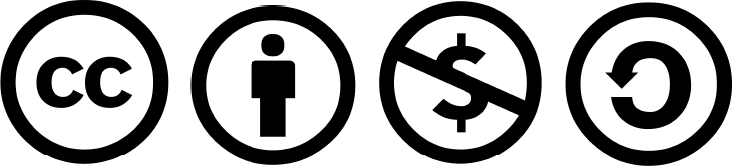
Report Contest-CUDA

***Radix Sort***



Course: High Performance Computing 2021/2022 Lecturer: Francesco Moscato - [fmoscato@unisa.it](mailto:fmoscato@unisa.it)

Group:

Lamberti Martina 0622701476 [m.lamberti61@studenti.unisa.it](mailto:m-lamberti61@studenti.unisa.it) Salvati Vincenzo 0622701550 [v.salvati10@studenti.unisa.it](mailto:v.salvati10@studenti.unisa.it) Silvitelli Daniele 0622701504 [d.silvitelli@studenti.unisa.it](mailto:d.silvitelli@studenti.unisa.it) Sorrentino Alex 0622701480 [a.sorrentino120@studenti.unisa.it](mailto:a.sorrentino120@studenti.unisa.it)

Summary

[Problem Description 3](#_Toc93833315)

[Experimental setup 3](#_Toc93833316)

[Hardware 3](#_Toc93833317)

[GPU 3](#_Toc93833318)

[CPU 4](#_Toc93833319)

[RAM 13](#_Toc93833320)

[Software 14](#_Toc93833321)

[Performance 15](#_Toc93833322)

[Global Memory 17](#_Toc93833323)

[Shared Memory 18](#_Toc93833324)

[Texture Memory 19](#_Toc93833325)

[Considerations 20](#_Toc93833326)

[Case Study 21](#_Toc93833327)

[1. CUMULATIVE\_SUM\_PER\_BLOCK 22](#_Toc93833328)

[2. COUNTING ONES PER GRID 23](#_Toc93833329)

[3. SORTING PER BIT 24](#_Toc93833330)

[Differences among different memory allocation 25](#_Toc93833331)

[Test case 26](#_Toc93833332)

[API 27](#_Toc93833333)

[Implemented functions 27](#_Toc93833334)

[Implemented Functions Documentation 29](#_Toc93833335)

[How to run 33](#_Toc93833336)

[Bibliography 34](#_Toc93833337)

[License 34](#_Toc93833338)

# Problem Description

Parallelizing and evaluating performances of "RADIX SORT" Algorithm, by using CUDA with different memory allocation:

1. Global Memory;
2. Shared Memory;
3. Texture Memory.

# Experimental setup

## Hardware

### GPU

The scripts “***info.cu***” and “***bandwidth.cu***” reports information about GPU’s configuration.

Device name

Compute capability

: NVIDIA GeForce RTX 2060 SUPER

: 7.5

Clock Rate Total SMs

Shared Memory Per SM Registers Per SM

Max threads per SM L2 Cache Size

Total Global Memory

Memory Clock Rate

: 1695000 kHz

: 34

: 65536 bytes

: 65536 32-bit

: 1024

: 4194304 bytes

: 8589606912 bytes

: 7001000 kHz

Max threads per block

Max threads in X-dimension of block

Max threads in Y-dimension of block Max threads in Z-dimension of block

: 1024

: 1024

: 1024

: 64

Max blocks in X-dimension of grid Max blocks in Y-dimension of grid

Max blocks in Z-dimension of grid

: 2147483647

: 65535

: 65535

Shared Memory Per Block Registers Per Block

Warp size

: 49152 bytes

: 65536 32-bit

: 32

Pageable transfers (16MB)

Host to Device bandwidth (GB/s)

Device to Host bandwidth (GB/s)

: 6.995637

: 9.796663

Pinned transfers (16MB)

Host to Device bandwidth (GB/s) Device to Host bandwidth (GB/s)

: 12.654791

: 12.685717

The NVIDIA GeForce RTX 2060 SUPER has cc=7.x, so that, the maximum number of manageable blocks for each SM is 16.

### CPU

The command ‘***cat /proc/cpuinfo***’ reports information about core’s configuration.

processor : 0

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 0

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 1

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 0

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor vendor\_id cpu family model model name stepping microcode

: 2

: GenuineIntel

: 6

: 165

: Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz

: 5

: 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 1

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags

: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36

clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 3

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 1

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

|  |  |
| --- | --- |
| processor :  vendor\_id :  cpu family : | 4  GenuineIntel  6 |
| model : | 165 |
| model name : | Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz |
| stepping : | 5 |
| microcode : | 0xffffffff |
| cpu MHz : | 3701.000 |
| cache size : | 256 KB |
| physical id : | 0 |
| siblings : | 20 |
| core id : | 2 |
| cpu cores : | 10 |
| apicid : | 0 |
| initial apicid : | 0 |

fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 5

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 2

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 6

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 3

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes

xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 7

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 3

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 8

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 4

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 9

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 4

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 10

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 5

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor vendor\_id cpu family model model name stepping

: 11

: GenuineIntel

: 6

: 165

: Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz

: 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 5

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags

: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36

clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 12

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 6

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor vendor\_id cpu family model model name stepping microcode cpu MHz cache size

physical id siblings core id

cpu cores apicid

: 13

: GenuineIntel

: 6

: 165

: Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz

: 5

: 0xffffffff

: 3701.000

: 256 KB

: 0

: 20

: 6

: 10

: 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6 wp

flags

: yes

: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36

clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 14

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 7

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 15

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 7

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes

xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 16

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 8

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 17

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 8

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 18

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 9

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

processor : 19

vendor\_id : GenuineIntel

cpu family : 6

model : 165

model name : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz stepping : 5

microcode : 0xffffffff

cpu MHz : 3701.000

cache size : 256 KB

physical id : 0

siblings : 20

core id : 9

cpu cores : 10

apicid : 0

initial apicid : 0 fpu : yes

fpu\_exception : yes cpuid level : 6

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm pni pclmulqdq dtes64 est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4\_1 sse4\_2 movbe popcnt tsc\_deadline\_timer aes xsave osxsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch fsgsbase bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt ibrs ibpb stibp ssbd

bogomips : 7402.00 clflush size : 64 cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual power management:

### RAM

The command ‘***cat /proc/meminfo*’** gives information about total amount of physical RAM, the

one left unused….

|  |  |  |
| --- | --- | --- |
| MemTotal: | 16693720 | kB |
| MemFree: | 8456180 | kB |
| Buffers: | 34032 | kB |
| Cached: | 188576 | kB |
| SwapCached: | 0 | kB |
| Active: | 167556 | kB |
| Inactive: | 157876 | kB |
| Active(anon): | 103104 | kB |
| Inactive(anon): | 17440 | kB |
| Active(file): | 64452 | kB |
| Inactive(file): | 140436 | kB |
| Unevictable: | 0 | kB |
| Mlocked: | 0 | kB |
| SwapTotal: | 50331648 | kB |
| SwapFree: | 50221984 | kB |
| Dirty: | 0 | kB |
| Writeback: | 0 | kB |
| AnonPages: | 102824 | kB |
| Mapped: | 71404 | kB |
| Shmem: | 17720 | kB |
| Slab: | 13868 | kB |
| SReclaimable: | 6744 | kB |
| SUnreclaim: | 7124 | kB |
| KernelStack: | 2848 | kB |
| PageTables: | 2524 | kB |
| NFS\_Unstable: | 0 | kB |
| Bounce: | 0 | kB |
| WritebackTmp: | 0 | kB |
| CommitLimit: | 515524 | kB |
| Committed\_AS: | 3450064 | kB |
| VmallocTotal: | 122880 | kB |
| VmallocUsed: | 21296 | kB |
| VmallocChunk: | 66044 | kB |
| HardwareCorrupted: 0 kB  AnonHugePages: 2048 kB  HugePages\_Total: 0  HugePages\_Free: 0  HugePages\_Rsvd: 0  HugePages\_Surp: 0  Hugepagesize: 2048 kB  DirectMap4k: 12280 kB  DirectMap4M: 897024 kB | | |

### Software

* SO: Windows 11.
* The command “***nvcc –-version***” reports CUDA libraries.

nvcc: NVIDIA (R) Cuda compiler driver Copyright (c) 2005-2020 NVIDIA Corporation

Built on Mon\_Nov\_30\_19:15:10\_Pacific\_Standard\_Time\_2020 Cuda compilation tools, release 11.2, V11.2.67

Build cuda\_11.2.r11.2/compiler.29373293\_0

* The command “***nvidia-smi***” reports monitoring and management capabilities.

+ +

| NVIDIA-SMI 511.23 Driver Version: 511.23 CUDA Version: 11.6 |

| + + +

| GPU Name TCC/WDDM | Bus-Id

| Fan Temp Perf Pwr:Usage/Cap|

Disp.A | Volatile Uncorr. ECC |

Memory-Usage | GPU-Util Compute M. |

| | | MIG M. |

|===============================+======================+======================|

| 0 NVIDIA GeForce ... WDDM | 00000000:01:00.0 On |

| 0% 59C P0 41W / 175W |

| |

1120MiB / 8192MiB |

|

5%

N/A | Default |

N/A |

+ + + +

# Performance

In our implementation, the array to be sorted is read from the file "*random\_numbers.txt*", because we wanted to generate the random numbers once, so that it is always used the same numbers to make comparisons.

Hence, to generate this file it was implemented a script called “*main.java*” placed in the folder which is called “*Random\_Int\_2000000\_Saving\_File*”.

In particular, our algorithm is able to execute automatically the same code by following parameters:

* size\_of\_array: 5120;
* thread\_per\_block: 64, 128, 256, 512, 1024.

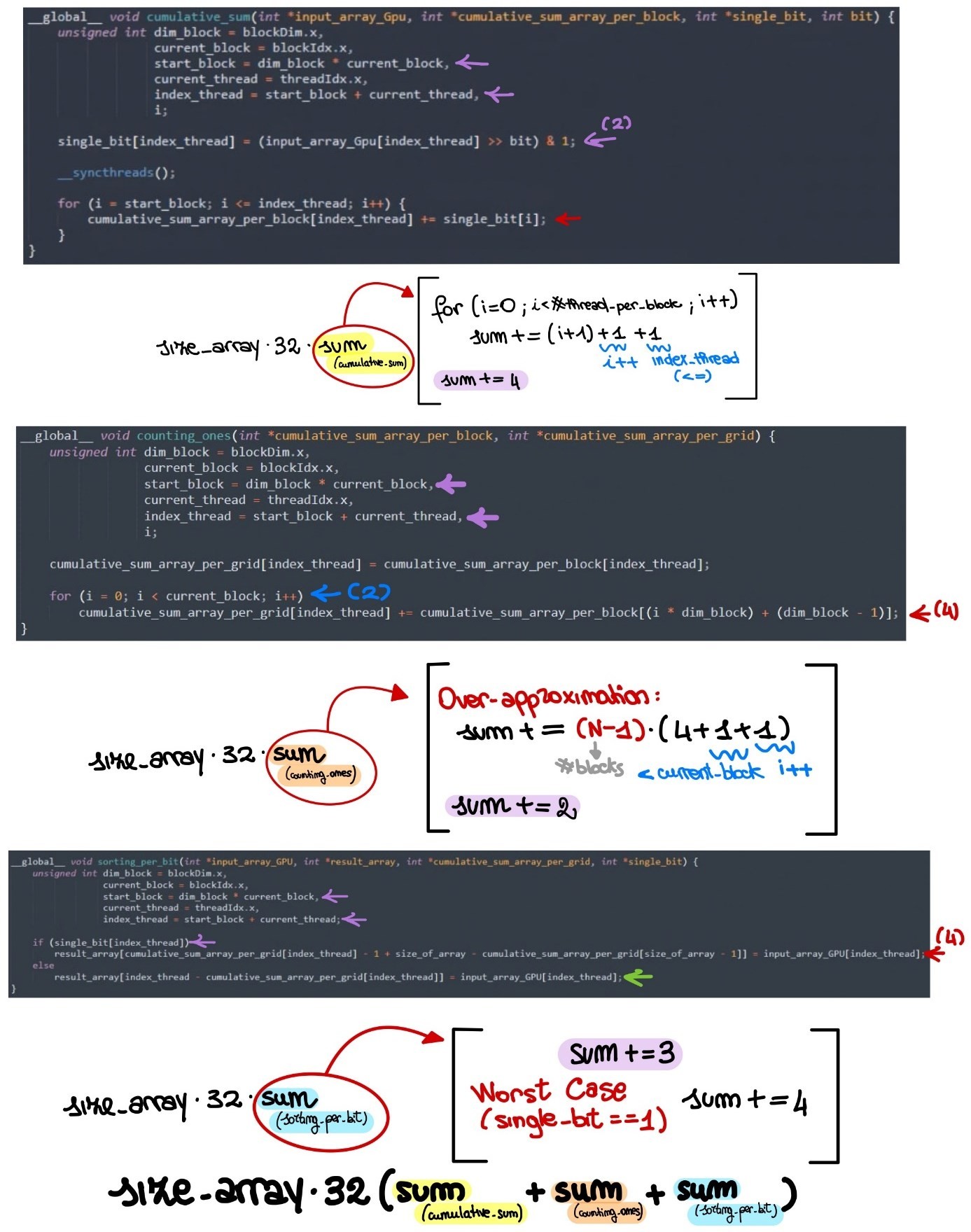
The choice of this threads is not random, since the GPU’s SM is able to deal with max 16 blocks:

* 🡪 16 = 16 🡪 🡪 the occupancy is 100%;
* 🡪 8 = 16 🡪 🡪 the occupancy is 100%;
* 🡪 4 = 16 🡪 🡪 the occupancy is 100%;
* 🡪 2 = 16 🡪 🡪 the occupancy is 100%;
* 🡪 1 = 16 🡪 🡪 the occupancy is 100%;

As far as this program concerned, it takes into account about the elapsed time and the Mflops referred R*adix Sort algorithm* performed by CPU and GPU.

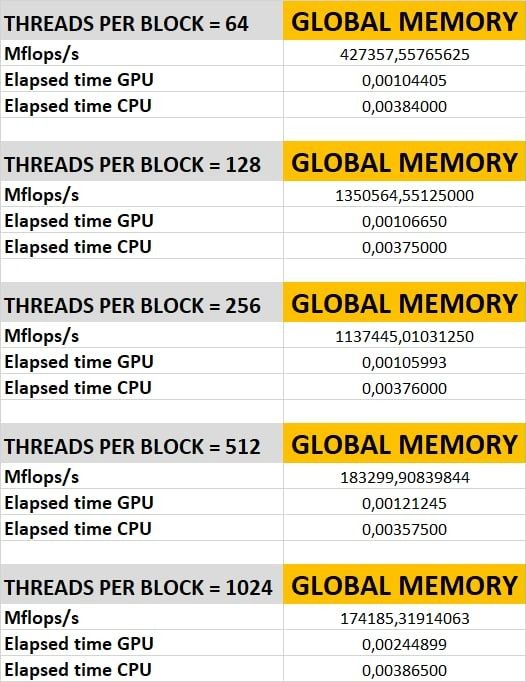
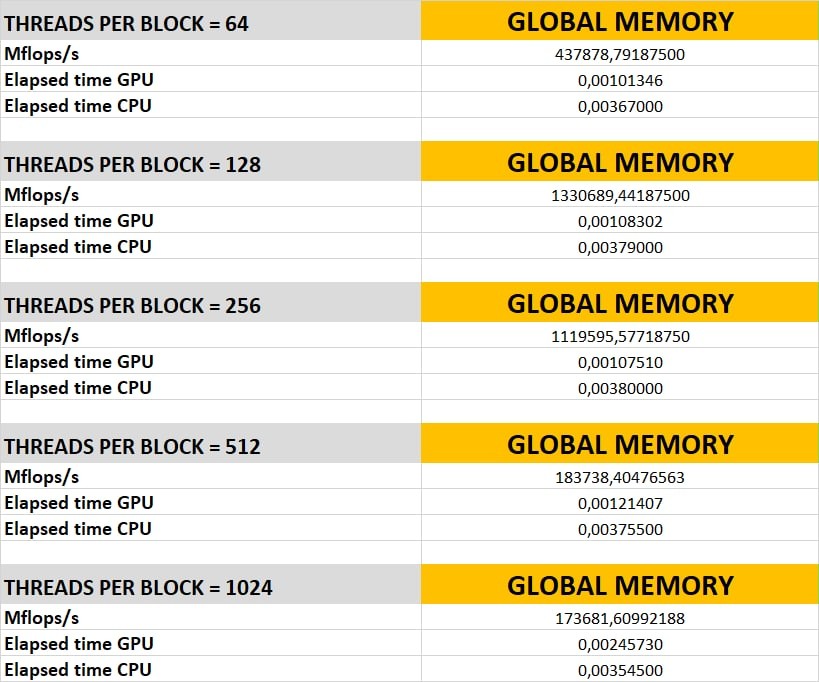
The bash file allows us to perform this automation 200 times in order to calculate a mean of each measure.

To perform the number of flops, it has been considered all ALU operations from each implemented kernel. In addition to that, each number of them has been multiplied for the number of elements (each element is associated to each active thread, since the expected occupancy is always 100%) and for 32 (because of the considered binary digit).

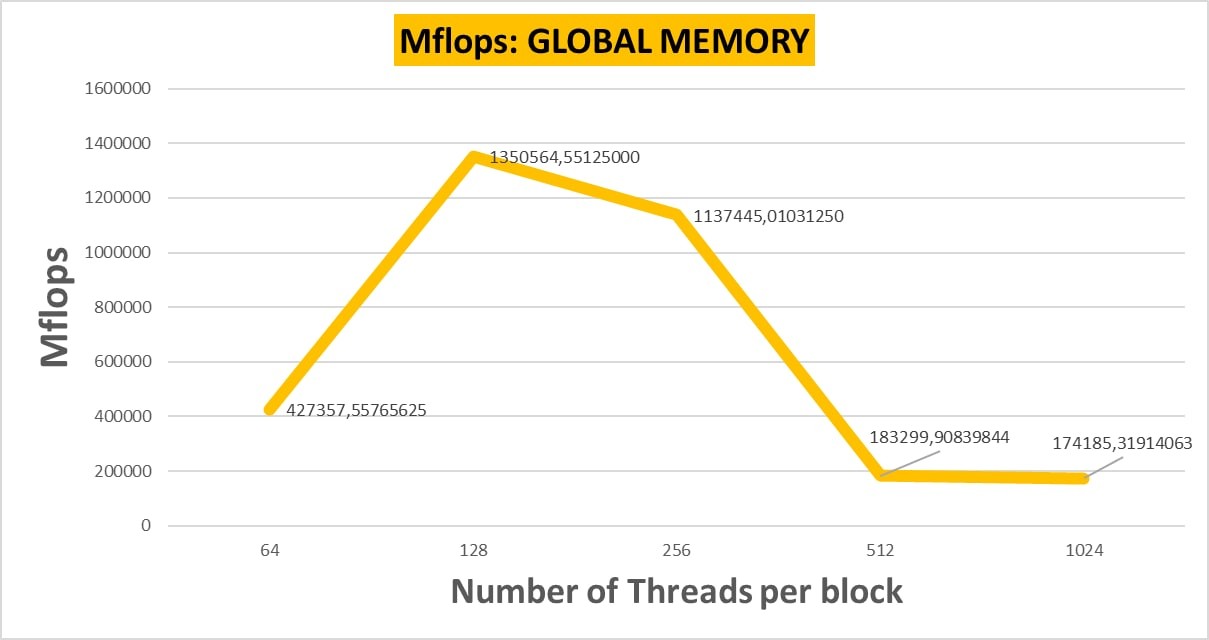


*Figure 1 - counting flops*

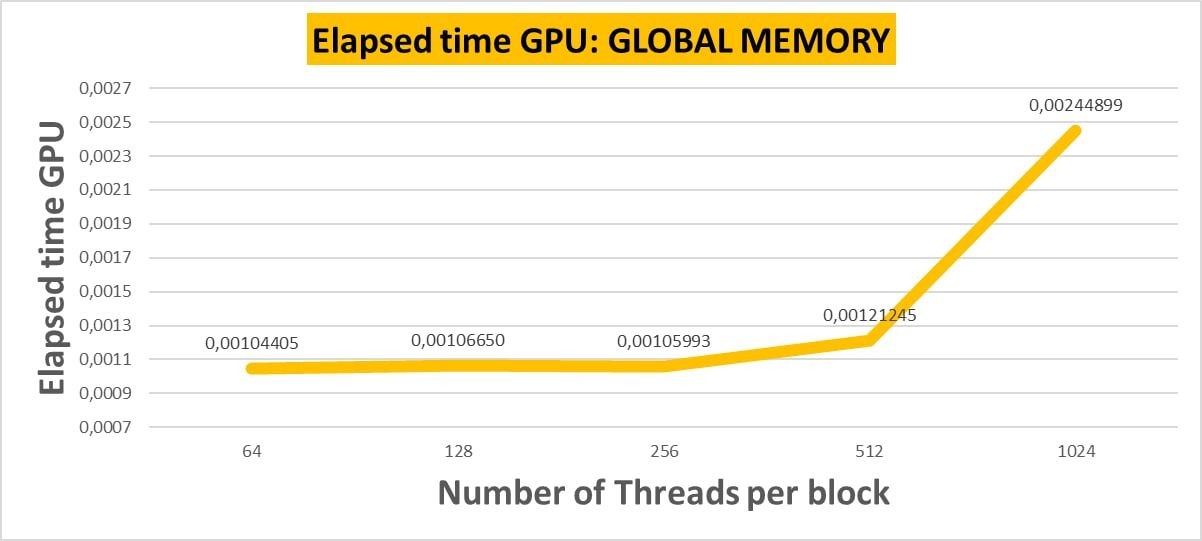
## Global Memory



*Figure 2 - measures of global memory*

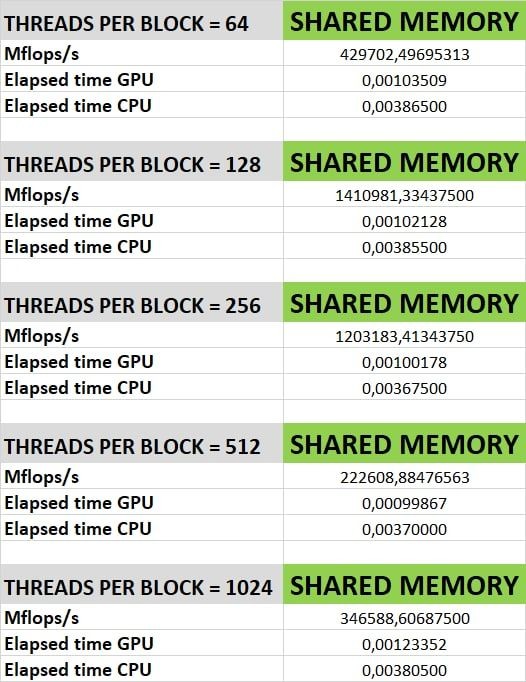
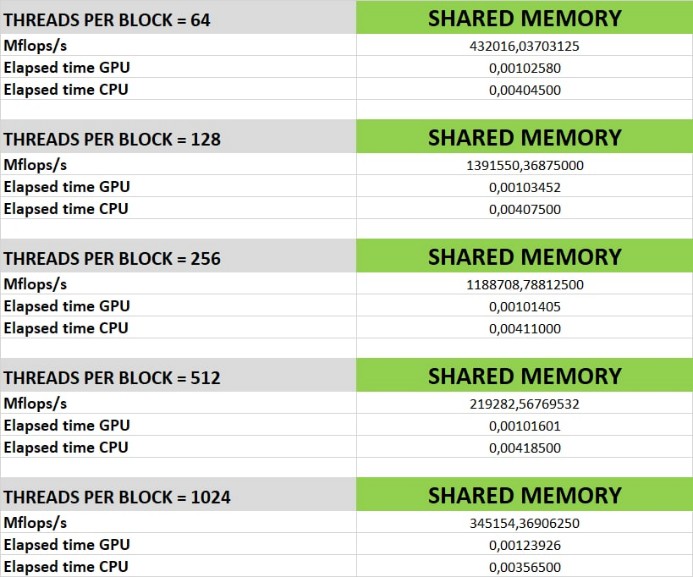


*Figure 3 - Mflops of global memory*

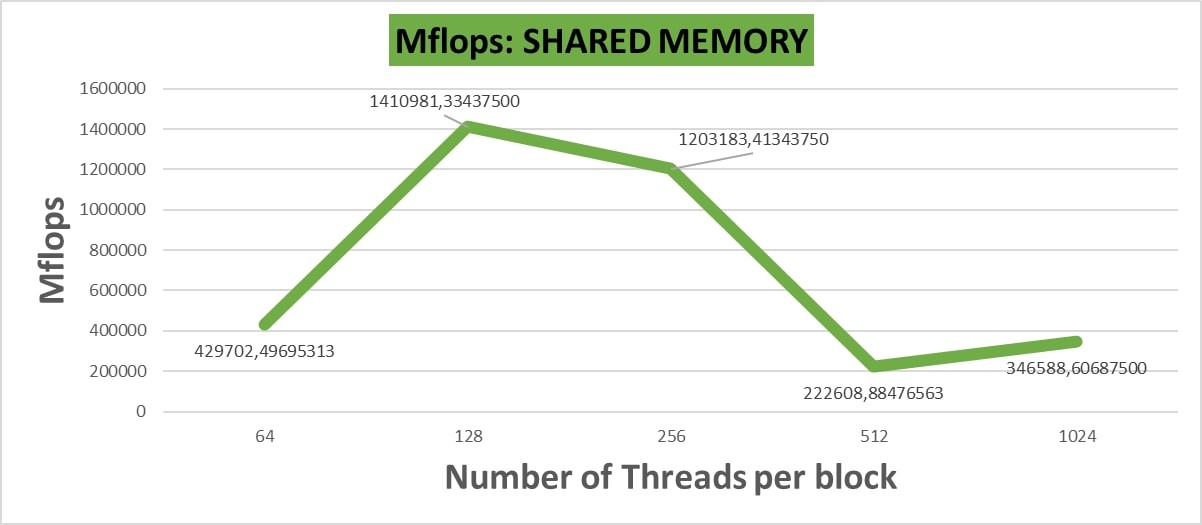


*Figure 4 - Elapsed time GPU of global memory*

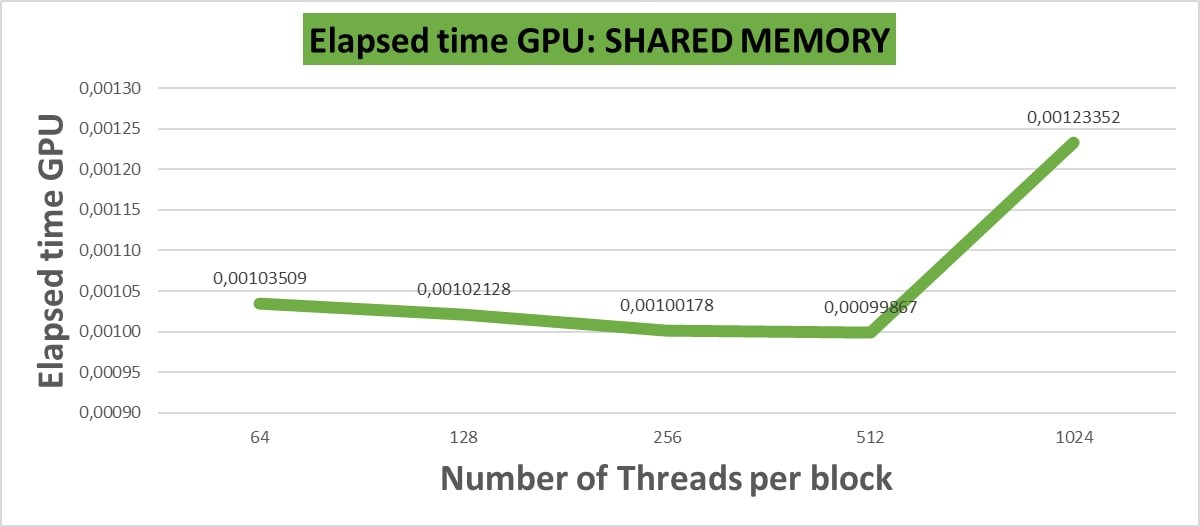
## Shared Memory



*Figure 5 - measures of shared memory*

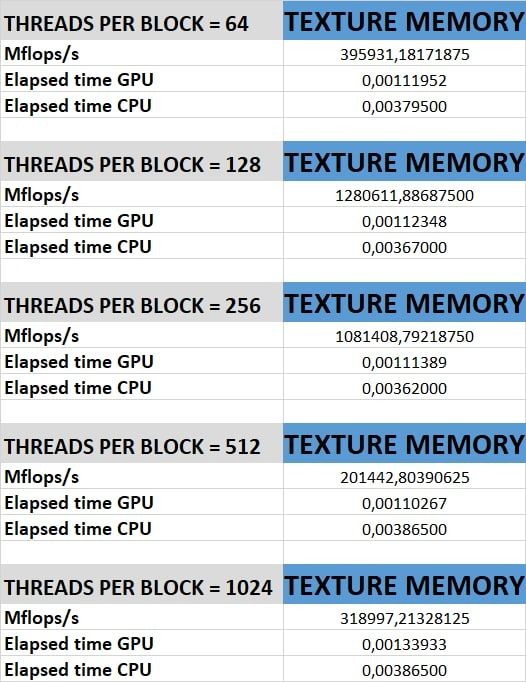
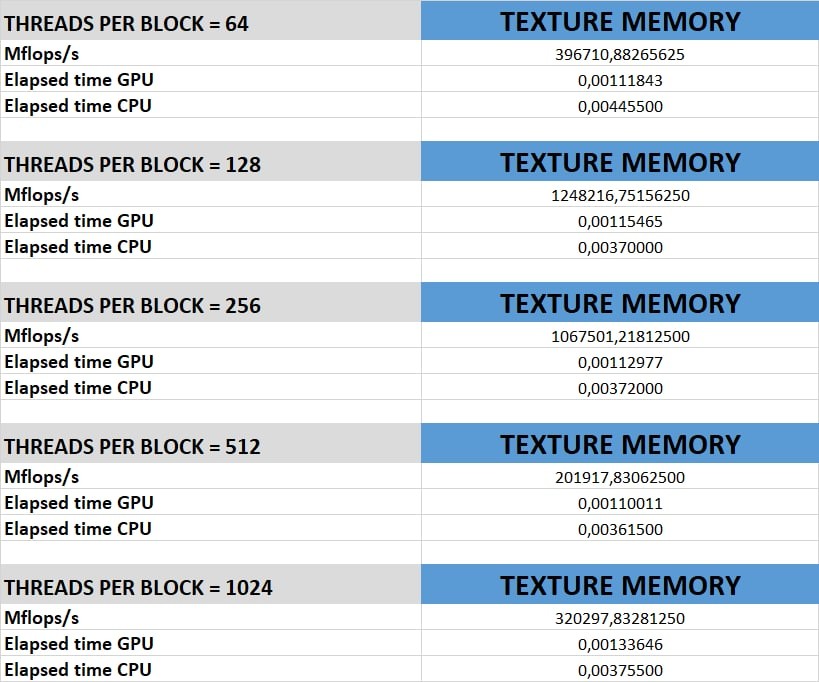


*Figure 6 - Mflops of shared memory*

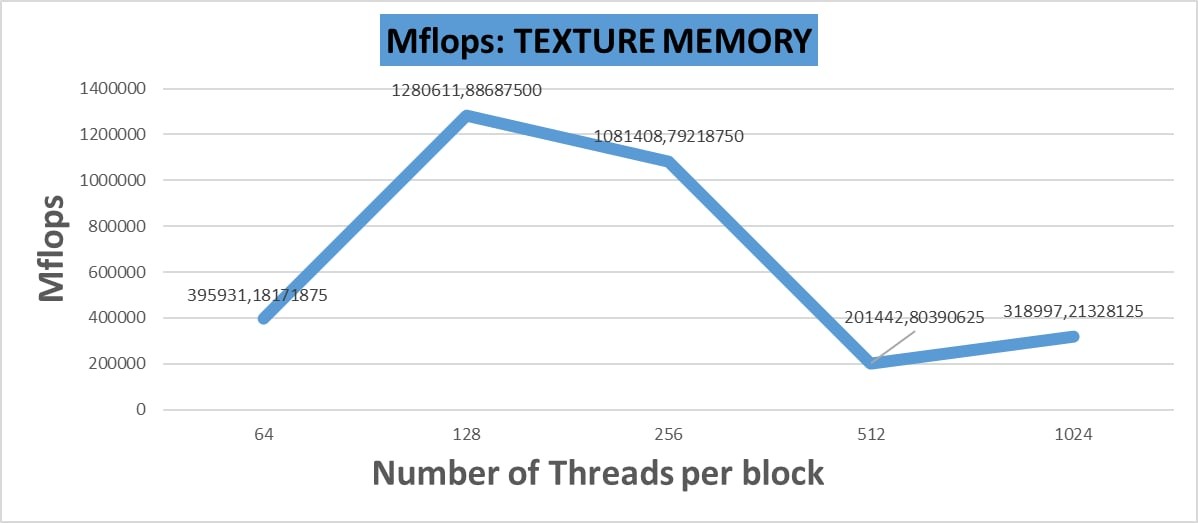


*Figure 7 - Elapsed time GPU of shared memory*

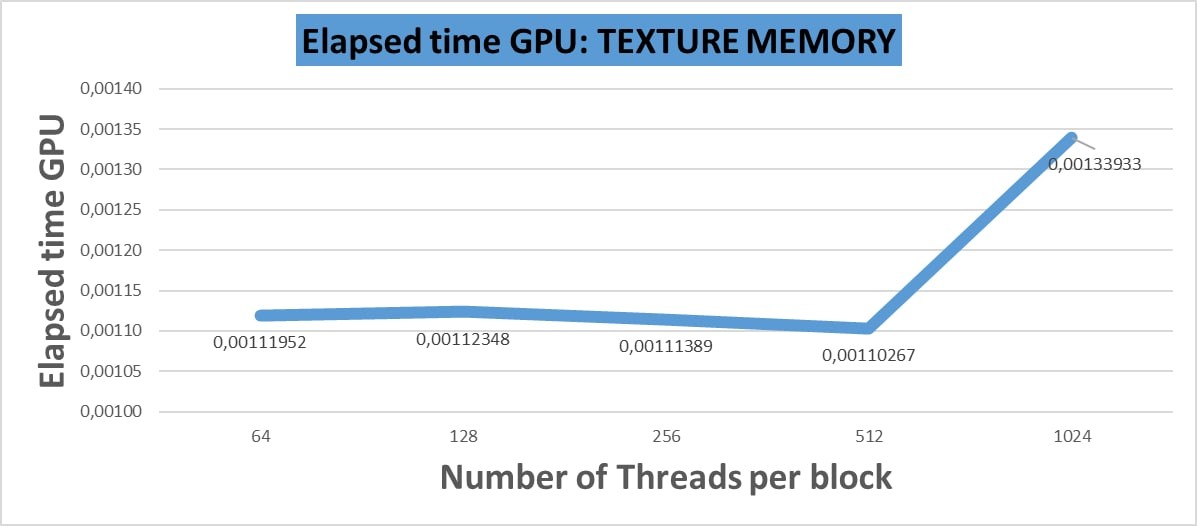
## Texture Memory



*Figure 8 - measures of texture memory*

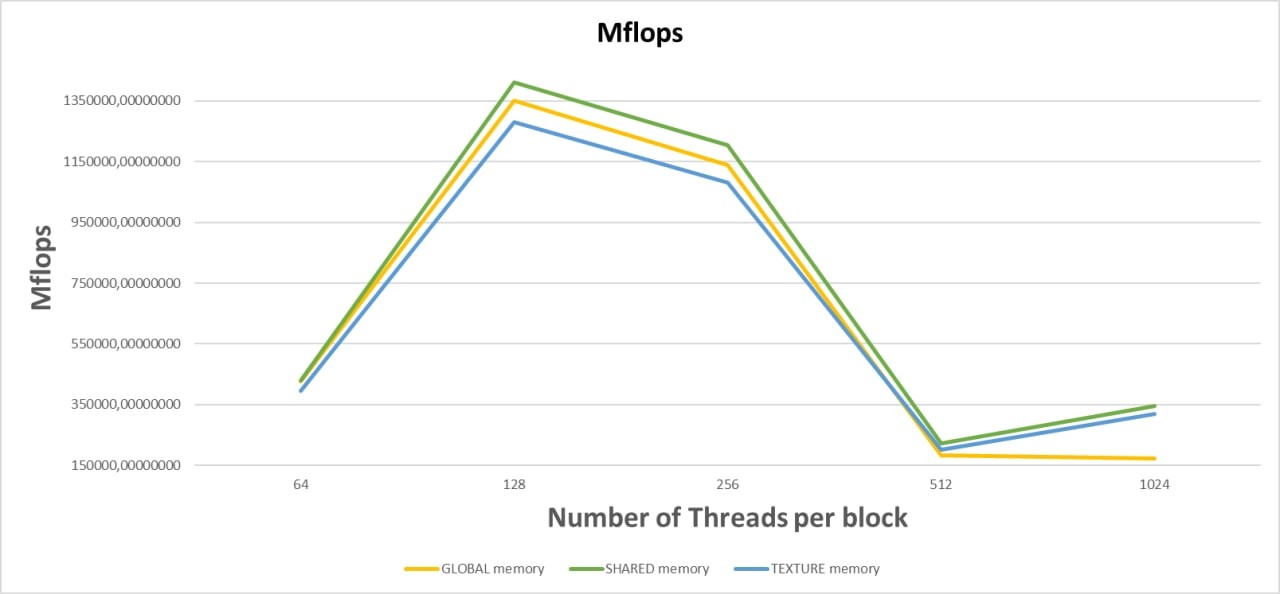


*Figure 9 - Mflops of texture memory*

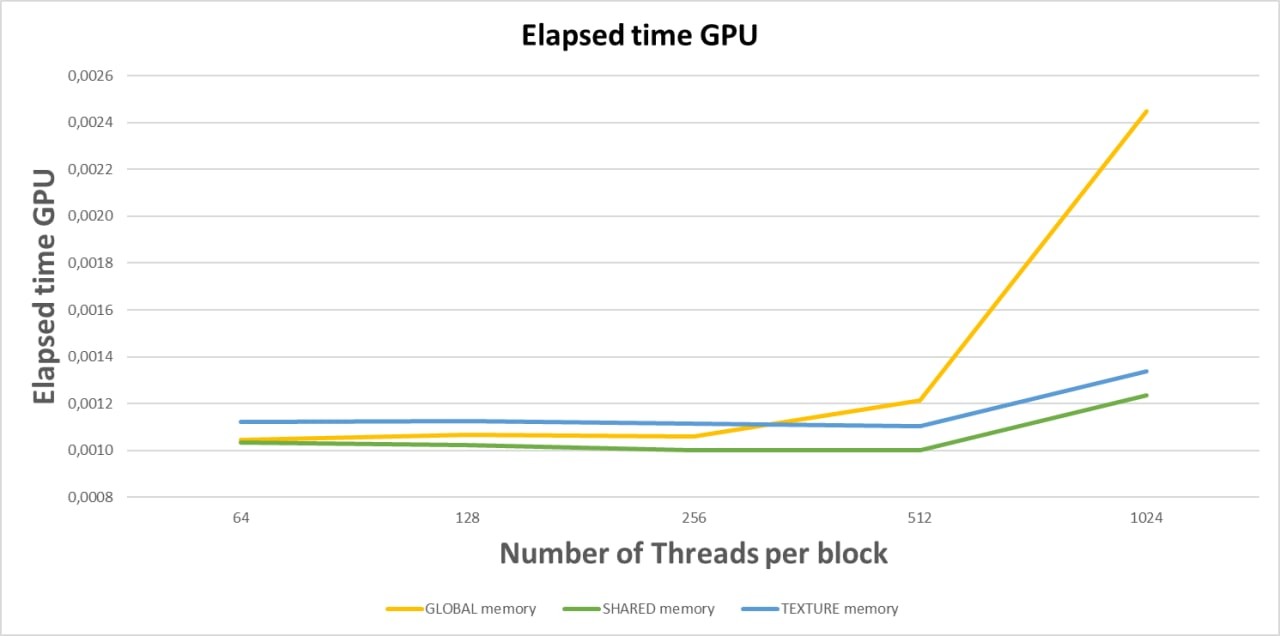


*Figure 10 - Elapsed time GPU of texture memory*

## Considerations



*Figure 11 - Mflops of each memory allocation*



*Figure 12 - Elapsed time GPU of each memory allocation*

The plots take into account about the Mflops and the elapsed time for each different memory allocation. As follow, it will be explained three particular considerations:

* **Global memory has high latency due to technological reasons**: global memory has the worst performance than others, because of the presence of the L2 cache. In fact, every access pass through the cache, so that it could have cache misses;
* **The profitable way of performing computation on the device is to use shared memory**: shared memory has better performance than others, despite a necessary overhead to load subsets of data from global memory. In fact, it has lower latency than others and this compensates for the overhead;
* **Texture memory is a read-only cache ad has high latency due to technological reasons**: texture memory has better performance than global memory for high number of threads, anyway, as global memory, it could have cache misses.

# Case Study

We have produced solutions with the same algorithm, but they differ in the way the memory is allocated.

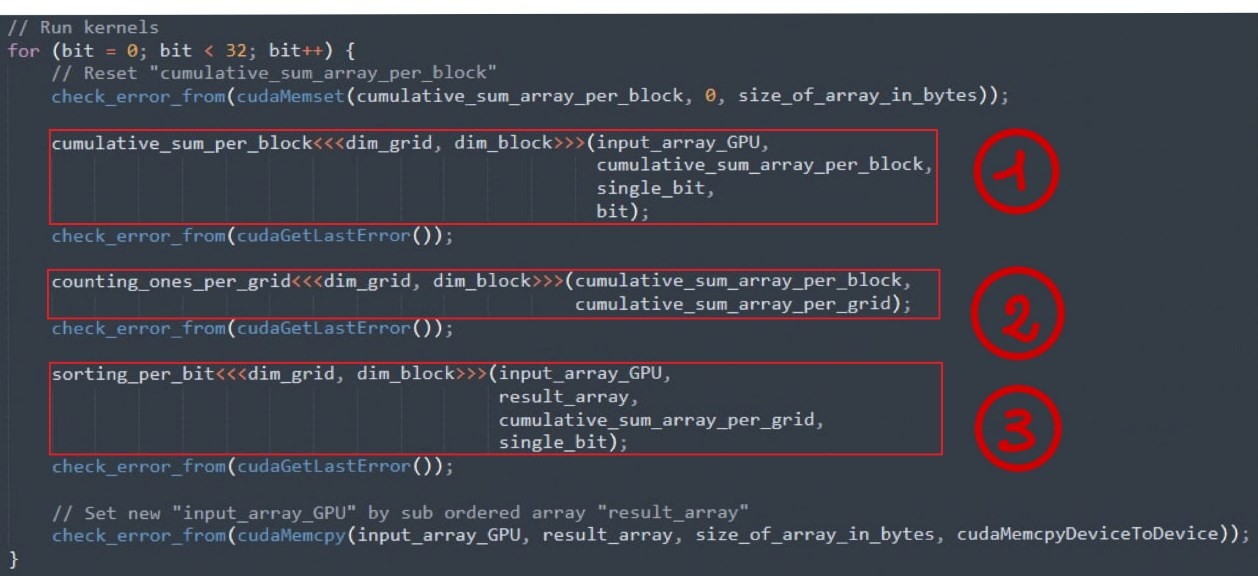
The kernel is divided into blocks of dimension

and each thread carry on specific

element of input array.

So, the implementation operates with unsigned int numbers since the Radix Sort works only with positive numbers, moreover, they are considered in binary numerical system ***(that led us to represent decimal numbers on 32 bits)***. Hence, we iterate over all the bits of each number from the LSB to MSB.

To better use the GPU, three kernels have been used. The reason is related to the synchronization of all threads of the grid and not only the ones in a single block. In this way, the operations within the single kernel will be executed only after the conclusion of previous kernel operations.



*Figure 13 - Kernels*

The iteration of this loop led us to consider a specific bit of the decimal numbers.

## 1. CUMULATIVE\_SUM\_PER\_BLOCK

*Figure 14 - 1° Kernel*

In this kernel it is considered single\_bit array which contains the i-th bit of the

index\_thread-th decimal number stored into input\_array\_GPU.

The i-th bit of this array is obtained by two operations:

1. a shift operation;
2. and logic AND operation.

As each thread need to know the binary elements considered in own block, it is placed *\_\_syncthreads()* function and, at this point, each thread performs the sum of each element until itself, limiting to its own block.

Thus, each performed cumulative sum is stored into cumulative\_sum\_array\_per\_block.

At the end of this kernel, we have filled all cumulative\_sum\_array\_per\_block which can be properly used in the next kernel.

## 2. COUNTING ONES PER GRID

*Figure 15 - 2° Kernel*

Given a thread of a specific block, the goal of this kernel is to compute a cumulative sum obtained considering the cumulative sum per block performed by each last thread of each previous block plus the cumulative sum per block performed by itself in its own block.

## 3. SORTING PER BIT

*Figure 16 - 3° Kernel*

Finally, the last kernel is used to sort the decimal element by using properly the binary digits and the performed cumulative sum. In particular, given a specific decimal number, the considered binary digit:

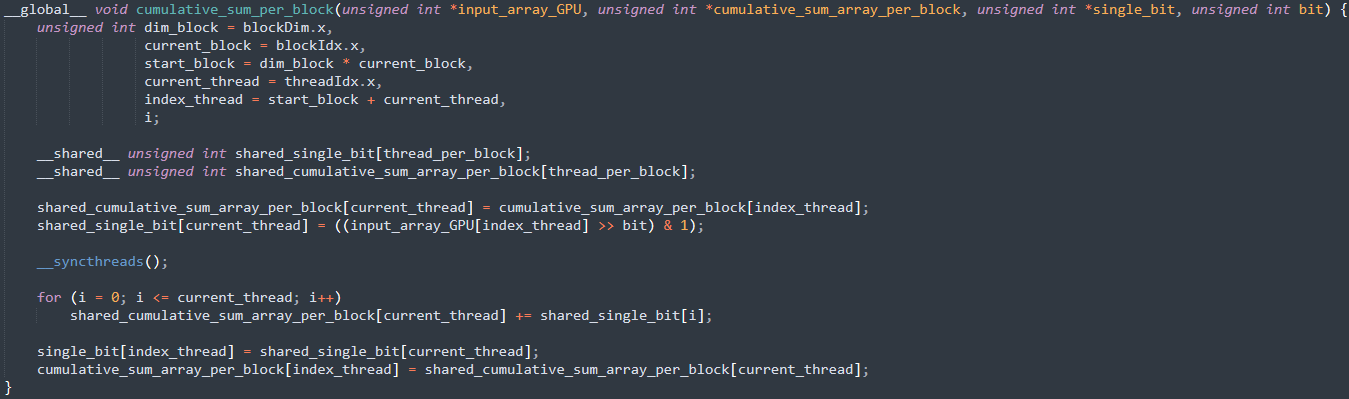
* if it is one, it must be placed in a particular position of result\_array obtained by putting all binary zero digits (𝑠𝑖𝑧𝑒\_𝑜𝑓\_𝑎𝑟𝑟𝑎𝑦 – 𝑐𝑢𝑚𝑢𝑙𝑎𝑡𝑖𝑣𝑒\_𝑠𝑢𝑚\_𝑎𝑟𝑟𝑎𝑦\_𝑝𝑒𝑟\_𝑔𝑟𝑖𝑑[𝑠𝑖𝑧𝑒\_𝑜𝑓\_𝑎𝑟𝑟𝑎𝑦 − 1]) and all previous binary one digits (𝑐𝑢𝑚𝑢𝑙𝑎𝑡𝑖𝑣𝑒\_𝑠𝑢𝑚\_𝑎𝑟𝑟𝑎𝑦\_𝑝𝑒𝑟\_𝑔𝑟𝑖𝑑[𝑖𝑛𝑑𝑒𝑥\_𝑡ℎ𝑟𝑒𝑎𝑑] − 1) before it;
* if it is zero, it must be placed in a particular position of result\_array obtained by subtracting the current position (𝑖𝑛𝑑𝑒𝑥\_𝑡ℎ𝑟𝑒𝑎𝑑) to the number of all previous binary one digits (𝑐𝑢𝑚𝑢𝑙𝑎𝑡𝑖𝑣𝑒\_𝑠𝑢𝑚\_𝑎𝑟𝑟𝑎𝑦\_𝑝𝑒𝑟\_𝑔𝑟𝑖𝑑[𝑖𝑛𝑑𝑒𝑥\_𝑡ℎ𝑟𝑒𝑎𝑑]) before it.

## Differences among different memory allocation

As mentioned above, each version runs the same procedure, however specific arrays, used in the kernels, are allocated differently.

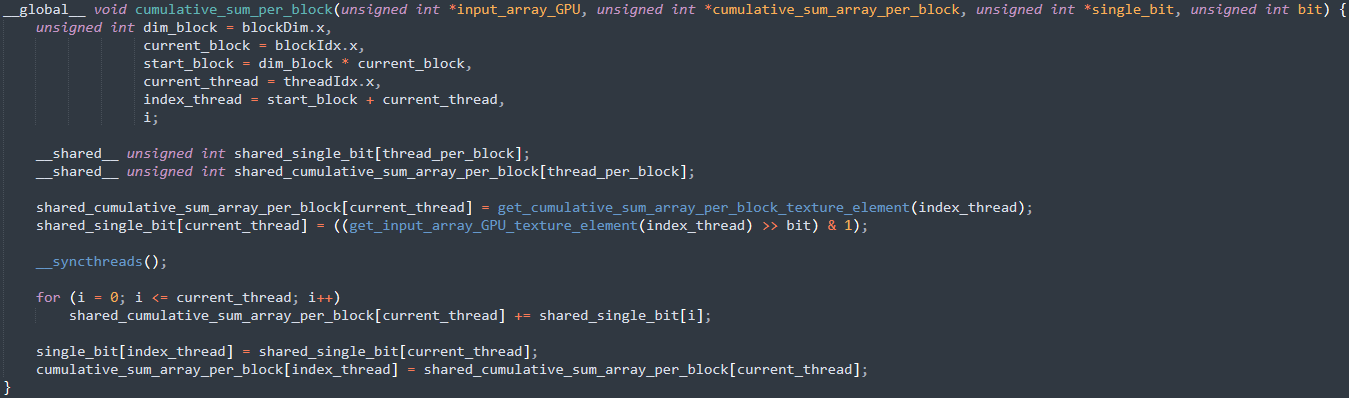
In particular, respect to the easy version that uses global memory, the shared memory is designed to optimize cyclic operations which require multiple memory accesses in order to justify the overhead of the data initialization from global to shared. In addition to that, the decision to use shared only for specific arrays is also linked to the fact that operations are limited to the same block; having a very limited amount of shared memory per block avoids saturation of the same as it is certain that the data stored does not fill it.

The use of *\_\_syncthreads()* has been adopted to use a consistent shared memory.



*Figure 17 – use case of shared memory*

Finally, the texture memory version followed a mixed approach, in that shared memory was also used, paying attention about read-only cache.



*Figure 18 – use case of texture memory*

# Test case

To check the correctness of the ordered array, it has been implemented a function which compare the sorted array produced by the CPU and the one produced by GPU:

* void ***compare\_output\_between\_CPU\_and\_GPU*** (unsigned int \*output\_array\_CPU, unsigned int \*output\_array\_GPU)

Test function to compare output between CPU and GPU.

**Parameters**

* + output\_array\_CPU pointer to the sorted vector by CPU.
  + output\_array\_GPU pointer to the sorted vector by GPU.

Indeed, to determine the number wrong elements it has been implemented another function:

* unsigned int ***errors\_in\_comparison\_between*** (unsigned int \*output\_array\_CPU, unsigned int \*output\_array\_GPU)

Test function to detect errors between CPU and GPU.

**Parameters**

* output\_array\_CPU pointer to the sorted vector by CPU.
* output\_array\_GPU pointer to the sorted vector by GPU.

# API

## Implemented functions

|  |  |  |
| --- | --- | --- |
| Type | Name | Used by |
| unsigned int | ***get\_max*** (unsigned int \**array*)  Function to get the largest element from an array. | ***CPU*** |
| void | ***localsort*** (unsigned int \**array*, unsigned int *place*)  Function to execute localsort. | ***CPU*** |
| void | ***radix\_sort\_on\_CPU*** (unsigned int \*input\_array, unsigned int \*output\_array\_CPU, double \*elapsed\_from\_CPU)  Function to perform RadixSort by using CPU. | ***CPU*** |
| unsigned int | ***get\_input\_array\_GPU\_texture\_element*** (unsigned int index)  Device function to get element from texture reference. | ***Texture*** |
| unsigned int | ***get\_cumulative\_sum\_array\_per\_block\_texture\_element*** (unsigned int index)  Device function to get element from texture reference. | ***Texture*** |
| unsigned int | ***get\_cumulative\_sum\_array\_per\_grid\_texture\_element*** (unsigned int index)  Device function to get element from texture reference. | ***Texture*** |
| unsigned int | ***get\_single\_bit\_texture\_element*** (unsigned int index)  Device function to get element from texture reference. | ***Texture*** |
| void | ***cumulative\_sum\_per\_block*** (unsigned int \*input\_array\_GPU, unsigned int \*cumulative\_sum\_array\_per\_block, unsigned int \*single\_bit, unsigned int bit)  Kernel to perform cumulative sum per block. | ***Global Shared Texture*** |
| void | ***counting\_ones\_per\_grid*** (unsigned int \*cumulative\_sum\_array\_per\_block, unsigned int \*cumulative\_sum\_array\_per\_grid)  Kernel to count previous ones for each thread. | ***Global Shared Texture*** |

|  |  |  |
| --- | --- | --- |
| void | ***sorting\_per\_bit*** (unsigned int \*input\_array\_GPU, unsigned int \*result\_array, unsigned int \*cumulative\_sum\_array\_per\_grid, unsigned int \*single\_bit)  Kernel to perform sorting per bit. | ***Global Shared Texture*** |
| float | ***perform\_milliseconds\_to\_seconds*** (float elapsed\_in\_ms)  Function to perform time from milliseconds to seconds. | ***Global Shared Texture*** |
| unsigned int | ***count\_flops*** (unsigned int number\_of\_all\_blocks)  Function to count flops. | ***Global***  ***Shared***  ***Texture*** |
| float | ***perform\_Mflop\_per\_sec*** (float elapsed\_in\_ms, unsigned int number\_of\_all\_blocks)  Function to perform Mflops. | ***Global Shared Texture*** |
| void | ***check\_error\_from*** (cudaError\_t cuda\_error, bool show\_result = true)  Function to check error from CUDA operation. | ***Global Shared Texture*** |
| void | ***radix\_sort\_on\_GPU*** (unsigned int \*input\_array, unsigned int \*output\_array\_GPU, unsigned int \*number\_of\_all\_blocks, float \*mflop\_per\_sec, float \*elapsed\_from\_GPU)  Function to perform RadixSort by using GPU. | ***Global Shared Texture*** |
| void | ***read\_input\_from\_file*** (unsigned int \**array*)  Function to read an unsorted array from a file. | ***CPU*** |
| bool | ***file\_exist*** (char \*filename)  Function to check if file exists. | ***CPU*** |
| void | ***make\_csv\_of*** (unsigned int number\_of\_all\_blocks, float mflop\_per\_sec, float elapsed\_from\_GPU, double elapsed\_from\_CPU)  Function to make a csv of the measurements. | ***CPU*** |
| unsigned int | ***errors\_in\_comparison\_between*** (unsigned int \*output\_array\_CPU, unsigned int \*output\_array\_GPU)  Test function to detect errors between CPU and GPU. | ***CPU*** |
| void | ***compare\_output\_between\_CPU\_and\_GPU*** (unsigned int \*output\_array\_CPU, unsigned int \*output\_array\_GPU)  Test function to compare output between CPU and GPU. | ***CPU*** |

## Implemented Functions Documentation

* unsigned int ***get\_max*** (unsigned int \*array)

Function to get the largest element from an array.

**Parameters**

* + array pointer to the vector to be sorted.
* void ***localsort*** (unsigned int \**array*, unsigned int *place*)

Function to detect errors between CPU and GPU.

**Parameters**

* + *array* pointer to the vector to be sorted.
  + *place* current digit which it is considered (units, tens, hundreds, ...).
* void ***radix\_sort\_on\_CPU*** (unsigned int \*input\_array, unsigned int

\*output\_array\_CPU, double \*elapsed\_from\_CPU)

Function to compare output between CPU and GPU.

**Parameters**

* + input\_array pointer to the vector to be sorted.
  + output\_array\_GPU pointer to the sorted vector.
  + elapsed\_from\_CPU elapsed time committed from CPU
* \_\_device\_\_ unsigned int ***get\_input\_array\_GPU\_texture\_element*** (unsigned int

\*index)

Device function to get element from texture reference.

**Parameters**

- index of input\_array\_GPU\_texture\_reference.

* \_\_device\_\_ unsigned int ***get\_cumulative\_sum\_array\_per\_block\_texture\_element***

(unsigned int \*index)

Device function to get element from texture reference.

**Parameters**

- index index of cumulative\_sum\_array\_per\_block\_texture\_reference.

* \_\_device\_\_ unsigned int ***get\_cumulative\_sum\_array\_per\_grid\_texture\_element***

(unsigned int \*index)

Device function to get element from texture reference.

**Parameters**

- index index of cumulative\_sum\_array\_per\_grid\_texture\_reference.

* \_\_device\_\_ unsigned int ***get\_single\_bit\_texture\_element*** (unsigned int \*index) Device function to get element from texture reference.

**Parameters**

* + index index of single\_bit\_texture\_reference.
* \_\_global\_\_ void **cumulative\_sum\_per\_block** (unsigned int \*input\_array\_GPU, unsigned int \*cumulative\_sum\_array\_per\_block, unsigned int \*single\_bit, unsigned int bit)

Kernel to perform cumulative sum per block.

**Parameters**

* + input\_array\_GPU pointer to the vector to be sorted.
  + cumulative\_sum\_array\_per\_block pointer to the vector which contains

cumulative sum for each thread starting from the beginning of the own block.

* + single\_bit pointer to the vector which contains the single bit of the numbers.
  + bit the current bit of the iteration taking into account.
* \_\_global\_\_ void **counting\_ones\_per\_grid** (unsigned int

\*cumulative\_sum\_array\_per\_block, unsigned int \*cumulative\_sum\_array\_per\_grid) Kernel to count previous ones for each thread.

**Parameters**

* + cumulative\_sum\_array\_per\_block pointer to the vector which contains

cumulative sum for each thread starting from the beginning of the own block.

* + cumulative\_sum\_array\_per\_grid pointer to the vector which contains

cumulative sum for each thread starting from the beginning of the grid.

* \_\_global\_\_ void **sorting\_per\_bit** (unsigned int \*input\_array\_GPU, unsigned int

\*result\_array, unsigned int \*cumulative\_sum\_array\_per\_grid, unsigned int \*single\_bit) Kernel to perform sorting per bit.

**Parameters**

* + input\_array\_GPU pointer to the vector to be sorted.
  + result\_array pointer to the vector to the sorted vector.
  + cumulative\_sum\_array\_per\_grid pointer to the vector which contains

cumulative sum for each thread starting from the beginning of the grid.

* + single\_bit pointer to the vector which contains the single bit of the numbers.
* float **perform\_milliseconds\_to\_seconds** (float elapsed\_in\_ms)

Function to perform time from milliseconds to seconds.

**Parameters**

* + elapsed\_in\_ms elapsed time in milliseconds.
* unsigned int **count\_flops** (unsigned int number\_of\_all\_blocks)

Function to count flops.

**Parameters**

* + number\_of\_all\_blocks number of all grid's blocks into the GPU.
* float **perform\_Mflop\_per\_sec** (float elapsed\_in\_ms, unsigned int number\_of\_all\_blocks)

Function to perform Mflops.

**Parameters**

* + elapsed\_in\_ms elapsed time in milliseconds.
  + number\_of\_all\_blocks number of all grid's blocks into the GPU.
* void **check\_error\_from** (cudaError\_t cuda\_error, bool show\_result = true) Function to check error from CUDA operation.

**Parameters**

* + cuda\_error CUDA error.
  + show\_result print eventual error into the console.
* void **radix\_sort\_on\_GPU** (unsigned int \*input\_array, unsigned int

\*output\_array\_GPU, unsigned int \*number\_of\_all\_blocks, float \*mflop\_per\_sec, float

\*elapsed\_from\_GPU)

Function to perform RadixSort by using GPU.

**Parameters**

* input\_array pointer to the vector to be sorted.
* output\_array\_GPU pointer to the sorted vector.
* number\_of\_all\_blocks number of all grid's blocks into the GPU.
* mflop\_per\_sec Mflops.
* elapsed\_from\_GPU elapsed time committed from GPU.
* void **read\_input\_from\_file** (unsigned int \*array)

Function to read an unsorted array from a file.

**Parameters**

* + array pointer to the sorted vector.
* bool **file\_exist** (char \*filename)

Function to check if file exists.

**Parameters**

* + filename directory.
* void **make\_csv\_of** (unsigned int number\_of\_all\_blocks, float mflop\_per\_sec, float elapsed\_from\_GPU, double elapsed\_from\_CPU)

Function to make a csv of the measures.

**Parameters**

* + number\_of\_all\_blocks number of all grid's blocks into the GPU.
  + mflop\_per\_sec Mflops.
  + elapsed\_from\_GPU elapsed time committed from GPU.
  + elapsed\_from\_CPU elapsed time committed from CPU.
* unsigned int ***errors\_in\_comparison\_between*** (unsigned int \*output\_array\_CPU, unsigned int \*output\_array\_GPU)

Test function to detect errors between CPU and GPU.

**Parameters**

* + output\_array\_CPU pointer to the sorted vector by CPU.
  + output\_array\_GPU pointer to the sorted vector by GPU.
* void ***compare\_output\_between\_CPU\_and\_GPU*** (unsigned int \*output\_array\_CPU, unsigned int \*output\_array\_GPU)

Test function to compare output between CPU and GPU.

**Parameters**

* + output\_array\_CPU pointer to the sorted vector by CPU.
  + output\_array\_GPU pointer to the sorted vector by GPU.

# How to run

**Folders**

* `*global*`: contains the script GLOBAL\_MEMORY.cu;
* `*shared*`: contains the script SHARED\_MEMORY.cu;
* `*texture*`: contains the script TEXTURE\_MEMORY.cu;
* `*measures*`: contains the measures of global, shared and texture memory **(DO NOT DELETE `*measures*` FOLDER!!)**

**Common files**

These files have been made to generate measures and extract means of them:

* `*cuda\_execute.bash*`  bash script that runs .cu files 200 times
* `*cuda\_means.py*`  python script that calculates means of the measures
* `*random\_numbers.txt*`  text file contains random numbers to be sorted

**How to run single program**

Each version of this program is executed from the local GPU:

1. `nvcc .\name\_of\_program.cu -o .\name\_of\_program` to compile the program;
2. `./name\_of\_program` to execute the program.

**How to run all programs**

To run the bash file, after any change, **it is necessary previously to compile global, shared and texture version**:

1. `bash cuda\_execute.bash`

So, to extract the mean of the measure and to print them on the terminal (paying attenction to the set correctly **memory\_type\_list** and **thread\_per\_block\_list** parameters since they depend from the content of `measures` folder):

1. `python3 cuda\_means.py`

# Bibliography

The sequential part of the algorithm is taken from Internet at the link: <https://www.programiz.com/dsa/radix-sort>

The idea of CUDA algorithm is taken from Internet at the link: [http://www.compsci.hunter.cuny.edu/~sweiss/course\_materials/csci360/lecture\_notes/radix\_](http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/lecture_notes/radix_sort_cuda.cc) [sort\_cuda.cc](http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/lecture_notes/radix_sort_cuda.cc)

# License

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike

4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.