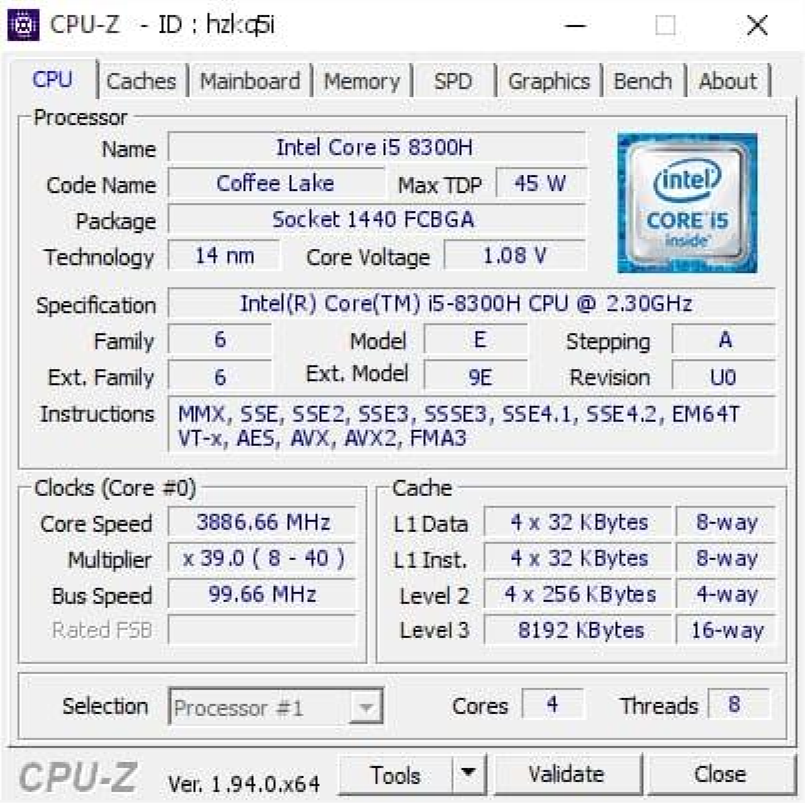
HPC LAB - Matrix Addition

Name: Paleti Krishnasai
Roll No: CED18I039
Programming Environment: OpenMP
Problem: Matrix Addition
Date: 19th August 2021

Hardware Configuration:
PU NAME: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Number of Sockets: 1
Cores per Socket : 4
Threads per core: 2
L1d cache: 128 KiB
L1i cache: 128 KiB
L2 cache: 1 MiB
L3 cache: 8 MiB



Processor
Name: Intel Core i5 8300H
Code Name: Coffee Lake
Max TDP: 45 W
Package: Socket 1440 FCBGA
Technology: 14 nm
Core Voltage: 1.08 V
Specification: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Family: 6
Model: E
Stepping: A
Ext. Family: 6
Ext. Model: 9E
Revision: U0
Instructions: MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T VT-x, AES, AVX, AVX2, FMA3

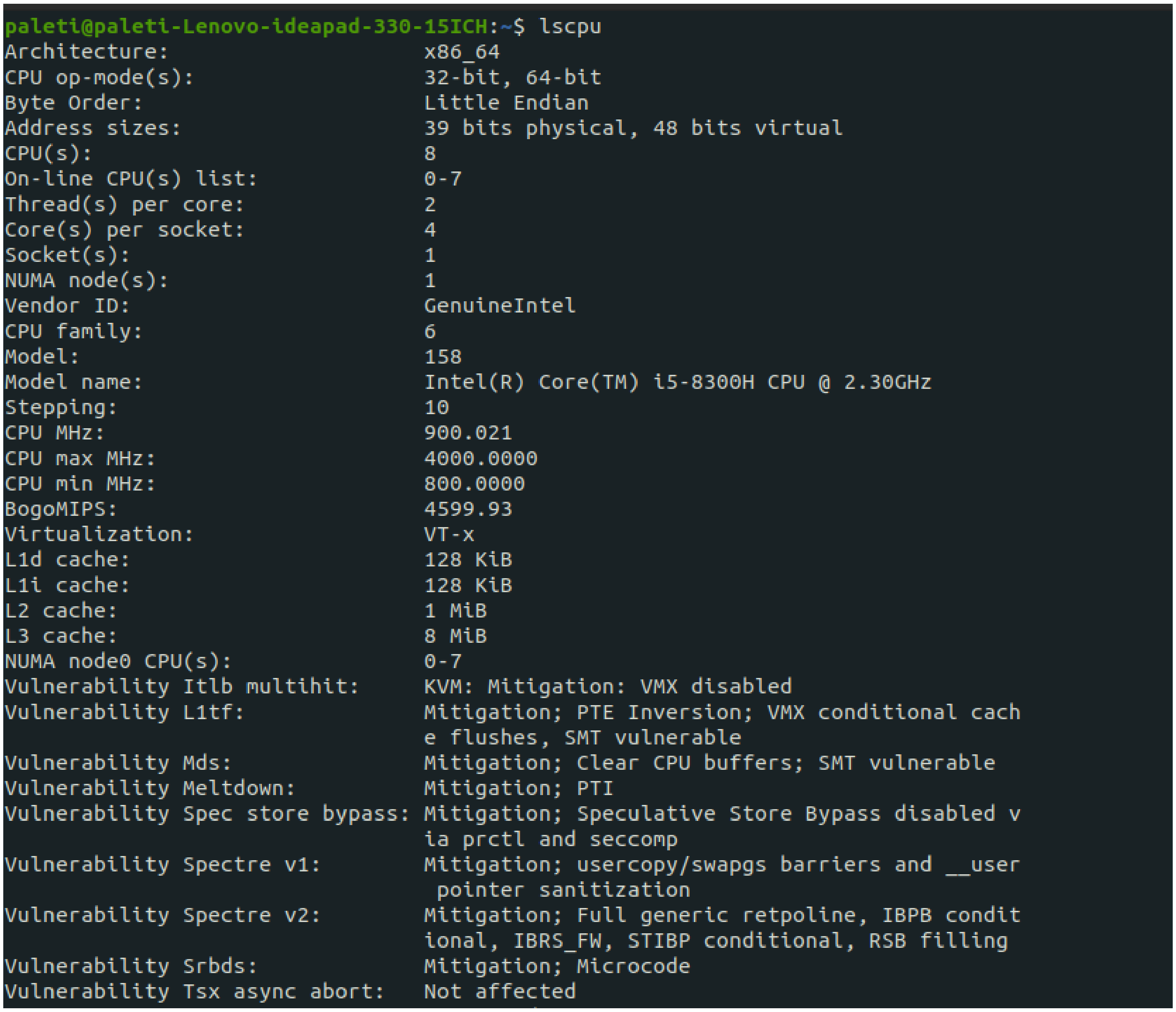
Clocks (Core #0)
Core Speed: 3886.66 MHz
Multiplier: x 39.0 (8 - 40)
Bus Speed: 99.66 MHz
Rated FSB:

Cache
L1 Data	4 x 32 KBytes	8-way
L1 Inst.	4 x 32 KBytes	8-way
Level 2	4 x 256 KBytes	4-way
Level 3	8192 KBytes	16-way

Selection: Processor #1
Cores: 4
Threads: 8

Ver. 1.94.0.x64

Paleti Krishnasai
CED18I039



```
paleti@paleti-Lenovo-ideapad-330-15ICH:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
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:                           158
Model name:                      Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
Stepping:                        10
CPU MHz:                         900.021
CPU max MHz:                     4000.0000
CPU min MHz:                     800.0000
BogoMIPS:                        4599.93
Virtualization:                  VT-x
L1d cache:                       128 KiB
L1i cache:                       128 KiB
L2 cache:                        1 MiB
L3 cache:                        8 MiB
NUMA node0 CPU(s):               0-7
Vulnerability Itlb multihit:     KVM: Mitigation: VMX disabled
Vulnerability L1tf:              Mitigation; PTE Inversion; VMX conditional cach
                                 e flushes, SMT vulnerable
Vulnerability Mds:               Mitigation; Clear CPU buffers; SMT vulnerable
Vulnerability Meltdown:          Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled v
                                 ia prctl and seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user
                                  pointer sanitization
Vulnerability Spectre v2:        Mitigation; Full generic retpoline, IBPB condit
                                 ional, IBRS_FW, STIBP conditional, RSB filling
Vulnerability Srbds:             Mitigation; Microcode
Vulnerability Tsx async abort:   Not affected
```

Paleti Krishnasai
CED18I039

Serial Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

#define size 400

int main()
{
   Double
M1[size][size]={1075.75},M2[size][size]={1594.97},M3[size][size];
   float startTime,endTime,execTime;
   int i,j;
   int omp_rank;

   startTime = omp_get_wtime();
   {
       omp_rank = omp_get_thread_num();
       for(i=0;i<400;i++)
       {
           for(j=0;j<400;j++)
           {
               for(int k=0;k<100000;k++)
                   M3[i][j] = M1[i][j] + M2[i][j];
           }
       }
   }
   endTime = omp_get_wtime();

   execTime = endTime-startTime;
   printf("%f \n",execTime);
   return(0);
}
```

Parallel Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

#define size 400

int main()
{
   long double
M1[size][size]={1075.75},M2[size][size]={1594.97},M3[size][size];
   float startTime,endTime,execTime;
   int i,j;
   int omp_rank;

   startTime = omp_get_wtime();
   #pragma omp parallel private (i,j) shared (M1,M2,M3)
   {
       omp_rank = omp_get_thread_num();
       #pragma omp for
       for(i=0;i<250;i++)
       {
           for(j=0;j<250;j++)
           {
               for(int k=0;k<100000;k++)
                   M3[i][j] = M1[i][j] + M2[i][j];
           }
       }
   }
   endTime = omp_get_wtime();

   execTime = endTime-startTime;
   printf("%f \n",execTime);
   return(0);
}
```

Paleti Krishnasai
CED18I039

Compilation and Execution:

To enable OpenMP environment, use -fopenmp flag while compiling using gcc.

```
gcc -O0 -fopenmp  matrix_add_mp.c
```

Then,

```
export OMP_NUM_THREADS= no of threads for parallel execution.
```

```
./a.out
```

Observations :

Threads (n)	Runtime	Speedup (s)	Parallelization Fraction
1	55.125	1	
2	27.640625	1.994347089	0.9971655329
4	17.144531	3.215311052	0.9186507997
8	14.648438	3.763199872	0.8391642268
16	14.230469	3.873730374	0.7913076293
32	13.699219	4.023952022	0.775729642
64	13.949219	3.951834149	0.7588093392
128	13.53125	4.073903002	0.760476369

Speed up can be found using the following formula,
$S(n)=T(1)/T(n)$
where, **$S(n)$** = Speedup for thread count ‘n’
$T(1)$ = Execution Time for Thread count ‘1’ (serial code)
$T(n)$ = Execution Time for Thread count ‘n’ (serial code)

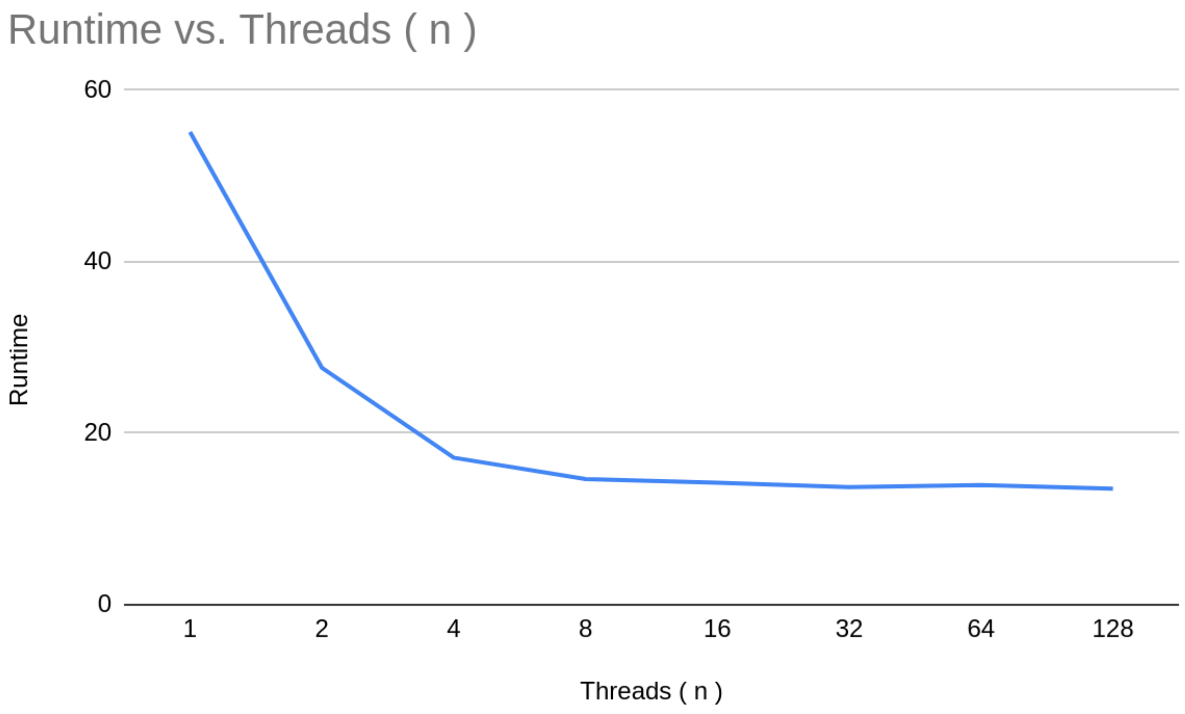
Parallelization Fraction can be found using the following formula,
$S(n)=1/((1 - p) + p/n)$
where, **$S(n)$** = Speedup for thread count ‘n’
n = Number of threads
p = Parallelization fraction

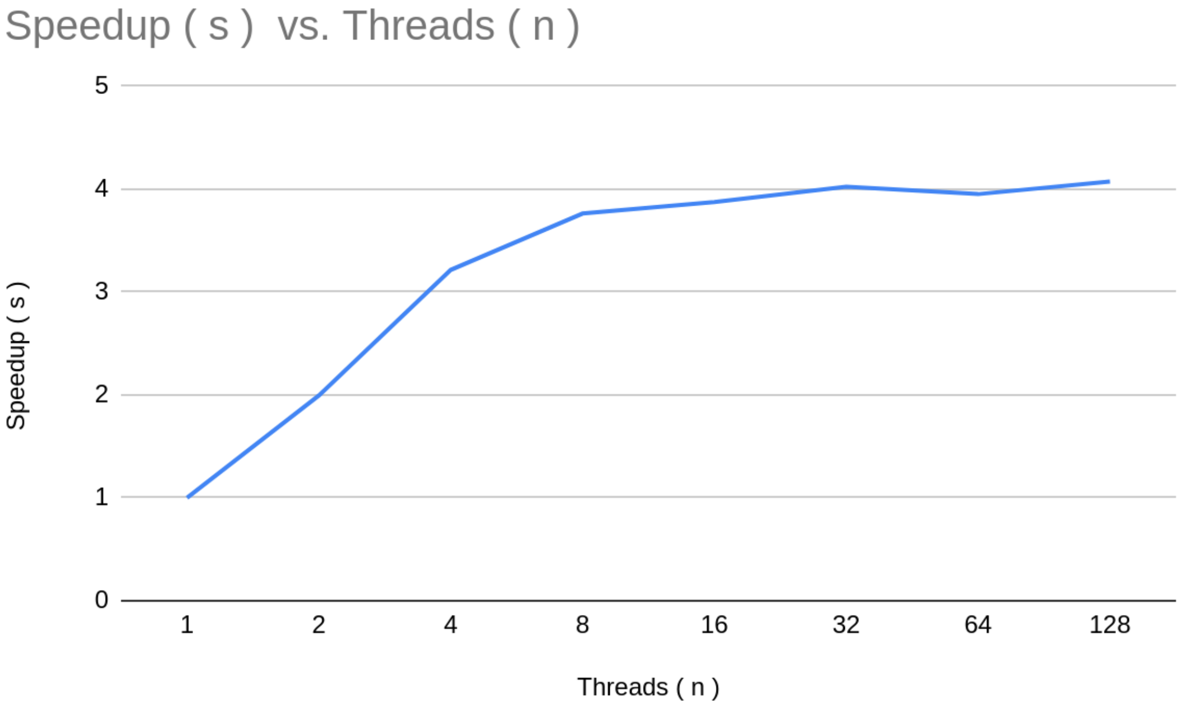
Paleti Krishnasai
CED18I039

Assumption:

Following extra for loop is added to increase the number of operations in the parallel region to visualize the effect of multi-threading in matrix addition.

for(int k=0;k<10000;k++)





Inference: (**Note**: Execution time, graph and inference will be based on hardware configuration)

- At thread count 128 maximum speedup is observed.
- The Runtime is least at thread count is 128, but it fluctuates from there on rising slightly. This may be due to the fact that there are 4 cores and 2 threads on each core are active at a moment for this task concurrently.