	t. read		t. CUDA		t. write		t. total	
n	**in-place**	**rezultat**	**in-place**	**rezultat**	**in-place**	**rezultat**	**in-place**	**rezultat**
10	0.062ms	0.046ms	6.004ms	0.215ms	0.190ms	0.156ms	6.256ms	0.418ms
100	0.579ms	0.638ms	6.011ms	0.244ms	0.985ms	0.961ms	7.577ms	1.844ms
1000	62.417ms	60.048ms	8.615ms	3.428ms	78.090ms	66.809ms	149.123ms	130.28ms
10000	7007.530ms	6603.817ms	479.460ms	323.953ms	7366.452ms	8666.08ms	14853.440ms	15593.860ms
10k (bs=32)		7287.02ms		289.719ms		6551.77ms		14128.5ms
10k(bs=64)		7300.64ms		291.28ms		6671.61ms		14263.5ms
10k(bs=128)		5207.65ms		287.461ms		9223.87ms		14719ms
10k(bs=256)		6599.92ms		286.181ms		7090.87ms		13977ms
10k(bs=512)		7524.22ms		293.04ms		6599.9ms		14416.9ms
10k(bs=1024)		5336.25ms		294.75ms		9380ms		15011ms

bs=block size

Concluzii

- Cel mai mult timp se pierde la operatiile de scriere si citire
- Varianta cu matrice rezultat este mai rapida decat variant in-place, din cauza ca nu mai exista overheadul de la sincronizarea core-urilor
- Varianta CUDA e mai rapida decat varianta CPU de la lab 2
- Pe variant cu rezultat, se observa dependenta aprox. liniara intre n*n si timpul de executie (dureaza de 100x mai mult sa procesam n=10000 fata de n=1000). Nu este valabil pe variant in-place