Example questions in
EDAN26 Multicore Programming

You may answer in English, på svenska, auf Deutsch, или по-русски. All questions below are worth ten points except when noted otherwise (one or two may be reduced to eight or nine points on the exam). On the exam, there will be max 60 points and the grade will be $\lfloor$ your score / 10 $\rfloor$.

Write clearly. Uninterpretable text will be marked with a question mark and zero points.

Examiner: Jonas Skeppstedt

1. Describe sequential consistency. How is the cache coherence protocol used to implement it? Why are compilers and processors not allowed to change the order between two memory accesses?
2. What is the fundamental idea of relaxed memory models that improves performance over sequential consistency, and how does that affect the programmer?
3. Have relaxed memory models any consequences for compilers and hardware design as well? If so, give examples of that!
4. Define Amdahl's Law and explain why it is essential. Give a simple example.
5. Explain at least three important goals of data partitioning!
6. Explain briefly the synchronization primitives mutex and condition variable. Why should the waiting for a condition be in a loop?
7. Cache misses are more costly on large multicore machines than on uniprocessors and one approach to reduce their performance impact is to prefetch data. Why can that be more difficult on multicores, especially if you prefetch and request ownership of the data?
8. What is false sharing and give examples of what programmers can do to reduce the risk of it?
9. Assume you use hardware transactional memory such as on POWER8. Why should you be extra careful to avoid false sharing misses?
10. Explain the difference between release consistency and weak ordering.
11. Compare traditional threads programming with mutexes and condition variables with hardware transactional memory. What is similar and what is different? Which approach do you expect will have most success and why? (there is no right or wrong answer to this, obviously, but you need to make valid arguments for why you write what you write)
12. Suppose you have taken a mutex, what are the effects of unlocking it first and then signaling a condition variable versus signaling first and unlocking afterward?
13. How can you avoid data races in Scala without using locks? Is this technique applicable to other languages as well?
14. Hardware data prefetching can improve performance of many codes. Consider their use with hardware transactional memory. What can happen (good or bad) if the processor starts a transaction, prefetches data, the prefetched data arrives to the cache and is treated by the cache (and the cache coherence protocol) as a read?
15. How does the Rust help you to avoid data races?

16. What does it mean in Rust that an object has *moved*?

17. Why is there a particular risk for bad performance for software transactional memory for languages such as C and C++ ? How is that problem avoided in Clojure?

18. What can `helgrind` help multicore programmers with?

19. What does the atomic test and set machine code instruction do and why do some processor architectures split that into two separate instructions? What do these instructions do? What are they sometimes called (there are different answers and any correct is sufficient).

20. Which memory consistency model is used in the following atomic operation in C/C++ ?

    ```
    count++;
    ```

21. How can a write buffer with read bypass break a sequentially consistent machine?

22. What does `volatile` mean in Java? In C? Why is it not acceptable to use volatile variables in transactions in C?

23. What is good or bad in your opinion with Scala actors? There obviously are different answers but the arguments should be valid.

24. What does it mean that a memory write (such as in the Java code `object.a = 1`) has completed?

25. What is the purpose of the *consume* operation in C/C++? Which other operation is it most similar to and what are the differences?

26. What is the purpose of having tags in the DMA requests on the Cell processor? How can they be used by the programmer to improve performance?

27. Why are linked-lists and trees problematic in the Cell processor if the main POWER processor creates them and the synergistic processing units (SPUs) should read them? What is a better approach to structure data in principle (for the Cell processor)?

28. Suppose you have a very large source code base for numerical computing written as a single-threaded C program, which will be used by ESS. After some measurements you discover that in fact only a small fraction of the loops take most of the time. It's easy to determine that there actually are no data dependencies in these loops. What would you do (several steps probably) to exploit a multicore for this code and why? State your assumptions. There are of course different correct answers.

29. Consider the program fragment below with three threads. Explain how a sequentially consistent machine guarantees that if T_2 reads the value one from A, then if T_3 reads one from B, T_3 will also read one from A. Assume the instructions are executed by the different processors in the order shown.

    ```
    int A = B = C = 0;

    T1:             T2:             T3:
    A = 1;
                    if (A)
                        B = 1;
                                    if (B)
                                        C = A;
    ```

30. Consider again the code in the previous question, a and assume a machine with release consistency. Why would it not be guaranteed that T_3 would see A having the value 1 even if it sees B having the value 1? What can happen?

    What did the programmer forget? Again assume the threads execute their code in the order T_1, T_2, T_3 as shown (which of course is not guaranteed by simply writing the source code like that).

31. Consider the code below. Can it be transformed so that a new outer loop can run in pararallel? Why or why not?

```
for (i = 1; i <= 10000; i += 1) {
        for (j = 1; j <= 10000; j += 1) {
                a[i][j] = a[i-1][j] * 2;
                b[i][j] = b[i+1][j] * 3;
        }
}
```

32. (1p) Who invented sequential consistency?

33. (1p) In which decade were both of weak ordering and transactional memory invented ?