1. What is Cuda Architecture?

a.CUDA Architecture included a unified shader pipeline, allowing each and every chip to be marshaled by a program.

b.CUDA Architecture included a unified shader pipeline, allowing each and every unit on the chip to be marshaled by a program intending to perform general-purpose computations

c.CUDA Architecture included a unified shader pipeline, allowing each and every logic unit on the chip to be marshaled by a program intending to perform general-purpose computations

d.CUDA Architecture included a unified shader pipeline, allowing each and every arithmetic logic unit (ALU) on the chip to be marshaled by a program intending to perform general-purpose computations

Ans.D

1. For the following code write a kernel

\_\_global\_\_ void kernel( void ) { }

int main( void ) {

// Write a kernel here

printf( "Hello, World!\n" ); return 0; }

a.kernel<1, 1>(1,1);

b.kernel<<<1, 1>>>(1,1);

c.kernel<<<1, 1>>>();

d.kernel<<1, 1>>();

Ans. c

1. Find out which is the kernel from following code:

#include <iostream>

\_\_global\_\_ void add( int a, int b, int \*c ) {

\*c = a + b;

}

int main( void ) {

int c; int \*dev\_c;

HANDLE\_ERROR( cudaMalloc( (void\*\*)&dev\_c, sizeof(int) ) );

add<<<1,1>>>( 2, 7, dev\_c );

HANDLE\_ERROR( cudaMemcpy( &c, dev\_c, sizeof(int), cudaMemcpyDeviceToHost ) );

printf( "2 + 7 = %d\n", c );

cudaFree( dev\_c );

return 0;

}

a.cudaMalloc( (void\*\*)&dev\_c, sizeof(int) )

b.add<<<1,1>>>(2, 7, dev\_c)

c.add<<1,1>>( 2, 7, dev\_c );

d.add<<<1,1>>>()

Ans.b

1. From following code which particular line is responsible for copying between device to host

#include <iostream>

\_\_global\_\_ void add( int a, int b, int \*c ) {

\*c = a + b;

}

int main( void ) {

int c; int \*dev\_c;

HANDLE\_ERROR( cudaMalloc( (void\*\*)&dev\_c, sizeof(int) ) );

add<<<1,1>>>( 2, 7, dev\_c );

HANDLE\_ERROR( cudaMemcpy( &c, dev\_c, sizeof(int), cudaMemcpyDeviceToHost ) );

printf( "2 + 7 = %d\n", c );

cudaFree( dev\_c );

return 0;

}

1. c, dev\_c, sizeof(int);
2. HANDLE\_ERROR( &c, dev\_c, sizeof(int), cudaMemcpyDeviceToHost );
3. HANDLE\_ERROR( cudaMemcpy( &c, dev\_c, sizeof(int), cudaMemcpyDeviceToHost ) );
4. cudaMemcpy( &c, dev\_c, sizeof(int), cudaMemcpyDeviceToHost ) ;

Ans.c

1. What is output of the following code:

#include <iostream>

\_\_global\_\_ void add( int a, int b, int \*c ) {

\*c = a + b;

}

int main( void ) {

int c; int \*dev\_c;

HANDLE\_ERROR( cudaMalloc( (void\*\*)&dev\_c, sizeof(int) ) );

add<<<1,1>>>( 2, 7, dev\_c );

HANDLE\_ERROR( cudaMemcpy( &c, dev\_c, sizeof(int), cudaMemcpyDeviceToHost ) );

printf( "2 + 7 = %d\n", c );

cudaFree( dev\_c );

return 0;

}

a.2

b.9

c.7

d.0

Ans. b

6.what is function of e \_\_global\_\_ qualifier in cuda program

1. alerts the compiler that a function should be compiled to run on a device instead of the host
2. alerts the interpreter that a function should be compiled to run on a device instead of the host
3. alerts the interpreter that a function should be interpreted to run on a device instead of the host
4. alerts the interpreter that a function should be compiled to run on a host instead of the device

ans.a

7. The on-chip memory which is local to every multithreaded Single Instruction Multiple Data (SIMD) Processor is called

a. Local Memory

b. Global Memory

c. Flash memory

d. Stack

Ans. a

8. The machine object created by the hardware, managing, scheduling, and executing is a thread of

a. DIMS instructions

b. DMM instructions

c. SIMD instructions

d. SIM instructions

Ans. c

9. The primary and essential mechanism to support the sparse matrices is

a. Gather-scatter operations

b. Gather operations

c. Scatter operations

d. Gather-scatter technique

Ans. a

10. Which of the following architectures is/are not suitable for realizing SIMD ?

a. Vector Processor

b. Array Processor

c. Von Neumann

d. All of the above

Ans . c

11. Multithreading allowing multiple-threads for sharing the functional units of a

a.Multiple processor

b.Single processor

c.Dual core

d. Corei5

Ans . b

12. Which compiler is used to compile the cude source code:

a.gcc

b.nvc++

c.nc++

d.nvcc

Ans.d

13. which command line is used to execute a cuda program :

a.nvcc hello.cu -o hello

b.nvg++ heloo.cpp -o hello

c.ncc hello.c -o hello

D.g++ hello.cu -o hello

Ans.a

14.The syntax of kernel execution configuration is as follows

a.<<< M , T >>> with a grid of M thread blocks. Each thread block has T parallel blocks

b.<<< M , T >>> with a grid of M blocks. Each thread block has T parallel threads

c.<<< M , T >>> with a grid of M thread blocks. Each thread block has T parallel threads

d.<<< M , T >>> with a grid of M thread blocks. Each thread block has T threads

Ans. c

15.what it contains threadIdx.x

A.contains the index of the thread within the block

b.contains the index of the block within the thread

c.contains the index of the thread size within the block

d.contains the index of the block size within the thread

Ans. A

16.what it contains blockDim.x

a.contains the size of block

b.contains the size of block thread

c.contains the size of thread block (number of threads in the thread block).

d.the size of thread block

Ans. c

17.memory allocation of of variable x and y in cuda:

A.float \*b, \*a;

cudaMallocManaged(&, N\*sizeof(float));

cudaMallocManaged(&, N\*sizeof(float));

B.float \*x, \*y;

cudaMallocManaged(&a, N\*sizeof(float));

cudaMallocManaged(&b, N\*sizeof(float));

c.float \*a, \*b;

cudaMallocManaged(&x, N\*sizeof(float));

cudaMallocManaged(&y, N\*sizeof(float));

d.float \*x, \*y;

cudaMallocManaged(&x, N\*sizeof(float));

cudaMallocManaged(&y, N\*sizeof(float));

Ans. d

18.which function is used for free the memory in cuda

a.cudaFree()

b.Free()

c.Cudafree()

d.CudaFree()

Ans. a

19. Which of the following is *not* a form of parallelism supported by CUDA

a.Vector parallelism - Floating point computations are executed in parallel on wide vector units

b.Thread level task parallelism - Different threads execute a different tasks

c.Block and grid level parallelism - Different blocks or grids execute different tasks

d.Data parallelism - Different threads and blocks process different parts of data in memory

Ans . a

20.The style of parallelism supported on GPUs is best described as

a.SISD - Single Instruction Single Data

b.MISD - Multiple Instruction Single Data

c.SIMT - Single Instruction Multiple Thread

d.MIMD - Multiple Instruction Multiple Data

Ans. c

21. Which of the following correctly describes a GPU kernel

a.A kernel may contain a mix of host and GPU code

b.All thread blocks involved in the same computation use the same kernel

c.A kernel is part of the GPU's internal micro-operating system, allowing it to act as in independent host

d.All thread blocks involved in the same computation use the different kernel

Ans .b

22.Shared memory in CUDA is accessible to:

a.All threads in a single block

b.Both the host and GPU

c.All threads associated with a single kernel

d.one thread in a single block

Ans.a

23.Which of the following correctly describes the relationship between Warps, thread blocks, and CUDA cores?

a.A warp is divided into a number of thread blocks, and each thread block executes on a single CUDA core

b.A thread block may be divided into a number of warps, and each warp may execute on a single CUDA core

c.A thread block is assigned to a warp, and each thread in the warp is executed on a separate CUDA core

d. A block index is same as thread index

Ans .b

24. A processor assigned with a thread block, that executes a code ,which we usually call a

A. multithreaded MIMD processor

b. multithreaded SIMD processor

c. multithreaded

D. multicore

Ans. c

25. Thread blocked altogether and being executed in the sets of 32 thread called as

a.block of thread

b.thread block

c.thread

d.block

Ans. b

26.Who developed CUDA :

a. ARM

b. INTEL

c. AMD

d. NVIDIA

Ans. d