Technische Universität Wien

GPU Architecture & Computing — In-class Assignment

Name: _____ **Surname:** _____

Matriculation No.: _____

Version A **Duration: 60 minutes** **Total: 20 points**

Instructions. Answer all questions clearly. Unless explicitly requested, short and precise answers are sufficient. Calculations and reasoning must be shown when relevant.

Exercise 1 — Architecture and Parallelism [5 pts]

Answer the following briefly.

1.1. State the idea behind Moore's law and explain why simply increasing CPU frequency is no longer the only practical way to improve performance.

1.2. What does Dennard scaling claim, and what key issue caused it to break down in modern processors?

1.3. Distinguish between *instruction-level parallelism*, *data-level parallelism*, and *thread-level parallelism*. Give one example strategy for each.

1.4. In one or two sentences, explain the difference between *low latency* and *high throughput* hardware design.

Exercise 2 — CUDA Execution Model [5 pts]

Consider the kernel launch

```
myKernel<<<dim3(3,2), dim3(4,2)>>>(...);
```

Assume 2D blocks and 2D grids.

2.1. How many thread blocks are launched in total?

2.2. How many threads are contained in each block?

2.3. How many threads are launched in total?

2.4. For the thread with `blockIdx=(2,1)` and `threadIdx=(3,1)`, compute its global coordinates (i_x, i_y):

$i_x = \texttt{blockIdx.x} \cdot \texttt{blockDim.x} + \texttt{threadIdx.x}, \qquad i_y = \texttt{blockIdx.y} \cdot \texttt{blockDim.y} + \texttt{threadIdx.y}.$

Explain why thread blocks are designed to be independent.

Exercise 3 — Reading CUDA Code [5 pts]

Consider the following kernel and launch:

```
__global__ void kernel(int *a)
{
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    a[idx] = blockIdx.x + threadIdx.x;
}

// launch configuration
kernel<<<4,4>>>(d_a);
```

3.1. How many output elements are written?

3.2. Write the values stored in `a[0]` to `a[15]` after execution.

3.3. Modify only the right-hand side of the assignment so that all four threads of the same block write the same value, and different blocks write different values.

Exercise 4 — Race Conditions and Atomics [5 pts]

Suppose many threads execute the following statement concurrently:

```
colors[c] = colors[c] + 1;
```

4.1. Explain why this statement may produce incorrect results.

4.2. Rewrite the statement using a CUDA atomic operation.

4.3. Name two examples of CUDA atomic operations other than `atomicAdd`.

4.4. Why can heavy use of atomics become a performance bottleneck?

4.5. Give one high-level idea to reduce contention when many threads update the same global value.