Register number _____

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu
Academic Year: 2023-24 (EVEN)
B.Tech-Computer Science & Engineering **SET - B**

Test: CLA-T2 **Date: 28.03.2024**
Course Code & Title: 18CSE419T & GPU Programming **Duration: 2 periods**
Year & Sem: III Year /VI Sem **Max. Marks: 50**

Course articulation matrix:

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO-1	3														3
CO-2		3	2												3
CO-3		3	3												3
CO-4		3	3												3
CO-5			3	1									2		3

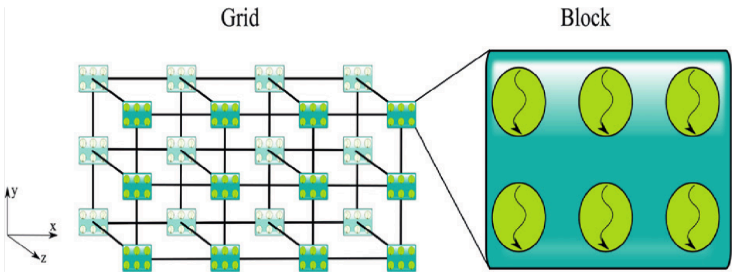
Part – A(1*10=10 Marks)
Answer All the Questions

Q. No	Questions	Marks	BL	CO	PO	PI Code
1	NVDIA GTX680 GPU has a) 12,384 threads b) 16,384 threads c) 4096 threads d) 32,384 threads **Ans:B**	1	3	CO 3	2 & 3	**4.2.1**
2	In CUDA Programming blockDim.x represents a) Number of threads per block b) Number of blocks in a kernel c) Number of blocks in a grid d) Number of blocks in a warp **Ans:A**	1	3	CO 3	2 & 3	**4.2.1**

3	What is the CUDA function call required to copy an array h_A from the CPU memory to the GPU memory as d_A?
	a) cudaMemcpy(h_A,d_A,size,cudaMemcpyHost to Device);
	b) cudaMemcpy(d_A,h_A,size,cudaMemcpyHost to Device);
	c) cudaMemcpy(h_A,d_A,size,cudaMemcpyDevice to Host);
	d) cudaMemcpy(d_A,h_A,size, cudaMemcpyDevice to Host);
	Ans:B
		1	3	CO 3	2 & 3	**4.2.1**

4	Each warp of GPU receives a single instruction and "broadcasts" it to all of its threads is a ---------- operation. a) SIMD b) SIMT c) SISD d) SIST **Ans:B**	1	3	CO 3	2 & 3	**4.2.1**
5	NVDIA Fermi is a ---------- CUDA core device a) 512 b) 128 c) 256 d) 64 **Ans:A**	1	3	CO 3	2 & 3	**4.2.1**

Register number _____

6	The smallest CUDA thread block dimension is a) 8 b) 16 c) 32 d) 64 **Ans:C**	1	3	CO 3	2 & 3	**4.2.1**
7	Which of the following memory locations is common for all the SMs in a typical CUDA GPU? a) Thread-local memory b) L1 cache c) L2 cache d) Shared memory **Ans:C**	1	3	CO 3	2 & 3	**4.2.1**
8	What is the term used for the combination of CPU and GPU in a hybrid computing system? a) Homogenous computing b) Many-core architecture c) Hardware accelerator d) Heterogenous computing **Ans:D**	1	3	CO 3	2 & 3	**4.2.1**
9	The scope of a constant memory is a) Thread b) Block c) Warp d) Grid **Ans:D**	1	3	CO 3	2 & 3	**4.2.1**
10	If each CUDA block can hold a maximum of 512 threads then how many CUDA blocks would be created to process 4000 vector elements a) 7 b) 8 c) 10 d) 16 **Ans:B**	1	3	CO 3	2 & 3	**4.2.1**
	Part – B (4*4=16 marks) **Answer any four Questions**					
Q. No	**Question**	**Marks**	**BL**	**CO**	**PO**	**PI Code**
11	Sketch threads , blocks and grids in CUDA programming model and mention the functions to retrieve the values of it.  Threads are organized in a hierarchy of two levels, as shown in Figure 6.3. At the lower level, threads are organized in blocks that can be of one, two or three dimensions. Blocks are then organized in grids of one, two, or three dimensions. The sizes of the blocks and grids are limited by the capabilities of the target device. Each of the CUDA threads is aware of its position in the grid/block hierarchy via the following intrinsic/built-in structures, all having 3D components x, y, and z:	4	2	CO 2	2 & 3	**4.2.1**

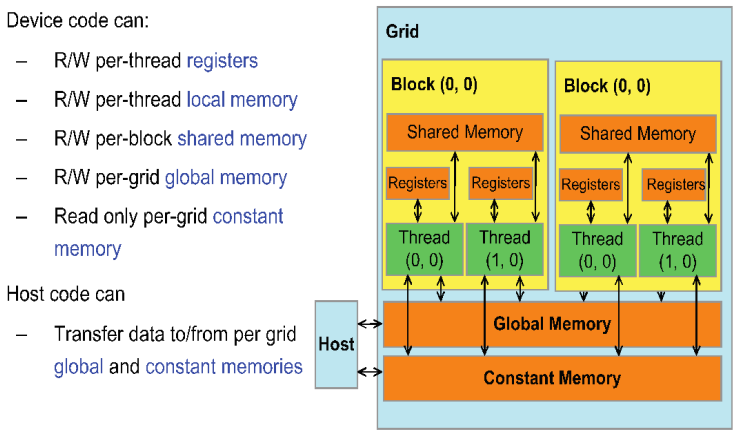
	• blockDim: Contains the size of each block, e.g., (B_x, B_y, B_z). • gridDim: Contains the size of the grid, in blocks, e.g., (G_x, G_y, G_z). • threadIdx: The (x, y, z) position of the thread within a block, with $x \in [0, B_x - 1]$, $y \in [0, B_y - 1]$, and $z \in [0, B_z - 1]$. • blockIdx: The (b_x, b_y, b_z) position of a thread's block within the grid, with $b_x \in [0, G_x - 1]$, $b_y \in [0, G_y - 1]$, and $b_z \in [0, G_z - 1]$.					
12	Sketch the CUDA functions as a 3D array of blocks and state the functions involved.	4	3	CO 2 & CO 3	2 & 3	**4.2.1**

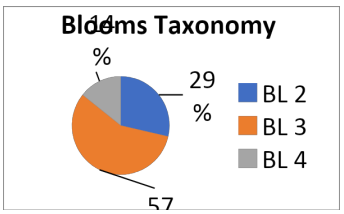
13	What is pinned memory in GPU?. State the functions of it.
	• The term page-locked or pinned memory refers to host memory that cannot be swapped out as part of the regular virtual memory operations employed by most contemporary operating systems.
	• Pinned memory is **used to hold critical code and data t**hat cannot be moved out of the main memory, such as the OS kernel.
	• It is also needed for performing Direct Memory Access (DMA) transfers across the PCIe bus.
	• When one uses regular memory for holding the host data, upon a request to transfer the data to the device, the Nvidia driver allocates paged-locked memory, copies the data to it, and, upon the completion of the transfer, frees up the pinned memory.
	• This buffering overhead can be eliminated by using pinned memory for all data that are to be moved between the host and the device.
	Pinned memory can be allocated with:
	• malloc(), followed by a call to mlock().
	Deallocation is done in the reverse order, i.e., calling munlock(), then free().
	• Or by calling the **cudaMallocHost()** function. Memory allocated in this fashion has to be deallocated with a call to **cudaFreeHost().** Otherwise, the program may behave in an unpredictable manner:
	cudaError_t cudaMallocHost(
	void ∗∗ ptr, // Addr . of pointer to pinned
	// memory (IN/OUT)
	size_t size) ; // Size in bytes of request (IN)
	cudaError_t cudaFreeHost (v o i d ∗ ptr) ;
		4	3	CO 3	2 & 3	**4.2.1**

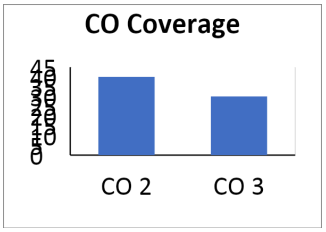
14	Discuss about CUDA's variable type qualifiers. **Table 5.1** CUDA Variable Type Qualifiers Variable Declaration — Memory — Scope — Lifetime Automatic variables other than arrays — Register — Thread — Kernel Automatic array variables — Local — Thread — Kernel `__device__ __shared__ int SharedVar;` — Shared — Block — Kernel `__device__ int GlobalVar;` — Global — Grid — Application `__device__ __constant__ int ConstVar;` — Constant — Grid — Application	4	3	CO 3	2 & 3	**4.2.1**
15	Write short notes on Global memory of GPU. • GPU global memory is global because it's writable from both the GPU and the CPU. • It can actually be accessed from any device on the PCI-E bus. • GPU cards can transfer data to and from one another, directly,	4	3	CO 3	2 & 3	**4.2.1**

Register number _____

	without needing the CPU.
	• The memory from the GPU is accessible to the CPU host processor in one of three ways:
	  • Explicitly with a blocking transfer.
	  • Explicitly with a nonblocking transfer.
	  • Implicitly using zero memory copy.
	• The memory on the GPU device sits on the other side of the PCI-E bus.
	• This is a bidirectional bus that, in theory, supports transfers of up to 8 GB/s (PCI-E 2.0) in each direction.
	• In practice, the PCI-E bandwidth is typically 4–5 GB/s in each direction.
	• The usual model of execution involves the CPU transferring a block of data to the GPU, the GPU kernel processing it, and then the CPU initiating a transfer of the data back to the host memory.
	• A slightly more advanced model of this is where we use streams (covered later) to overlap transfers and kernels to ensure the GPU is always kept busy, as shown in Figure 6.16.
	• Figure 6.16, the memory accesses are pipelined.
	• By creating a ratio of typically 10:1 of threads to number of memory accesses, you can hide memory latency, but only if you access global memory in a pattern that is coalesced.					

	Part – C (2*12=24 marks) **Answer any two Questions**					
16	Examine the different CUDA device memory types with a neat sketch.  **FIGURE 5.2** Overview of the CUDA device memory model.	12	3	CO 3	2 & 3	**4.2.1**
17	How kernel structure can be optimized (i) to solve stalling problem (ii) smaller number of threads per block.	12	3	CO 3	2 & 3	**4.2.1**
18	Interpret a dynamic shared memory allocation kernel through a CUDA Histogram calculation.	12	3	CO 3	2 & 3	**4.2.1**





Approved by Audit Professor/ Course Coordinator

Register number _____