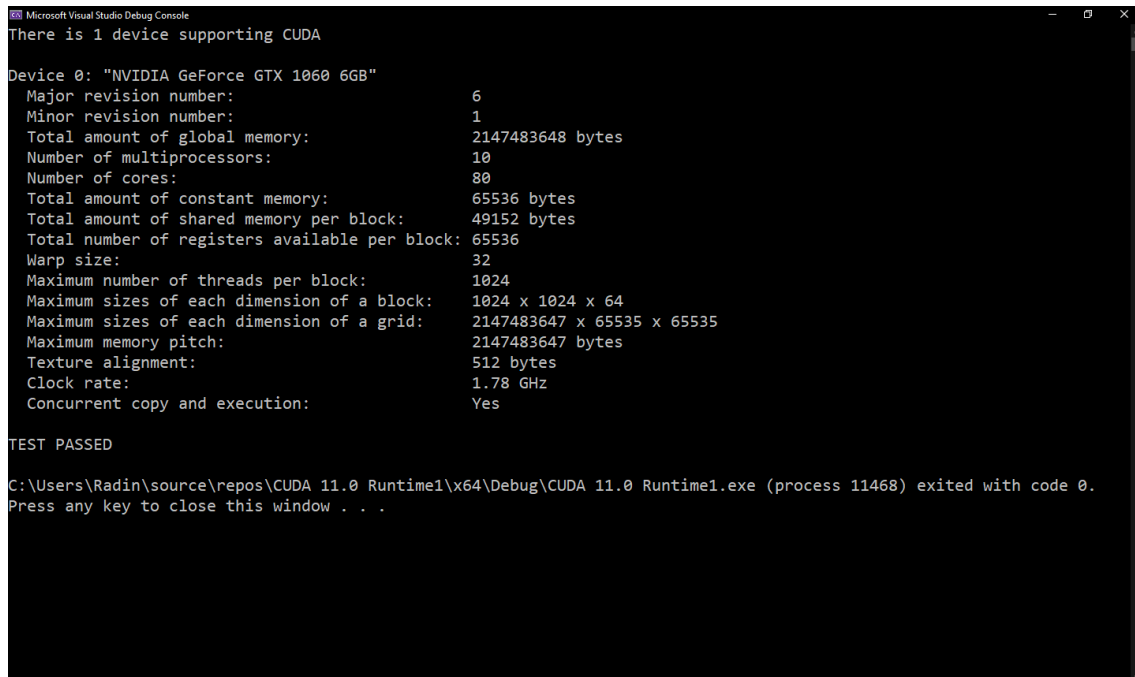


## ۱ گام اول

کد deviceQuery.cu داده شده را کامپایل و اجرا می‌کنیم. مشخصات دستگاه را در شکل ۱ مشاهده می‌کنیم.



```
Microsoft Visual Studio Debug Console
There is 1 device supporting CUDA

Device 0: "NVIDIA GeForce GTX 1060 6GB"
Major revision number:      6
Minor revision number:      1
Total amount of global memory: 2147483648 bytes
Number of multiprocessors:  10
Number of cores:            80
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 49152 bytes
Total number of registers available per block: 65536
Warp size:                  32
Maximum number of threads per block: 1024
Maximum sizes of each dimension of a block: 1024 x 1024 x 64
Maximum sizes of each dimension of a grid: 2147483647 x 65535 x 65535
Maximum memory pitch:       2147483647 bytes
Texture alignment:          512 bytes
Clock rate:                 1.78 GHz
Concurrent copy and execution: Yes

TEST PASSED

C:\Users\Radin\source\repos\CUDA 11.0 Runtime1\x64\Debug\CUDA 11.0 Runtime1.exe (process 11468) exited with code 0.
Press any key to close this window . . .
```

شکل ۱: مشخصات دستگاه با استفاده از کد deviceQuery

## ۲ گام دوم

زمان‌های اجرا میانگین ۱۰ بار اجرا است. همچنین زمان پر و کپی کردن بردارها و آزاد کردن حافظه در نظر گرفته نشده است.

برنامه جمع دو بردار را در حالت سریال بر روی CPU اجرا می‌کنیم. زمان اجرا در حالت سریال به علت کوچک بودن برابر صفر گزارش می‌شود.

با اضافه کردن تابع addWithCuda و بردن محاسبات بر روی GPU زمان اجرا به ۰/۰۰۰۰۴۸ ثانیه می‌رسد. دلیل کاهش سرعت اجرا، بالا بودن سربارهای بردن محاسبات روی GPU نسبت به اندازه مسئله است.

## ۳ گام سوم

به کد قسمت قبل متغیرهای NUM\_THREADS، ELEMENTS\_PER\_THREAD و NUM\_BLOCKS را اضافه می‌کنیم و تغییرات لازم را در کرنل انجام می‌دهیم. زمان‌های اجرای آزمایش شده در جدول ۱ آمده است (تسریع با میانگین‌گیری تسریع دو ستون انتهایی محاسبه شده است).

جدول ۱: زمان‌های اجرا (ثانیه) به ازای اندازه ورودی‌های مختلف

تسریع	اندازه ورودی			موازی‌سازی
	$2^{28}$	$2^{27}$	$2^{26}$	
–	۰/۲۸۱۴۲۶	۰/۱۴۱۷۹۰	۰/۰۷۱۸۰۲	سریال
۰/۳۶	۱/۰۳۸۵۷۰	۰/۳۰۲۵۶۹	۰/۱۵۳۹۵۹	پردازش $n$ المان توسط هر نخ
۱۳/۰۰	۰/۰۲۱۴۷۱	۰/۰۱۰۹۹۰	۰/۰۰۵۵۶۱	پردازش با $n$ بلوک

در حالتی که هر نخ یک المان را پردازش می‌کند متغیر ELEMENTS\_PER\_THREAD برابر یک قرار داده شده است. در حالت پردازش  $n$  المان توسط هر نخ، مقدار این متغیر به شکل زیر محاسبه شده است.

```
int ELEMENTS_PER_THREAD = size / 1024;
```

در شکل‌های ۲ و ۳ به ترتیب خروجی برنامه در حالت پردازش  $n$  المان توسط هر نخ و پردازش یک المان توسط  $n$  بلوک برای اندازه ورودی  $2^{28}$  آمده است.

همانطور که می‌بینیم در حالت اول تنها یک بلوک نخ ۱۰۲۴ تایی داریم و هر نخ ۲۶۲۱۴۴ المان را پردازش می‌کند. این حالت به دلیل شباهت نحوه محاسبه به محاسبات روی CPU، روی هسته‌های کوچک و ضعیف GPU به خوبی جواب نمی‌دهد و اجرای آن بسیار کندتر است.

در حالت دوم هر نخ تنها یک المان را پردازش می‌کند اما تعداد بلوک‌های ۱۰۲۴ تایی ۲۶۲۱۴۴ است. این کار (دادن کارهای کوچک به هر نخ و زیاد کردن تعداد نخ‌ها با افزایش تعداد بلوک‌ها) باعث استفاده حداکثری از قدرت GPU می‌شود و زمان اجرا به شدت کاهش می‌یابد.

```
Microsoft Visual Studio Debug Console
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.037100 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.040595 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.038132 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.037942 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.037367 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.039149 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.038575 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.039040 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.038712 Secs
elements per thread: 262144, threads per blocks: 1024, blocks: 1
[-] Time Elapsed: 1.039093 Secs

[-] The average running time was: 1.038570

C:\Users\Radin\source\repos\CUDA 11.0 Runtime1\x64\Release\CUDA 11.0 Runtime1.exe (process 11236) exited with code 0.
Press any key to close this window . . .
```

شکل ۲: پردازش  $n$  المان توسط هر نخ (درشت دانگی) منجر به کاهش شدید سرعت روی GPU می‌شود.

```
Microsoft Visual Studio Debug Console
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021675 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021613 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021651 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021338 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021634 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021354 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021325 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021391 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021387 Secs
elements per thread: 1, threads per blocks: 1024, blocks: 262144
[-] Time Elapsed: 0.021345 Secs

[-] The average running time was: 0.021471

C:\Users\Radin\source\repos\CUDA 11.0 Runtime1\x64\Release\CUDA 11.0 Runtime1.exe (process 21024) exited with code 0.
Press any key to close this window . . .
```

شکل ۳: پردازش یک المان توسط هر نخ (ریز دانگی) منجر به کاهش شدید زمان اجرا و تسريع مناسب روی GPU می‌شود.

## ۴ گام چهارم

از آنجا که امکان استفاده از printf در کرنل وجود ندارد، هر نخ موارد خواسته شده را داخل ۳ آرایه سراسری می نویسد. سپس در سمت host این ۳ آرایه خوانده و چاپ می شوند. نمونه اجرای کد در شکل ۴ قابل مشاهده است (کد این بخش در فایل whoAmI.cu قرار دارد).

```
Microsoft Visual Studio Debug Console
Calculated Thread: 0, Block: 0, Warp 0, Thread 0
Calculated Thread: 1, Block: 0, Warp 0, Thread 1
Calculated Thread: 2, Block: 0, Warp 0, Thread 2
Calculated Thread: 3, Block: 0, Warp 0, Thread 3
Calculated Thread: 4, Block: 0, Warp 0, Thread 4
Calculated Thread: 5, Block: 0, Warp 0, Thread 5
Calculated Thread: 6, Block: 0, Warp 0, Thread 6
Calculated Thread: 7, Block: 0, Warp 0, Thread 7
Calculated Thread: 8, Block: 0, Warp 0, Thread 8
Calculated Thread: 9, Block: 0, Warp 0, Thread 9
Calculated Thread: 10, Block: 0, Warp 0, Thread 10
Calculated Thread: 11, Block: 0, Warp 0, Thread 11
Calculated Thread: 12, Block: 0, Warp 0, Thread 12
Calculated Thread: 13, Block: 0, Warp 0, Thread 13
Calculated Thread: 14, Block: 0, Warp 0, Thread 14
Calculated Thread: 15, Block: 0, Warp 0, Thread 15
Calculated Thread: 16, Block: 0, Warp 0, Thread 16
Calculated Thread: 17, Block: 0, Warp 0, Thread 17
Calculated Thread: 18, Block: 0, Warp 0, Thread 18
Calculated Thread: 19, Block: 0, Warp 0, Thread 19
Calculated Thread: 20, Block: 0, Warp 0, Thread 20
Calculated Thread: 21, Block: 0, Warp 0, Thread 21
Calculated Thread: 22, Block: 0, Warp 0, Thread 22
Calculated Thread: 23, Block: 0, Warp 0, Thread 23
Calculated Thread: 24, Block: 0, Warp 0, Thread 24
Calculated Thread: 25, Block: 0, Warp 0, Thread 25
Calculated Thread: 26, Block: 0, Warp 0, Thread 26
Calculated Thread: 27, Block: 0, Warp 0, Thread 27
Calculated Thread: 28, Block: 0, Warp 0, Thread 28
Calculated Thread: 29, Block: 0, Warp 0, Thread 29
Calculated Thread: 30, Block: 0, Warp 0, Thread 30
Calculated Thread: 31, Block: 0, Warp 0, Thread 31
Calculated Thread: 32, Block: 0, Warp 1, Thread 32
Calculated Thread: 33, Block: 0, Warp 1, Thread 33
Calculated Thread: 34, Block: 0, Warp 1, Thread 34
Calculated Thread: 35, Block: 0, Warp 1, Thread 35
Calculated Thread: 36, Block: 0, Warp 1, Thread 36
Calculated Thread: 37, Block: 0, Warp 1, Thread 37
Calculated Thread: 38, Block: 0, Warp 1, Thread 38
Calculated Thread: 39, Block: 0, Warp 1, Thread 39
Calculated Thread: 40, Block: 0, Warp 1, Thread 40
Calculated Thread: 41, Block: 0, Warp 1, Thread 41
Calculated Thread: 42, Block: 0, Warp 1, Thread 42
Calculated Thread: 43, Block: 0, Warp 1, Thread 43
Calculated Thread: 44, Block: 0, Warp 1, Thread 44
Calculated Thread: 45, Block: 0, Warp 1, Thread 45
Calculated Thread: 46, Block: 0, Warp 1, Thread 46
Calculated Thread: 47, Block: 0, Warp 1, Thread 47
Calculated Thread: 48, Block: 0, Warp 1, Thread 48
Calculated Thread: 49, Block: 0, Warp 1, Thread 49
Calculated Thread: 50, Block: 0, Warp 1, Thread 50
Calculated Thread: 51, Block: 0, Warp 1, Thread 51
Calculated Thread: 52, Block: 0, Warp 1, Thread 52
Calculated Thread: 53, Block: 0, Warp 1, Thread 53
Calculated Thread: 54, Block: 0, Warp 1, Thread 54
Calculated Thread: 55, Block: 0, Warp 1, Thread 55
Calculated Thread: 56, Block: 0, Warp 1, Thread 56
Calculated Thread: 57, Block: 0, Warp 1, Thread 57
Calculated Thread: 58, Block: 0, Warp 1, Thread 58
Calculated Thread: 59, Block: 0, Warp 1, Thread 59
Calculated Thread: 60, Block: 0, Warp 1, Thread 60
Calculated Thread: 61, Block: 0, Warp 1, Thread 61
Calculated Thread: 62, Block: 0, Warp 1, Thread 62
Calculated Thread: 63, Block: 0, Warp 1, Thread 63
Calculated Thread: 64, Block: 1, Warp 0, Thread 0
Calculated Thread: 65, Block: 1, Warp 0, Thread 1
Calculated Thread: 66, Block: 1, Warp 0, Thread 2
Calculated Thread: 67, Block: 1, Warp 0, Thread 3
```

شکل ۴: نحوه دسته بندی نخها داخل warp و block