Desfasurarea lucrarii CUDA

1. Se explica arhitectura CUDA, programul VectorAdd.cu.
2. Partea experimentala:

- Se lanseaza programul VectorAdd din C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.2\0_Simple\vectorAdd - saxpy\vectorAdd

  In acest program se pot modifica:

  #define ALFA 2.0 – parametrul alfa din SAXPY

  #define NR_ELEM 100000 – numarul de elemente de prelucrat
  #define THREAD_PER_BLOCK 256 – numarul de thread-uri pe bloc

  Daca se doreste executia Vector_ADD se va scrie:

  //#define VECT_ADD
  #define VECT_SAXPY

  Daca se doreste executia Vector_SAXPY se va scrie:

  #define VECT_ADD
  //#define VECT_SAXPY

- Se executa programul pentru diferite valori ale NR_ELEM, THREAD_PER_BLOCK si pentru cele doua cazuri (Vector_ADD si Vector SAXPY) csi se completeaza tabelul:

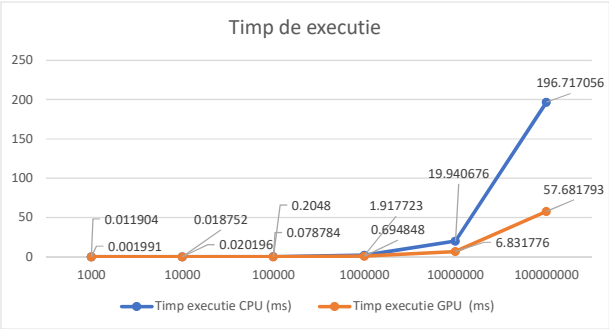
Crestere intre iteratii	NR_ELEM	Timp executie CPU (ms)	Timp executie GPU (ms)	raport crestere timp executie CPU	raport crestere timp executie GPU	raport timp GPU/ timp CPU	Rata de transfer CPU GPU (Mbs)	Rata de transfer CPU GPU (Mbs)
	1000							
10	10000							
10	100000							
10	1000000							
10	10000000							
10	100000000							

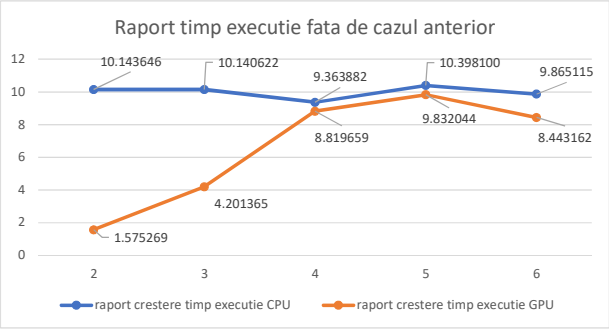
- Se determina numarul maxim de elemente (placa CUDA are 2Go memorie)
- Se interpreteaza (explica) rezultatele

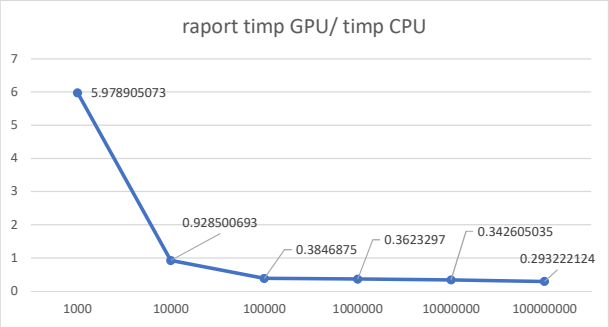
SAXPY

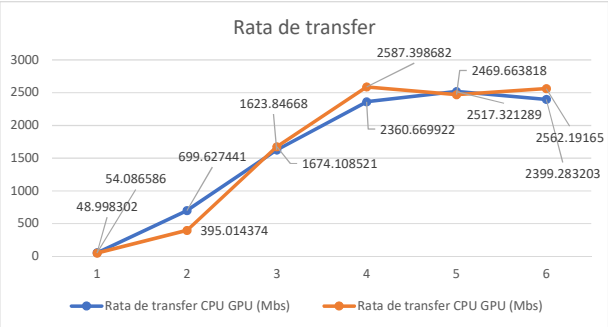
NR THREAD BLOCK **256**

Crestere intre iteratii	NR_ELEM	Timp executie CPU (ms)	Timp executie GPU (ms)	raport crestere timp executie CPU	raport crestere timp executie GPU	raport timp GPU/ timp CPU	Rata de transfer CPU GPU (Mbs)	Rata de transfer CPU GPU (Mbs)
	1000	0.001991	0.011904			5.978905073	54.086586	48.9983
10	10000	0.020196	0.018752	10.143646	1.575269	0.928500693	699.627441	395.0144
10	100000	0.2048	0.078784	10.140622	4.201365	0.3846875	1623.84668	1674.109
10	1000000	1.917723	0.694848	9.363882	8.819659	0.3623297	2360.669922	2587.399
10	10000000	19.940676	6.831776	10.398100	9.832044	0.342605035	2517.321289	2469.664
10	100000000	196.717056	57.681793	9.865115	8.443162	0.293222124	2399.283203	2562.192









OBSERVATII

numarul de blocuri trebuie sa fie cit mai mare
numarul de thread-uri pe bloc sa fie cit mai mare
gradul de ocupare a CUDA sa fie cit mai mare
se executa in paralel 32 threaduri (grup numit warp)
se face acces concurent la memorie (in timpul executiei altor thread-uri)
thread-urile dintr-un bloc se planifica pentru executie paralela (in acelasi grup de 32 de threaduri)

daca numarul de elemente de prelucrat este mic (nu rezulta un numar mare de blocuri si un numar mare de thred-uri pe bloc)
atunci executia in CUDA poate fi mai lenta decit executia in CPU deoarece nu se incarca la maxim toate resursele CUDA
lansarea kernel-ului dureaza foarte putin si e aproximativ constanta ca timp de executie
timpul de executie in CUDA se masoara dupa terminarea tuturor thread-urilor