Terminal
Thu 13:43
ritik@ritik-X510UNR: ~
File Edit View Search Terminal Help
Tasks: 354 total, 1 running, 269 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.8 us, 0.5 sy, 0.0 ni, 98.4 id, 0.2 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem : 8036120 total, 1589684 free, 3035212 used, 3411224 buff/cache
KiB Swap: 2097148 total, 2018004 free, 79144 used. 3657980 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3279 ritik 20 0 1068452 109936 69836 S 4.9 1.4 14:52.89 Xorg
25817 ritik 20 0 802488 37876 28084 S 4.3 0.5 0:00.91 gnome-terminal-
3470 ritik 20 0 4251960 444236 134696 S 3.6 5.5 15:58.55 gnome-shell
2117 gdm 20 0 3742844 152844 98416 S 0.7 1.9 0:33.25 gnome-shell
3454 ritik 20 0 220788 6632 5924 S 0.7 0.1 0:04.91 at-spi2-registr
23592 ritik 20 0 581660 64268 46476 S 0.7 0.8 0:00.40 code
26212 ritik 20 0 51452 4088 3332 R 0.7 0.1 0:00.06 top
1 root 20 0 225604 9032 6580 S 0.3 0.1 0:29.66 systemd
1230 root 20 0 583272 18404 14424 S 0.3 0.2 0:06.38 NetworkManager
1719 mongodb 20 0 1018676 69668 34112 S 0.3 0.9 1:15.03 mongod
2262 root 20 0 1717396 95116 48824 S 0.3 1.2 2:30.21 dockerd
3883 ritik 20 0 1809708 548044 380036 S 0.3 6.8 13:23.99 chrome
4397 ritik 20 0 4808888 80796 62392 S 0.3 1.0 0:04.68 chrome
4465 ritik 20 0 4808888 81908 62384 S 0.3 1.0 0:04.67 chrome
4497 ritik 20 0 4808888 84412 65076 S 0.3 1.1 0:04.72 chrome
4525 ritik 20 0 4809912 85448 64672 S 0.3 1.1 0:04.84 chrome
23551 root 20 0 0 0 0 I 0.3 0.0 0:02.00 kworker/u16:0-e
23613 ritik 20 0 3772092 92788 56616 S 0.3 1.2 0:05.72 vsls-agent
23897 ritik 20 0 4975744 243588 122580 S 0.3 3.0 0:35.89 chrome
2 root 20 0 0 0 0 S 0.0 0.0 0:00.03 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-kb
9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
10 root 20 0 0 0 0 S 0.0 0.0 0:01.00 ksoftirqd/0
11 root 20 0 0 0 0 I 0.0 0.0 0:29.89 rcu_sched
12 root rt 0 0 0 0 S 0.0 0.0 0:00.10 migration/0
13 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1
16 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1
17 root rt 0 0 0 0 S 0.0 0.0 0:00.08 migration/1
18 root 20 0 0 0 0 S 0.0 0.0 0:00.62 ksoftirqd/1
21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2
22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2
23 root rt 0 0 0 0 S 0.0 0.0 0:00.10 migration/2
24 root 20 0 0 0 0 S 0.0 0.0 0:00.21 ksoftirqd/2
26 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/2:0H-kb
27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3
28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/3
29 root rt 0 0 0 0 S 0.0 0.0 0:00.08 migration/3
30 root 20 0 0 0 0 S 0.0 0.0 0:00.28 ksoftirqd/3
32 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/3:0H-kb
33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/4
34 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/4
35 root rt 0 0 0 0 S 0.0 0.0 0:00.08 migration/4
36 root 20 0 0 0 0 S 0.0 0.0 0:00.22 ksoftirqd/4
39 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/5
ritik@ritik-X510UNR:~$


Activities
Terminal
File Edit View Search Terminal Help
ritik@ritik-X510UNR:~$ nproc --all
8
ritik@ritik-X510UNR:~$ echo "Threads/core: $(nproc --all)"
Threads/core: 8
ritik@ritik-X510UNR:~$

1.Write a C/C++ simple parallel program to display the *thread_id* and total number of threads.

```c
#include<omp.h>
#include<stdio.h>
int main(){
   int nthreads,tid;
   #pragma omp parallel private(tid){
       tid=omp_get_thread_num();
       printf("Hello world from thread=%d\n",tid);
       if(tid==0){
           nthreads=omp_get_num_threads();
           printf("Number of threads=%d\n",nthreads);
       }
   }
}
```

o/p:

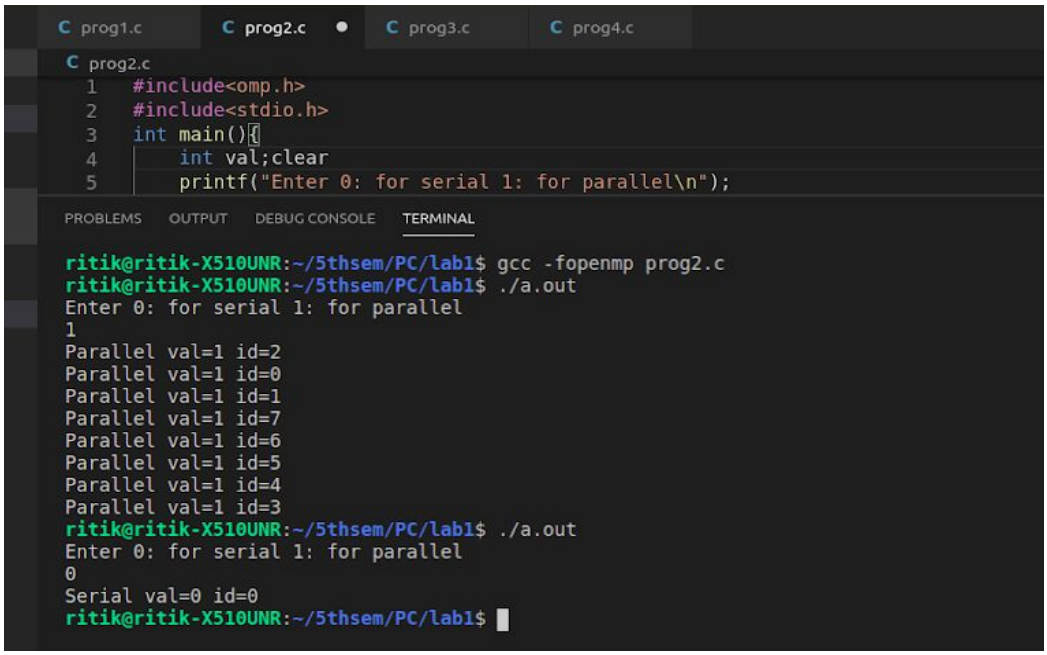


Here when we will change thread num no. of thread o/p changes accordingly.(default=8)

2.Check the output of following program:

```
#include<omp.h>
#include<stdio.h>
int main(){
   int val;
   printf("Enter 0: for serial 1: for parallel\n");
   scanf("%d",&val);
   #pragma omp parallel if(val)
   if(omp_in_parallel()) printf("Parallel val=%d id=%d\n",val,
omp_get_thread_num());
   else printf("Serial val=%d id=%d\n",val, omp_get_thread_num());
}
```

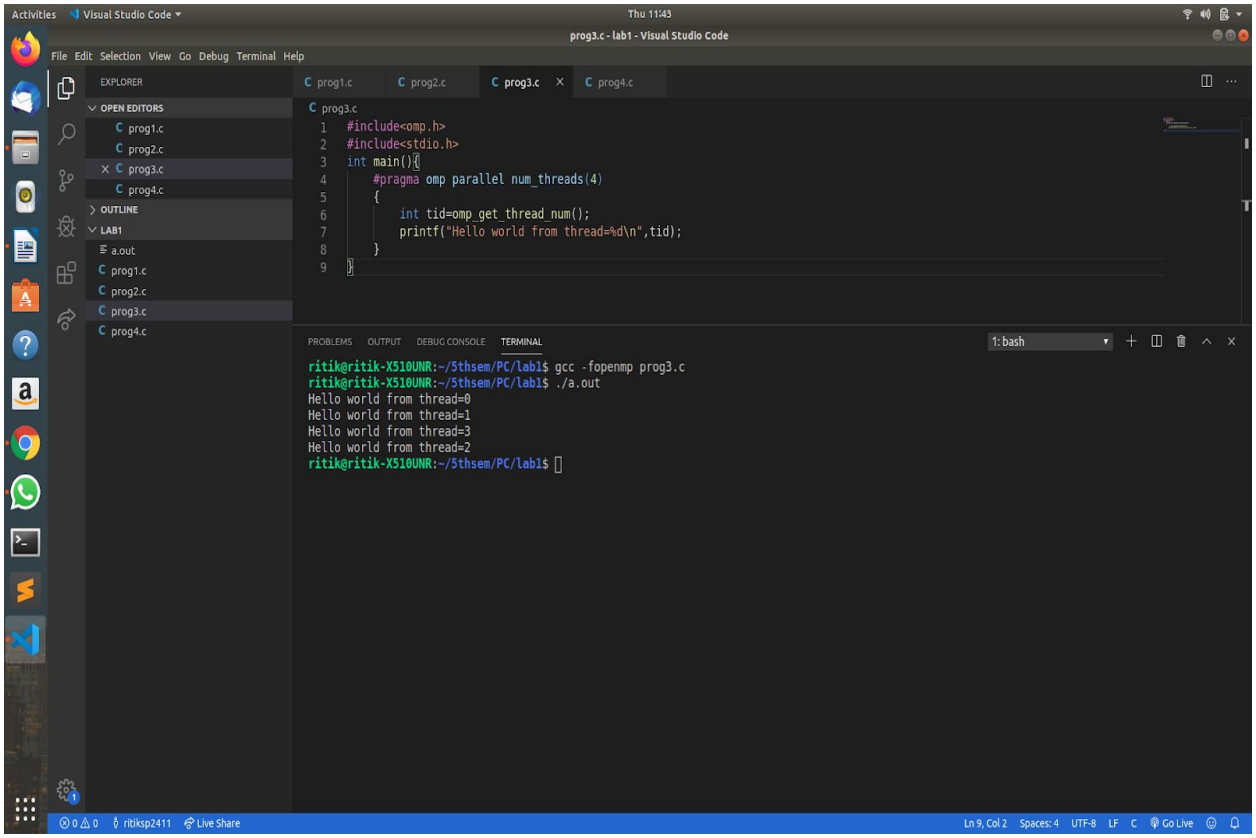
o/p:



Here we have both serial and parallel execution. So serial has 1 thread but parallel has 8 thread as default.

For changing the no. of threads:
export OMP_NUM_THREADS=2 (mention it in terminal before running)
num_threads() (mention it in line where we write #pragma)
omp_set_num_threads() (mention it inside the main code)

3.Observe and record the output of the following program.

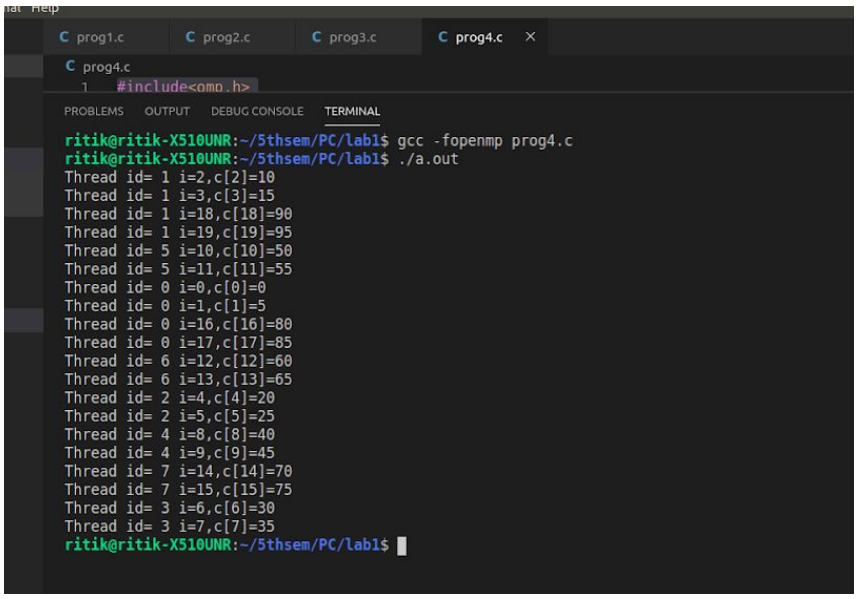


Here we have num_thread(4) so we have only 4 threads .

4.Write a C/C++ parallel program for adding corresponding elements of two arrays.

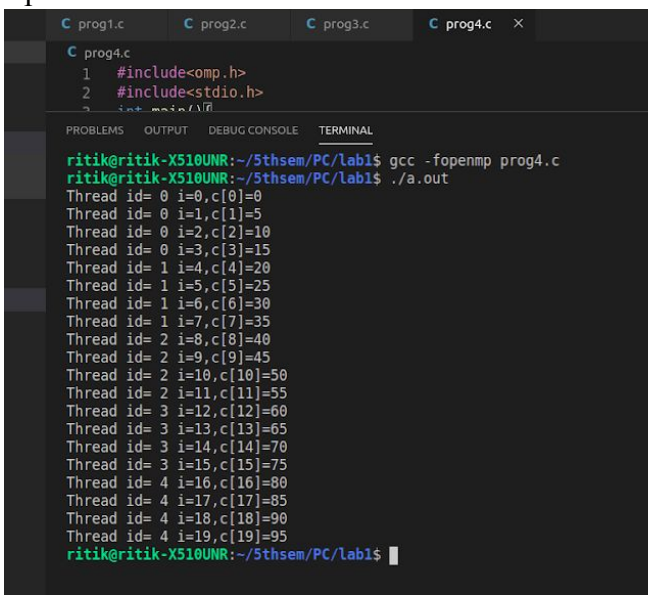
```
#include<omp.h>

#include<stdio.h>

int main(){

   int i,n,chunk;

   int a[20],b[20],c[20];

   n=20;

   chunk=2;

   /*initializing array*/

   for(i=0;i<n;i++)

   {

       a[i]=i*2;

       b[i]=i*3;

   }

   #pragma omp parallel for default(shared) private(i)
schedule(static,chunk)

   for(i=0;i<n;i++)

   {

       c[i]=a[i]+b[i];

       printf("Thread id= %d i=%d,c[%d]=%d\n",
omp_get_thread_num(),i,i,c[i]);

   }

}
```

o/p



The chunk size is the measure how many iterations should be given to one thread initially.
After that if we need more then they are assigned in a round robin manner.
Here my PC has 8 thread and chunk size is 2 so 16 out of 20 iterations are assigned.
Remaining 4 are assigned in round robin so 0th,1st thread gets 2 more iterations.
So if chunk size is 4 only 5 threads are assigned iterations because we need 20 in total.
So if chunk size is equal to no. of iterations, it is the same as serial execution.
Thus chunk size changes execution changes.

o/p when chunk size is 4 :



-Ritik Pansuriya(181IT237)