INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KANCHEEPURAM

LAB ASSIGNMENT 6 - REPORT
ON
ADDITION OF N NUMBERS
AND
VECTOR DOT PRODUCT IN CUDA

SUBMITTED BY
AMAR KUMAR
(CED17I029)
TO
DR. NOOR MAHAMMAD

ADDITION OF N NUMBERS

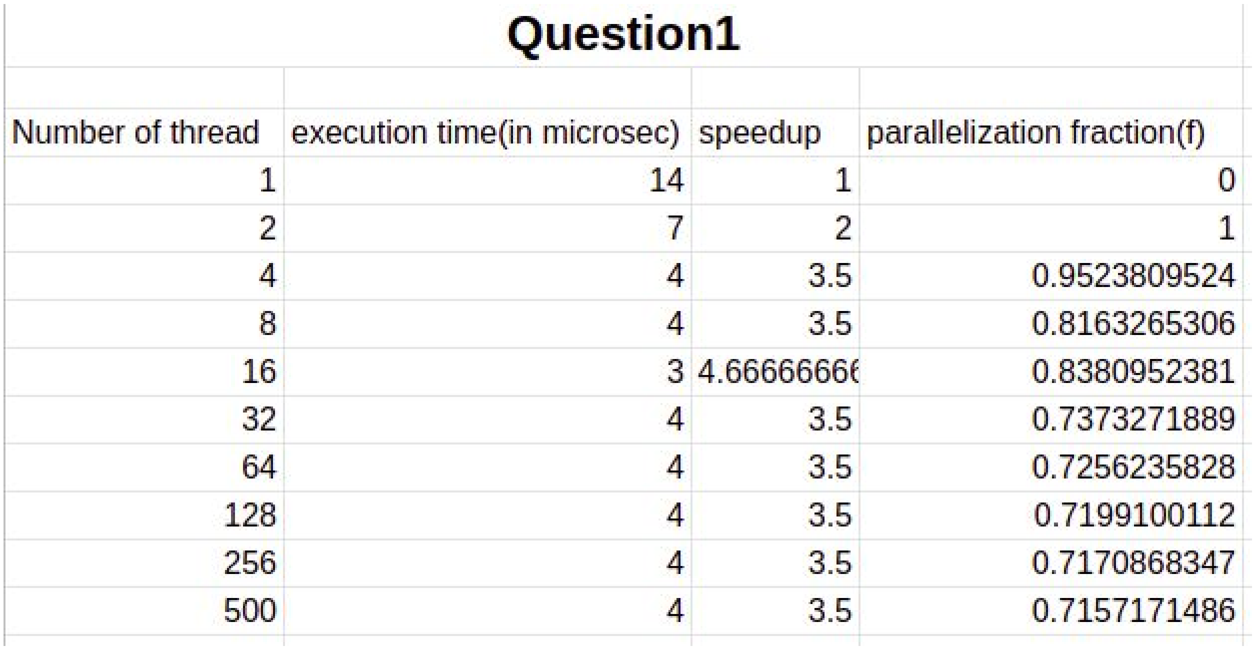
Strategy

In my program for addition of n numbers, the instruction which is running in parallel is **temp[i] =a[i] + b[i]; where b[i] = 0**

Here, i stored every value in temp and then added each value serially as told by sir. But i have used threads to add these number. For example :- suppose i have array of size 4 and number of threads 2 then thread1 will calculate i=0 and i=2 and thread2 will calculate i=1 and i=3. In this way thread is running parallel and thus the above statement is running in parallel.

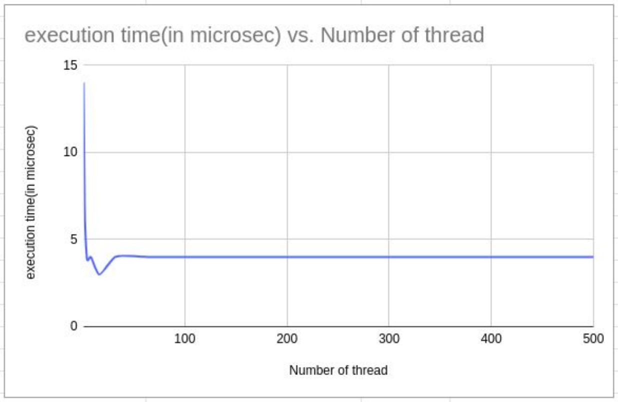
Graph and tables

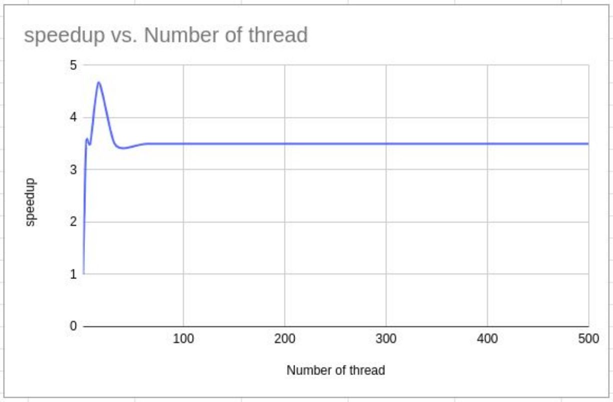
https://docs.google.com/spreadsheets/d/1F8bJfuRkgyIzHSLDMlJ5woJI9g84C_QWAo5C9PUzfLY/edit#gid=0



Question1

Number of thread	execution time(in microsec)	speedup	parallelization fraction(f)
1	14	1	0
2	7	2	1
4	4	3.5	0.9523809524
8	4	3.5	0.8163265306
16	3	4.66666666	0.8380952381
32	4	3.5	0.7373271889
64	4	3.5	0.7256235828
128	4	3.5	0.7199100112
256	4	3.5	0.7170868347
500	4	3.5	0.7157171486





Calculation of parallelization fraction

T(1) = 14 microsecond
Here , for P = 16 the execution time is minimum
T(P) = 3 microsecond
Speedup = $\frac{T(1)}{T(P)}$ = $\frac{14}{3}$ = 4.666666667
From Amdahl's Law,
Speedup = $\frac{1}{(f/P)+(1-f)}$ Where , f = Parallelization factor P = Thread Number

So, f = $\frac{(1-T(P)/T(1))}{(1-(1/P))}$
Therefore, f = 0.8380952381 which means that approx. 83% of the program is parallelizable.

VECTOR DOT PRODUCT

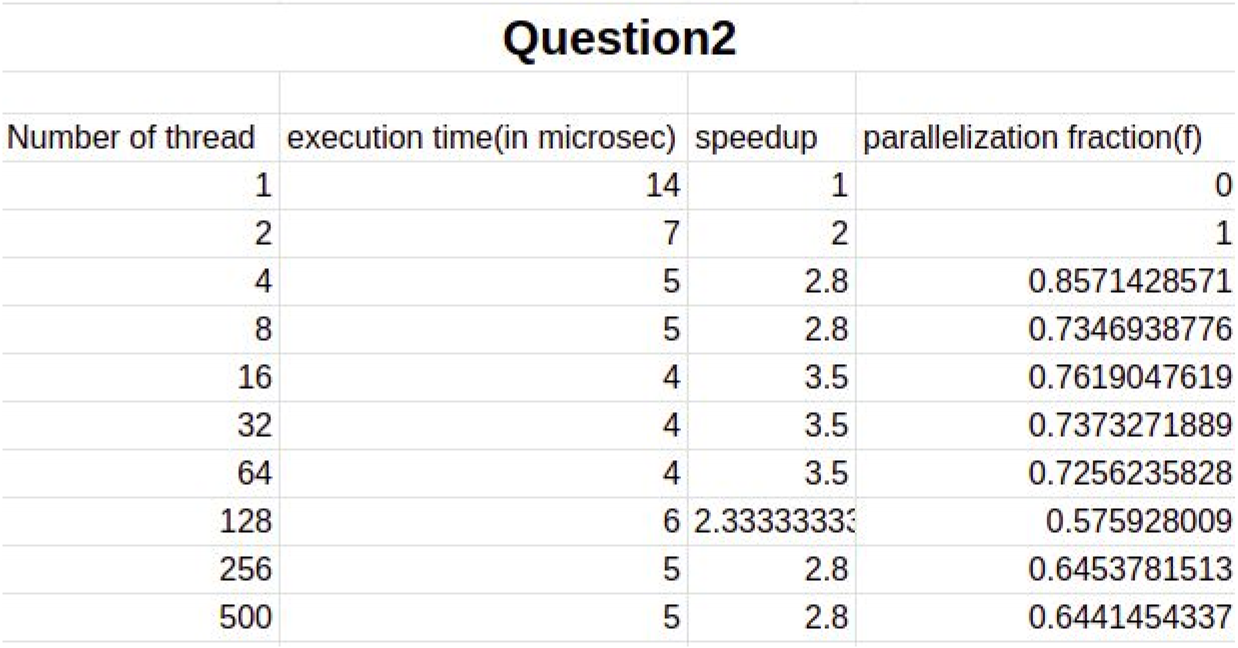
Strategy

In my program for vector dot product of n numbers, the instruction which is running in parallel is
temp[i] = a[i]*b[i];

Firstly, i am multiplying element wise and storing it in temp array and then temp array is being added serially and final sum is stored to c. For example :- suppose i have array of size 4 and number of threads 2 then thread1 will calculate i=0 and i=2 and thread2 will calculate i=1 and i=3. In this way thread is running parallel and thus the above statement is running in parallel.

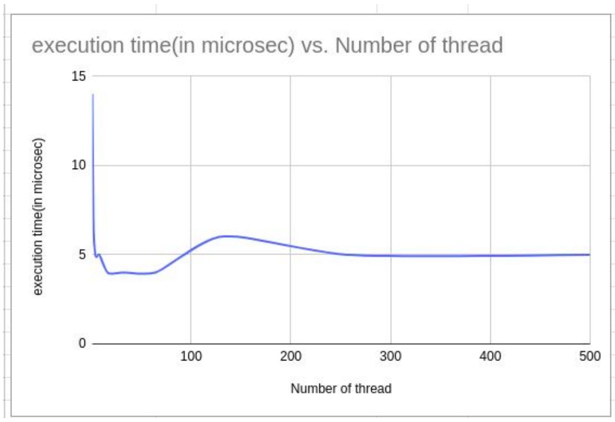
Graph and tables

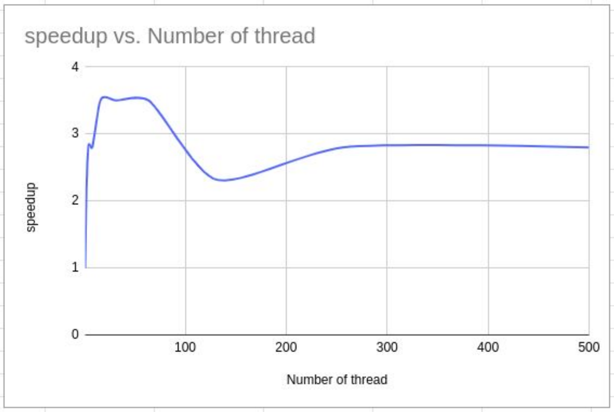
https://docs.google.com/spreadsheets/d/1F8bJfuRkgyIzHSLDMlJ5woJl9g84C_QWAo5C9PUzfLY/edit#gid=0



Question2

Number of thread	execution time(in microsec)	speedup	parallelization fraction(f)
1	14	1	0
2	7	2	1
4	5	2.8	0.8571428571
8	5	2.8	0.7346938776
16	4	3.5	0.7619047619
32	4	3.5	0.7373271889
64	4	3.5	0.7256235828
128	6	2.33333333	0.575928009
256	5	2.8	0.6453781513
500	5	2.8	0.6441454337





Calculation of parallelization fraction

T(1) = 14 microsecond
Here , for P = 16 the execution time is minimum
T(P) = 4 microsecond

Speedup = $\frac{T(1)}{T(P)}$ = $\frac{14}{4}$ = 3.5

From Amdahl's Law,

Speedup = $\frac{1}{(f/P) + (1-f)}$ Where , f = Parallelization factor P = Thread Number

So, f = $\frac{(1-T(P)/T(1))}{(1-(1/P))}$

Therefore, f = 0.7619047619 which means that approx. 760.4939913558% of the program is parallelizable.