DEPARTMENT OF INFORMATION TECHNOLOGY, NITK SURATHKAL

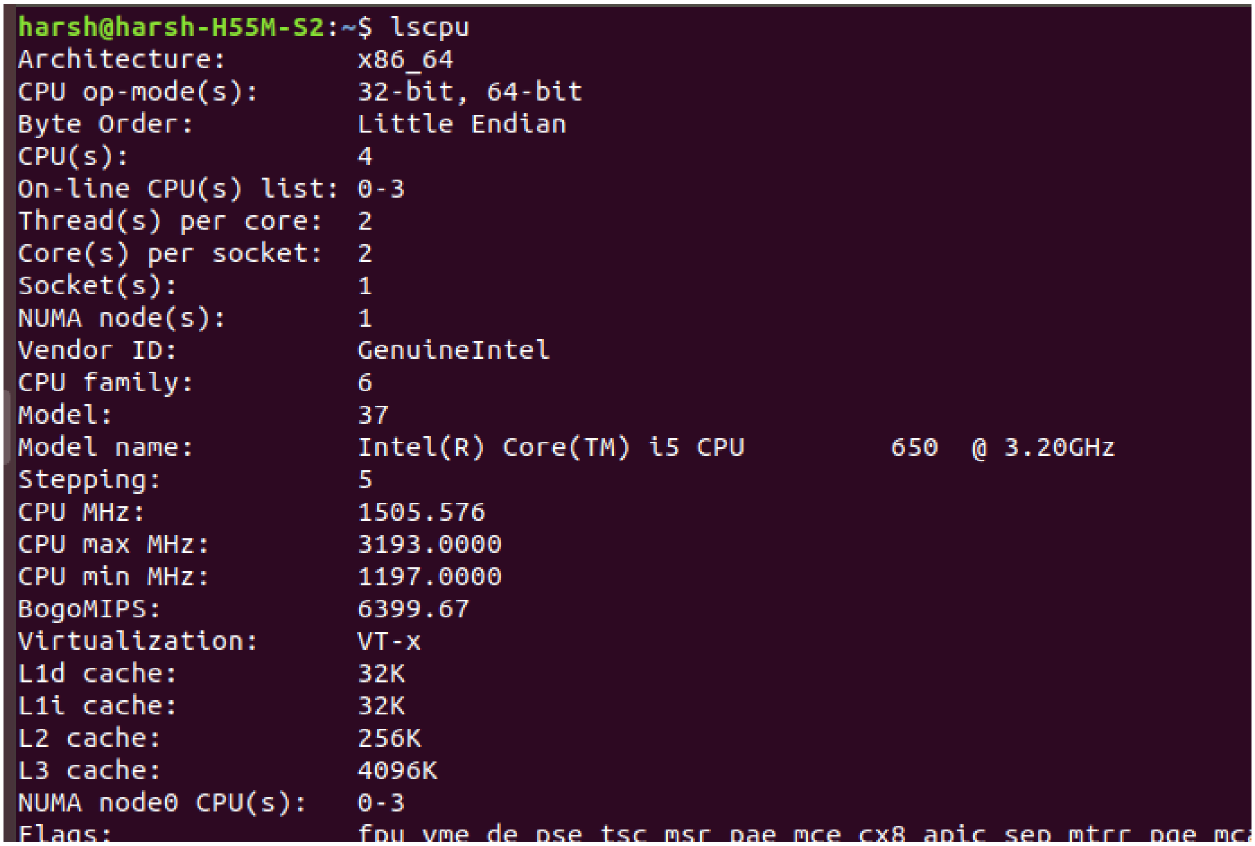
IT301 Parallel Programming

LAB 1 (5th August 2020)

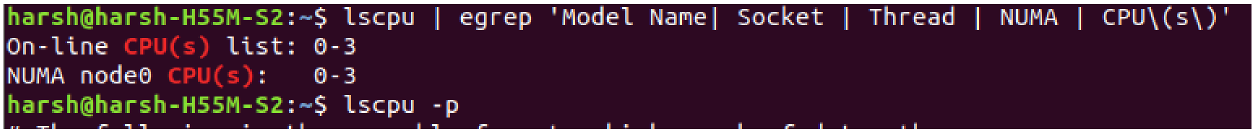
Submitted by -   **Name: Harsh Agarwal**   **Roll No. 181IT117**

1.Finding Number of CPUs in the system

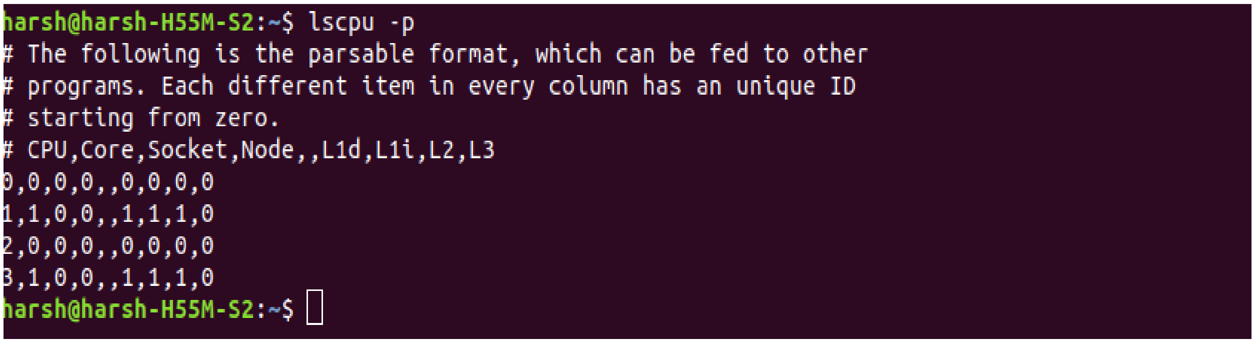
a) Using lscpu command



```
harsh@harsh-H55M-S2:~$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  2
Core(s) per socket:  2
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               37
Model name:          Intel(R) Core(TM) i5 CPU         650  @ 3.20GHz
Stepping:            5
CPU MHz:             1505.576
CPU max MHz:         3193.0000
CPU min MHz:         1197.0000
BogoMIPS:            6399.67
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            4096K
NUMA node0 CPU(s):   0-3
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
```

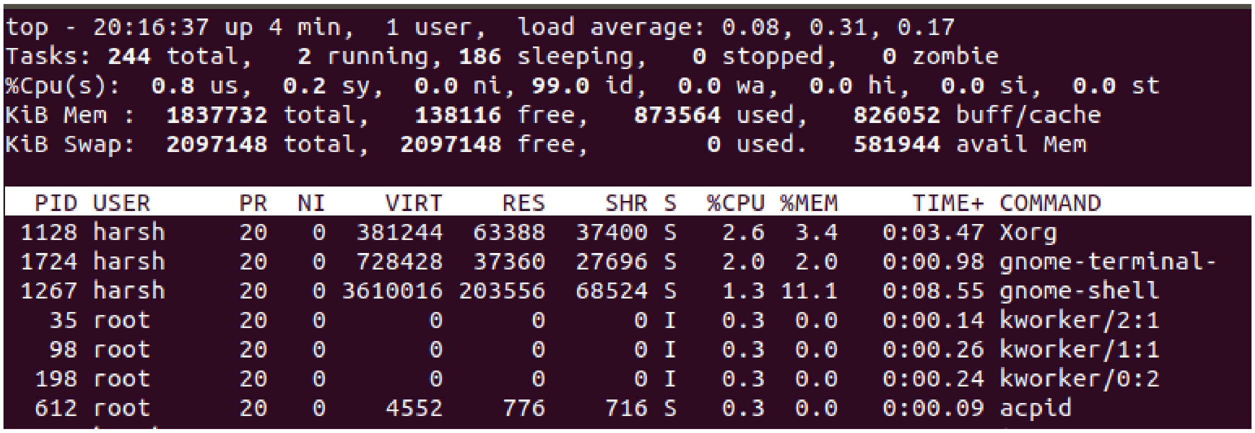


```
harsh@harsh-H55M-S2:~$ lscpu | egrep 'Model Name| Socket | Thread | NUMA | CPU\(s\)'
On-line CPU(s) list: 0-3
NUMA node0 CPU(s):   0-3
harsh@harsh-H55M-S2:~$ lscpu -p
```



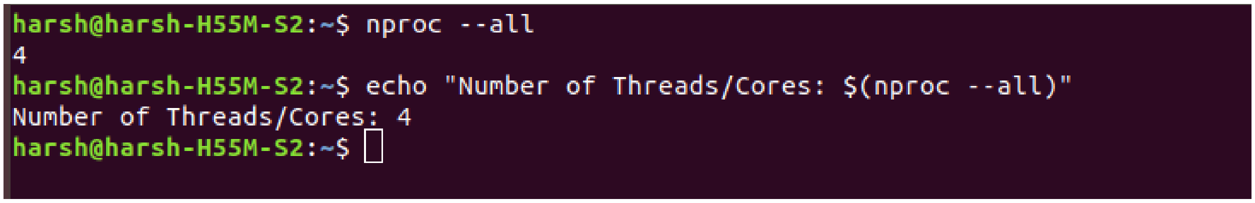
```
harsh@harsh-H55M-S2:~$ lscpu -p
# The following is the parsable format, which can be fed to other
# programs. Each different item in every column has an unique ID
# starting from zero.
# CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0,0,0,0,,0,0,0,0
1,1,0,0,,1,1,1,0
2,0,0,0,,0,0,0,0
3,1,0,0,,1,1,1,0
harsh@harsh-H55M-S2:~$ 
```

b) Using top command



```
top - 20:16:37 up 4 min,  1 user,  load average: 0.08, 0.31, 0.17
Tasks: 244 total,   2 running, 186 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.8 us,  0.2 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1837732 total,   138116 free,   873564 used,   826052 buff/cache
KiB Swap:  2097148 total,  2097148 free,        0 used.   581944 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1128 harsh     20   0  381244  63388  37400 S   2.6  3.4   0:03.47 Xorg
 1724 harsh     20   0  728428  37360  27696 S   2.0  2.0   0:00.98 gnome-terminal-
 1267 harsh     20   0 3610016 203556  68524 S   1.3 11.1   0:08.55 gnome-shell
   35 root      20   0       0      0      0 I   0.3  0.0   0:00.14 kworker/2:1
   98 root      20   0       0      0      0 I   0.3  0.0   0:00.26 kworker/1:1
  198 root      20   0       0      0      0 I   0.3  0.0   0:00.24 kworker/0:2
  612 root      20   0    4552    776    716 S   0.3  0.0   0:00.09 acpid
```

c) Using nproc command



```
harsh@harsh-H55M-S2:~$ nproc --all
4
harsh@harsh-H55M-S2:~$ echo "Number of Threads/Cores: $(nproc --all)"
Number of Threads/Cores: 4
harsh@harsh-H55M-S2:~$ 
```

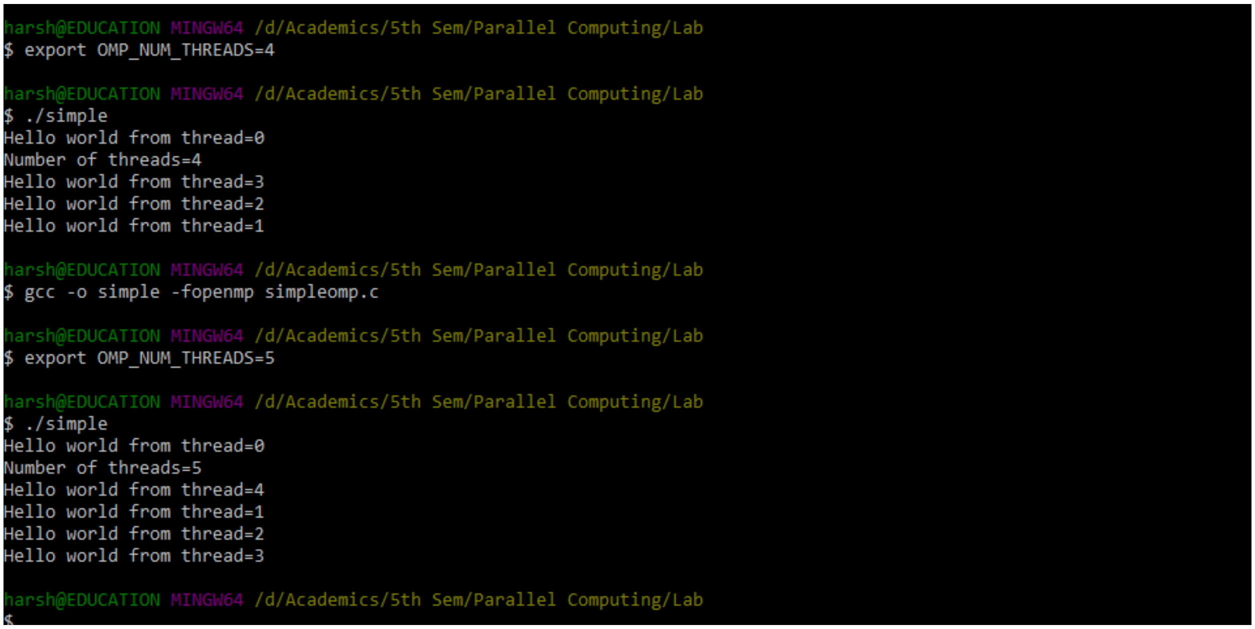
2.Write a C/C++ simple parallel program to display the *thread_id* and total number of threads.

Program:

```
#include<stdio.h>
```

```
#include<stdlib.h>
#include<omp.h>
int main(){
        int nthreads, tid;
        omp_set_num_threads(4);
        #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);
                }
        }
}
```

Output:



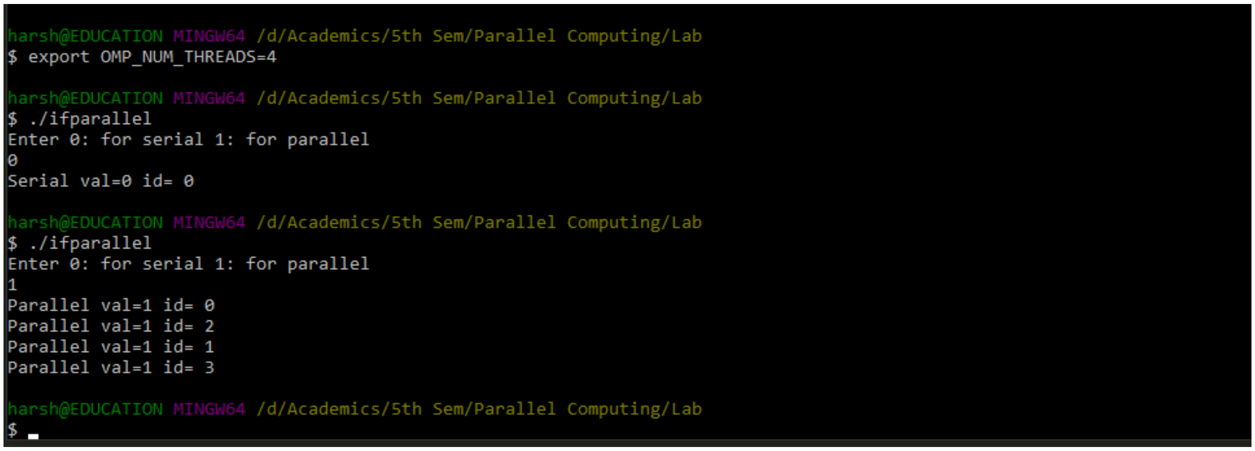
```
harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ export OMP_NUM_THREADS=4

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./simple
Hello world from thread=0
Number of threads=4
Hello world from thread=3
Hello world from thread=2
Hello world from thread=1

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ gcc -o simple -fopenmp simpleomp.c

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ export OMP_NUM_THREADS=5

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./simple
Hello world from thread=0
Number of threads=5
Hello world from thread=4
Hello world from thread=1
Hello world from thread=2
Hello world from thread=3

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$
```

2. Parallel or serial execution:

Program:

```
#include<stdio.h>

#include<stdlib.h>

#include<omp.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());

        }

}
```

Output:



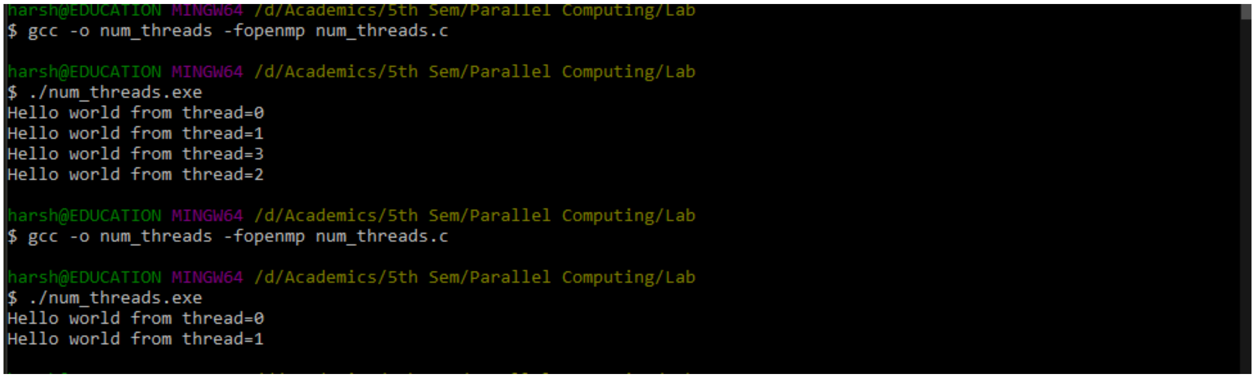
```
harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ export OMP_NUM_THREADS=4

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./ifparallel
Enter 0: for serial 1: for parallel
0
Serial val=0 id= 0

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./ifparallel
Enter 0: for serial 1: for parallel
1
Parallel val=1 id= 0
Parallel val=1 id= 2
Parallel val=1 id= 1
Parallel val=1 id= 3

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ 
```

3.Number of Threads

Program:

```
#include<stdio.h>

#include<stdlib.h>

#include<omp.h>

int main(){

        #pragma omp parallel num_threads(2)

        {

                int i=omp_get_thread_num();

                printf("Hello world from thread=%d\n",i);

        }

}
```

Output:



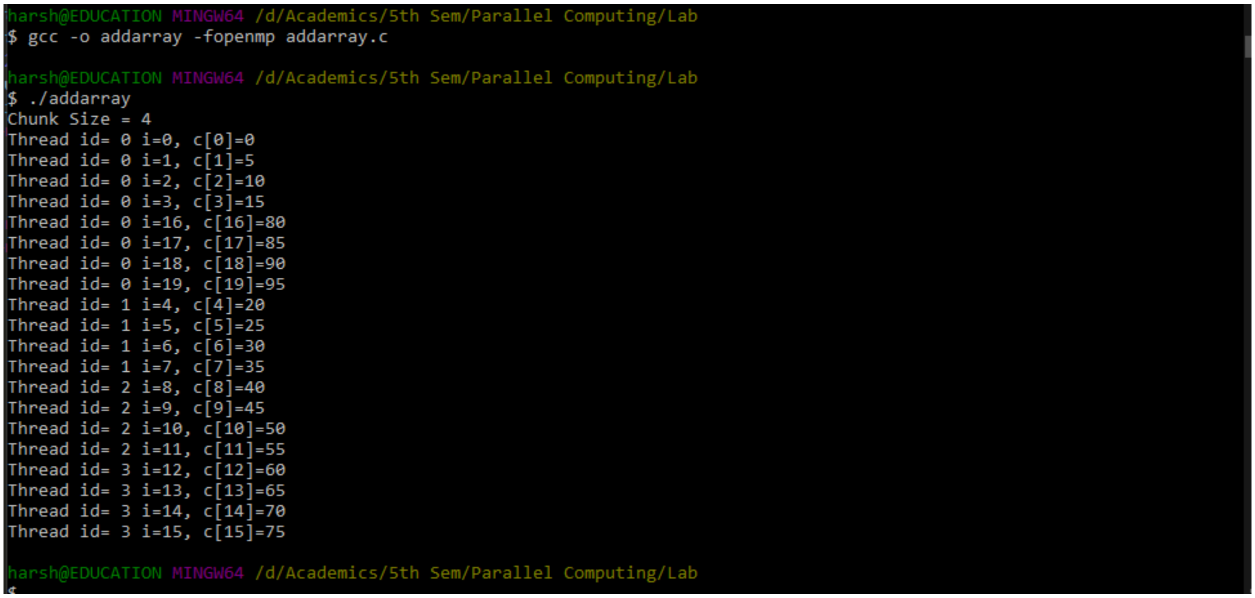
```
harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ gcc -o num_threads -fopenmp num_threads.c

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./num_threads.exe
Hello world from thread=0
Hello world from thread=1
Hello world from thread=3
Hello world from thread=2

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ gcc -o num_threads -fopenmp num_threads.c

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./num_threads.exe
Hello world from thread=0
Hello world from thread=1
```

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

Program:

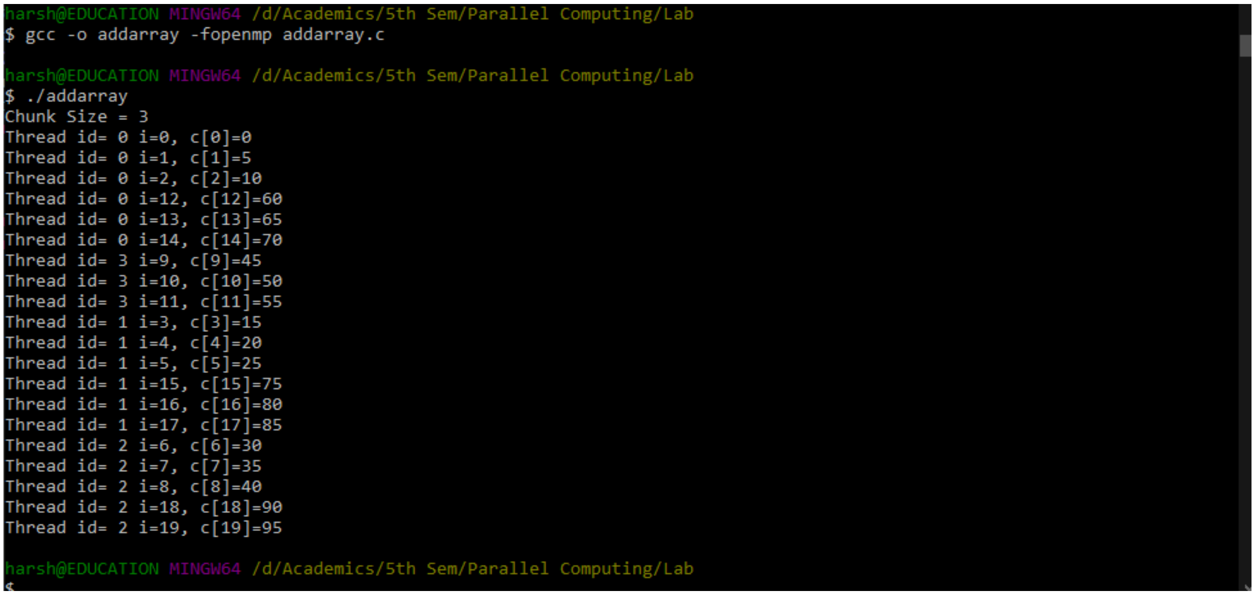
```
#include<stdio.h>

#include<stdlib.h>

#include<omp.h>

int main(){

        int i,n,chunk;

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

        n=20;

        chunk=3;

        printf("Chunk Size = %d\n", chunk);

        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]);

        }

}
```

Output_1: (With Chunk size = 4)



```
harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ gcc -o addarray -fopenmp addarray.c

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./addarray
Chunk Size = 4
Thread id= 0 i=0, c[0]=0
Thread id= 0 i=1, c[1]=5
Thread id= 0 i=2, c[2]=10
Thread id= 0 i=3, c[3]=15
Thread id= 0 i=16, c[16]=80
Thread id= 0 i=17, c[17]=85
Thread id= 0 i=18, c[18]=90
Thread id= 0 i=19, c[19]=95
Thread id= 1 i=4, c[4]=20
Thread id= 1 i=5, c[5]=25
Thread id= 1 i=6, c[6]=30
Thread id= 1 i=7, c[7]=35
Thread id= 2 i=8, c[8]=40
Thread id= 2 i=9, c[9]=45
Thread id= 2 i=10, c[10]=50
Thread id= 2 i=11, c[11]=55
Thread id= 3 i=12, c[12]=60
Thread id= 3 i=13, c[13]=65
Thread id= 3 i=14, c[14]=70
Thread id= 3 i=15, c[15]=75

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$
```

Output_2: (With Chunk size = 3)



```
harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ gcc -o addarray -fopenmp addarray.c

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$ ./addarray
Chunk Size = 3
Thread id= 0 i=0, c[0]=0
Thread id= 0 i=1, c[1]=5
Thread id= 0 i=2, c[2]=10
Thread id= 0 i=12, c[12]=60
Thread id= 0 i=13, c[13]=65
Thread id= 0 i=14, c[14]=70
Thread id= 3 i=9, c[9]=45
Thread id= 3 i=10, c[10]=50
Thread id= 3 i=11, c[11]=55
Thread id= 1 i=3, c[3]=15
Thread id= 1 i=4, c[4]=20
Thread id= 1 i=5, c[5]=25
Thread id= 1 i=15, c[15]=75
Thread id= 1 i=16, c[16]=80
Thread id= 1 i=17, c[17]=85
Thread id= 2 i=6, c[6]=30
Thread id= 2 i=7, c[7]=35
Thread id= 2 i=8, c[8]=40
Thread id= 2 i=18, c[18]=90
Thread id= 2 i=19, c[19]=95

harsh@EDUCATION MINGW64 /d/Academics/5th Sem/Parallel Computing/Lab
$
```