INFOMOV 2022 EXAM - June 27, 13:30 – 15:30 - THEATRON

Answer these questions as elaborate as necessary. Don't be too elaborate: incorrect statements in your answer may reduce your score. Negative scores for a question are not possible however. Your grade is calculated as `(pts*9.f/max_pts)+1`.

1. A few **"low-level"** questions:

5pt a) The CPU employs a Re-Order Buffer (ROB) that can hide latencies. The compiler can also re-order the order in which instructions are being executed. Explain why we both need the ROB and the compiler to achieve maximal throughput. The re-order buffer is limited in size, which means that it can only store few instructions to shuffle. The compiler complements this system, by re-ordering mix instructions such that the ROB has more chance to maintain mixed instructions in its buffer.

5pt b) How does the CPU ensure that, even though it re-orders instructions, program functionality remains the same? Instructions are always retired in-order.

5pt c) The CPU employs another method for hiding latencies called *Speculative Execution*. Explain how speculative execution works. Based on previous branching paths maintained by the ROB, the CPU tries to make an educated guess on which path will be taken next. It starts working on executing instructions in this path without retiring them.

5pt d) When your CPU speculates the incorrect path, it causes some overhead. Explain why we get this overhead. The CPU pipeline must be flushed before fetching the correct results.

2. On **Profiling**:

Consider the following scenario: We are optimizing an application that runs in a cloud environment. We have no control over any of the sources of noise in this environment. We would like to know if the new version of our code runs faster.

10pt a) Describe how you would try to achieve the most accurate performance comparison. Discuss which metrics you use, as well as the testing set-up. Use "time-independent" measure (e.g., number of cycles used), run multiple profiling runs, profile the application in a controlled offline environment.

5pt b) What does *CPU utilization* describe? The percentage of reference cycles that were used by our CPU.

5pt c) Low CPU utilization typically indicates a poor performance of our application. Why this is typically true? Because we do not fully satisfy our CPU execution units, indicating that we are probably often waiting for other tasks to finish (e.g., fetching memory, thread synchronization).

5pt d) Describe a situation where we would prefer low CPU utilization. When we develop applications that should run in the background such as Spotify.

3. On **Vectorization**:

10pt a) In the lectures we have discussed **SIMD** (CPU) and **SIMT** (GPU). Describe how both programming models are similar. Both programming models perform the same instructions simultaneously on different data. Branching is handled per vertical stream.

5pt b) How is the memory system on a GPU different from the memory system on a CPU? On the GPU we must explicitly indicate where our memory is being stored. On the CPU we are "talking" to an implicit memory system, meaning that we do not explicitly know where our data resides.

4. A few questions about **Caching**:

The Intel Atom S1260 CPU uses (per core) a 24KB 6-way set associative cache for level 1 data. On this CPU, a cache line is 64 bytes. One set is thus 6 * 64 = 384 bytes.

10pt a) How many sets does this cache consist of? Since you cannot use a calculator, you may also provide a formula. 24 * 1024 / 384 = 64

5pt b) How would you calculate to which <u>set</u> a particular address maps? (A / 64) & 63

5pt c) How would you perform this calculation efficiently, i.e. without floats?
Simple, (A/64)&63. Bit of a trick question; the 384 doesn't complicate anything.

5. Some **Caching** terminology and concepts - Explain, in 30 words or less:

5pt a) Bélády's algorithm Eviction scheme that tells you which datum will not be used for the longest amount of time.

5pt b) In the context of cache design: a 'tag' Remainder of the address, after removing offset and set bits.

5pt c) Also in the context of cache design: a 'collision' Two data mapping to the same set.

6. On **GPGPU**:

The Streaming Multiprocessors on a Nvidia GPU contain L1 cache and shared memory that both share the same physical memory component. We can have several configurations on how the available memory is shared between the two.

10pt a) Describe a situation when we would want more L1 cache and describe a situation when we would prefer more shared memory.
- L1 cache: when the threads on a SM operate on constant data while using random access patterns.
- Shared: when our kernel frequently operate on the same data. When traversing a BVH for example, we can store the top-levels of our BVH in shared memory as each thread processes this data multiple times.

In the second GPGPU lecture we discussed methods for thread-synchronization on a GPU. One example showed the use of *atomics* to create a lock to synchronize threads on a global (brick) level (see code below). We discussed that this example would actually not work in practice.

```
// myArray is an array of non-zero integers
__volatile __global int myArray[…];
__kernel void myKernel(__global int* counter, __global int* result) {
      int id = get_global_id( 0 );
      int ref1 = myArray[tid] * 1;
      atomic_inc(counter);
      while (*counter < NUM_THREADS) {}
      myArray[tid + 1] = 2;
      atomic_inc(counter);
      while (*counter < (NUM_THREADS * 2) - 1) { ... }
      int ref2 = myArray[tid] * 1;
      result[tid] = ref1 * ref2;
}
```

5pt b) Describe why this example does not work. A SM can only work on so many threads simultaneously (e.g., 2048 threads in total → often 2 full work-groups). Our active threads will be spinning forever when our total number of threads exceed the number of SMs times the maximum number of active threads on a SM.

7. A few questions about **"What Every Programmer Should Know About Memory"**:

5pt a) What is DMA? Direct Memory Access; peripherals talking to RAM without CPU assistance.

5pt b) List at least <u>two</u> problems with DRAM design that affect its performance.
Refresh (every 64ms), recharge, time it takes to measure capacitor (with amplification), low clock rate on RAM.

5pt c) Explain the difference between an inclusive and an exclusive cache.
Inclusive keeps L1 data also in L2 and L3; exclusive doesn't.

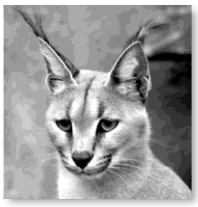
8. On **Data Oriented Design**:

In the lecture on Data Oriented Design it was claimed that the performance of an application is primarily determined by the number of transfers between cache and memory. The guest lecturer confirmed this.

10pt a) Why is this? Because of the memory gap: RAM cost is far higher than expensive opps.

5pt b) Describe one application or algorithm for which this is not true.
Fractals; they use virtually no mem, just calculations.

Gentle reminder: feed the Caracal! →



May the Light be with you!