IT301 Assignment 1

NAME: SUYASH CHINTAWAR

COURSE: PARALLEL COMPUTING

TOPIC: LAB 1

Q1. Find the number of CPUs in system

A) i) lscpu – Hence, no. of CPUs in the system is 8.

```
ubuntu@suyash-18-04:~$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               126
Model name:          Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
Stepping:            5
CPU MHz:             1292.757
CPU max MHz:         3600.0000
CPU min MHz:         400.0000
BogoMIPS:            2380.80
Virtualization:      VT-x
L1d cache:           48K
L1i cache:           32K
L2 cache:            512K
L3 cache:            6144K
NUMA node0 CPU(s):   0-7
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs
 bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_t
imer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep b
mi2 erms invpcid avx512f avx512dq rdseed adx smap avx512ifma clflushopt intel_pt avx512cd sha_ni avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_
epp hwp_pkg_req avx512vbmi umip pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg avx512_vpopcntdq rdpid md_clear flush_l1d arch_capabilities
ubuntu@suyash-18-04:~$
```

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

```
ubuntu@suyash-18-04:~$ lscpu | egrep 'Model name|Socket|Thread|NUMA|CPU\(s\)'
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Socket(s):           1
NUMA node(s):        1
Model name:          Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
NUMA node0 CPU(s):   0-7
ubuntu@suyash-18-04:~$
```

iii) lscpu -p

```
ubuntu@suyash-18-04:~$ 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
ubuntu@suyash-18-04:~$
```

B) top command

```
ubuntu@suyash-18-04:~$ top

top - 16:48:35 up  3:10,  1 user,  load average: 0.33, 0.45, 0.59
Tasks: 344 total,   1 running, 263 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.5 sy,  0.0 ni, 98.3 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  7817316 total,  1705284 free,  3052916 used,  3059116 buff/cache
KiB Swap:  2097148 total,  2089200 free,     7948 used.  3622888 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
10075 ubuntu    20   0 52.823g 472652 107680 S   5.3  6.0   5:22.03 chrome
 1589 ubuntu    20   0 1425476 121284  82048 S   3.0  1.6   3:26.09 Xorg
 2714 ubuntu    20   0  739200 155704  82532 S   2.0  2.0   6:56.87 chrome
  401 root     -51   0       0      0      0 S   1.3  0.0   0:52.50 irq/128-MSFT000
15638 ubuntu    20   0  802132  37728  28040 S   1.3  0.5   0:00.60 gnome-terminal-
 1737 ubuntu    20   0 4126316 273584 122624 S   1.0  3.5   4:02.08 gnome-shell
    1 root      20   0  225504   9172   6648 S   0.3  0.1   0:18.91 systemd
 1716 ubuntu    20   0  220792   6904   6180 S   0.3  0.1   0:01.59 at-spi2-registr
10627 ubuntu    20   0 7674500 220520 104628 S   0.3  2.8   0:22.70 teams
10718 ubuntu    20   0 13.742g 365924  85512 S   0.3  4.7   0:57.37 teams
13290 root      20   0       0      0      0 I   0.3  0.0   0:05.03 kworker/6:0-eve
16659 ubuntu    20   0   51456   4108   3360 R   0.3  0.1   0:00.03 top
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 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   0:00.10 ksoftirqd/0
   11 root      20   0       0      0      0 I   0.0  0.0   0:09.55 rcu_sched
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.04 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   0:00.06 ksoftirqd/1
   20 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H-kb
   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   0:00.09 migration/2
   24 root      20   0       0      0      0 S   0.0  0.0   0:00.06 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   0:00.10 migration/3
   30 root      20   0       0      0      0 S   0.0  0.0   0:00.07 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   0:00.11 migration/4
   36 root      20   0       0      0      0 S   0.0  0.0   0:00.05 ksoftirqd/4
   38 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/4:0H-kb
   39 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/5
   40 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/5
```

C) i) nproc –all

```
ubuntu@suyash-18-04:~$ nproc --all
8
ubuntu@suyash-18-04:~$
```

ii) echo "Threads/core: $(nproc –all)"

```
ubuntu@suyash-18-04:~$ echo "Threads/core: $(nproc --all)"
Threads/core: 8
ubuntu@suyash-18-04:~$
```

Q2. Write a C/C++ simple parallel program to display the thread_id and total number of threads.
Ans:

Method 1: Using OMP_NUM_THREADS

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp simpleomp.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ export OMP_NUM_THREADS=2
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Hello world from thread=0
Number of threads=2
Hello world from thread=1
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

Method 2: Using omp_set_num_threads(x),

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp simpleomp.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Hello world from thread=0
Number of threads (using omp_set_num_threads()) = 4
Hello world from thread=3
Hello world from thread=2
Hello world from thread=1
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

Program for omp_set_num_threads(x),

```c
/*simpleomp.c*/
#include<omp.h>
#include<stdio.h>

int main(){
    int nthreads,tid;
    //=========================================
    omp_set_num_threads(4);//setting 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 (using omp_set_num_threads()) = %d\n",nthreads);
        }
    }
}
```

Method 3: using num_threads(),

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp simpleomp.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Hello world from thread=0
Number of threads (using num_threads()) = 8
Hello world from thread=6
Hello world from thread=2
Hello world from thread=5
Hello world from thread=7
Hello world from thread=1
Hello world from thread=3
Hello world from thread=4
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

Program for num_threads(),

```c
/*simpleomp.c*/
#include<omp.h>
#include<stdio.h>

int main(){
    int nthreads,tid;
    #pragma omp parallel private(tid) num_threads(8) //set num_threads=8
    {
        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 (using num_threads()) = %d\n",nthreads);
        }
    }
}
```

The use of if clause to determine parallel execution is similar to Q3 below.

Q3. Check the output of following program and Note down the output in your observation book. (ifparallel.c)
Ans.

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp ifparallel.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Enter 0: for serial 1: for parallel
0
Serial val=0 id= 0
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Enter 0: for serial 1: for parallel
1
Parallel val=1 id= 0
Parallel val=1 id= 7
Parallel val=1 id= 2
Parallel val=1 id= 3
Parallel val=1 id= 5
Parallel val=1 id= 1
Parallel val=1 id= 4
Parallel val=1 id= 6
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

When 0 is entered, serial execution of program occurs and parallel execution if 1 is entered.

Q4. Observe and record the output of following program. Change the num_threads and observe the result. (num_threads.c)
Ans.

When num_threads=4,

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp num_threads.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Hello world from thread=0
Hello world from thread=2
Hello world from thread=1
Hello world from thread=3
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

When num_threads=16,

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp num_threads.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Hello world from thread=1
Hello world from thread=6
Hello world from thread=4
Hello world from thread=3
Hello world from thread=7
Hello world from thread=8
Hello world from thread=2
Hello world from thread=5
Hello world from thread=9
Hello world from thread=10
Hello world from thread=13
Hello world from thread=11
Hello world from thread=15
Hello world from thread=0
Hello world from thread=12
Hello world from thread=14
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

When num_threads=2,

```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ gcc -o simple -fopenmp num_threads.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$ ./simple
Hello world from thread=0
Hello world from thread=1
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 1$
```

num_threads specify the number of threads to be used for program execution. The part of the program specified under parallem execution gets executed by all the threads and hence we get “Hello World” message from all threads. The order in which threads execute may differ as shown in above output.

THANK YOU