...umerate and give a brief description of the main opportunities of parallelism.

1- insturction level paralleism: paralleism which are hidden in comput programs
2- single computer paralleism: paralleism using multi-core or ~~multi-pro~~ multi-processors which are in a single computer
3- multi computer paralleism: paralleism using multiple computers such as grids, servers, and clusters

3

2. Give the main differences between parallel processing and Distributed computing.

2

1- in distributed computing processors are distant geographically but in parallel processing processors are in the same machine
2- in distributed computing the goal is to provide avibility and realibity but in parallel processing the goal is to increase preformance
3- IN The back of the page →

3. Enumerate and give a brief description of the main aspects of parallel computing.

1- parallel computing archtiture: PRAM, CTA
2- parallel algorthims and application: Reasoning and designe
3- parallel programing:
paradigmes
models
parallel programing language
frame works
enviraments

3

Q1:

$\boxed{2}$

3-in disturbuted computing the interaction are infrequent and corase graind, but in parallel processin the interaction interaction are frequent and fine graind

Question 2

1. Explain the main differences between the **Blocking non-buffered** and the **Non-Blocking non-buffered** send/receive operations of the message passing paradigm.

* in blocking non-buffered when the sender issues a send operation he has to wait until the reciver match a recive operation, but in non-blocking non-buffered when the sender issues a send operation he can continue processing until the reciver send a interrupt

2

2. Describe the ***Task Farming*** and the ***Divide-and-Conquer*** programming models and explain the main differences between them.

- Task farming has two entitbes the master and the workers, the master split the problem into tasks and send it to the workers and gather the result from them, the worker take a task and process it then send it to the master
- divid-and-conquer decompose the problem into sub-tasks recursivly until it reach base case which can be solved directly

2

the main differnces:
1- in divde-and-conquer we decompose the task recursivly but in task farming we split the tasks
2- in divde-and-conquer sub-tasks have the same nature as the main problem but in task farming tasks may have diffrent nature

Question 3

1. Enumerate and explain the different types of memory adopted by CUDA.

1- register: per thread, not cached, on chip, fast, read\write
2- Local-memory: per thread, not cached, on DRAM, read\write
3- shared-memory: per block, not cached, on chip, synchronize block threads read\write
4- Global-memory: per Grid, not cached, on DRAM, synchronize between grids read\write
5- constant-memory: per Grid, cached, on DRAM, read only
6- texture-memory: per Grid, cached, on DRAM, read only

3

2. Explain why the Global memory is not cached in CUDA, while the Constant memory is cached.

Global memory is not cached because it is read and write so the values may change and it will be different if we cached it, but in constant memory it is only read so we can cache the value and it will not change

1

2

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Midterm Exam – Fall 2021

Question 4

Let's consider 2 arrays of integers A and B of size *S*. Let's consider that we would like to write a C program that runs in parallel and that computes the sum of 2 arrays as following:

C[i] = A[i] + B[i]

Let's consider the following kernel:

```
__global__ void add(int *a, int *b, int *c, int size) {
        int cell_id = ......................;
        if (cell_id < size)
                c[cell_id] = a[cell_id] + b[cell_id];
}
```

1. We would like to run this kernel on grid composed of **1 block** where every thread evaluates a single cell as shown in the following figure:

Block (0, 0)

Cell 0	Cell 3	Cell 6	Cell 9	Cell 12
Cell 1	Cell 4	Cell 7	Cell 10	Cell 13
Cell 2	Cell 5	Cell 8	Cell 11	Cell 14

threadidx.y + (threadidx.x * blockdim.y)

2(2x3) =6

- Give the formula that allows every thread to compute the cell_id of the cell he is going to process.

int cell_id = ThreadIdx.y + (ThreadIdx.x * blockDim.y)

2

... to run this kernel on grid configured as M * N matrix of thread blocks. Every thread evaluates a single cell as shown in the following figure:

Block (0, 0)

Cell 0	Cell 6			Cell 24
Cell 1	Cell 7			Cell 25
Cell 2	Cell 8			Cell 26

Block (1, 0)

Cell 30	Cell 36			Cell 54
Cell 31	Cell 37			Cell 55
Cell 32	Cell 38			Cell 56

Block (0, 1)

Cell 3	Cell 9			Cell 27
Cell 4	Cell 10			Cell 28
Cell 5	Cell 11			Cell 29

Block (1, 1)

Cell 33	Cell 39			Cell 57
Cell 34	Cell 40			Cell 58
Cell 35	Cell 41			Cell 59

- Give the formula that allows every thread to compute the cell_id of the cell he is going to process.

int cell_id = Thread.y + (blockIdx.y * blockDim.y) +
(ThreadIdx.x * blockDim.y * GridDim.y) +
(blockIdx.x * blockDim.x * blockDim.y * GridDim.y);

2

30
24
54

4 3 3 2

thread.y * (blockidx.y * blockdim.y) + (threadidx.x * blockdim.y * griddim.y) +
blockidx.x * blockdim.x * blockdim.y * GDim.y

5 3 9 2

Let's consider 2 arrays of integers A and B of size *S*. Let's consider that we would like to write a kernel in C that computes the sum of 2 arrays:

C[i] = A[i] + B[i] for i:0 to S-1.

We would like to run this kernel on a grid composed of **1 block** where every thread evaluates *N* cells as shown in the following figure (where *N* = 4 as a sample):

Block (0, 0)

Cells 0-3	Cells 4-7	Cells 8-11	Cells 12-15	Cells 16-19
Cells 20-23	Cells 24-27	Cells 28-31	Cells 32-35	Cells 36-39
Cells 40-43	Cells 43-47	Cells 48-51	Cells 52-55	Cells 56-60

- Write the kernel

```
__global__ void add(int *a, int *b, int *c, int size, int N) {

int index =(ThreadIdx.x + (ThreadIdx.y * blockDim.x)) * N
for(int i=0; i<N; i++){
  c[index+i]= a[index+i]+b[index+i]
}

}
```

4