Försättsblad till skriftlig tentamen vid Linköpings Universitet

	(fylls i av ansvarig)
Datum för tentamen	23 aug 2012
Sal	TER1
Tid	14-18
Kurskod	TDDD56
Provkod	TEN1
Kursnamn/benämning	Multicore and GPU Programming
Institution	IDA
Antal uppgifter som ingår i tentamen	8
Antal sidor på tentamen (inkl. försättsbladet)	5
Jour/Kursansvarig	Christoph Kessler, I. Ragnemalm
Telefon under skrivtid	0703-666687 070-6262628
Besöker salen ca kl.	16:00
Kursadministratör (namn + tfnnr + mailadress)	Gunilla Mellheden, 013-282297 e. 0705-979044, gunme@ida.liu.se
Tillåtna hjälpmedel	Engelsk ordbok, miniräknare
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	

Linköpings universitet
IDA Department of Computer and Information Sciences
Prof. Dr. Christoph Kessler

TENTAMEN / *EXAM*

TDDD56
Multicore and GPU Programming

23 aug 2012, 14:00–18:00 TER1

Jour: Christoph Kessler (070-3666687, 013-282406), visiting ca. 16:00.
Ingemar Ragnemalm (070-6262628)

Hjälpmedel / *Admitted material:*

– Engelsk ordbok / *Dictionary from English to your native language*

General instructions

- This exam has 8 assignments and 4 pages, including this one.
  Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet of paper for each assignment, because they will be corrected by different persons.
  Sort the pages by assignment, number them consecutively and mark each one on top with your exam ID and the course code.
- You may answer in either English or Swedish. **English is preferred** because not all correcting assistants understand Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- Grading: U, 3, 4, 5. The preliminary threshold for passing is 20 points.

  Students in international master programs and exchange students will receive ECTS grades. Due to the anonymization of written exam correction, ECTS grades will be set by one-to-one translation from swedish grades (5=A, 4=B, 3=C, U=FX), according to the regulations by Linköping university.

1. (7 p.) **Multicore Architecture Concepts**

   (a) There are three main technical limits that prevent a further significant increase of single-thread performance, which finally led to the development of multicore architectures. Name and explain two of these limits. (2p)

   (b) Define and explain the following technical terms:

      i. SIMD instructions
      ii. Symmetric multiprocessor (SMP)
      iii. Hardware multithreading
      iv. Bus snooping
      v. Sequential (memory) consistency

   (Remember that an example is not a definition. Be general and thorough.) (5p)

2. (6 p.) **Non-blocking Synchronization**

   (a) Name 2 problems of lock-based synchronization that are removed by non-blocking synchronization. (1p)

   (b) A bounded push buffer is a data structure consisting of an array B dimensioned for a capacity of N elements and an unsigned integer counter *top*, initially 0, which indicates the next free insert position at index *top*. B and *top* reside in shared memory.

   A number of threads push elements into the buffer concurrently. Use the atomic *compare-and-swap* (CAS) instruction to implement the operation *push*, such that a call *push*(e) atomically inserts an element e at the current top position. If a thread tries to push to a full buffer, *push* returns an error code. The application guarantees that not more than *MAXINT*-1 push operations will be performed in total, so that the counter *top* will never overflow.

   Use C or equivalent pseudocode. Make clear which of your variables are shared and which are thread-local.

   Explain your code!

   In particular, explain what CAS does and what its parameters mean.

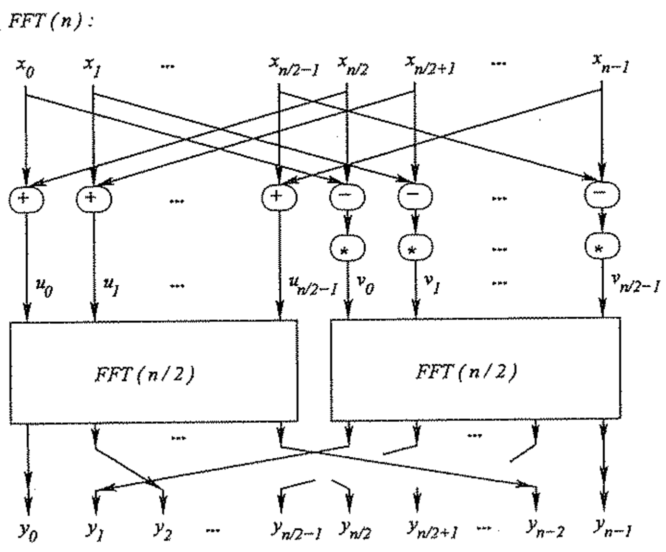
   Explain why, with your solution, concurrent *push* operations can not lead to "gaps" or lost entries in the buffer. (4p)

   (c) What is the *ABA problem* (in the context of CAS operations)? (1p)

3. (8 p.) **Design and Analysis of Parallel Algorithms**

   (a) Give an example of a *speedup anomaly* that may occur on a multicore computer, and explain its technical cause. (1p)

```
Algorithm FFT ( array $x[0..n-1]$ )
        returns array $y[0..n-1]$
{
    if $n = 2$ then
        $y[0] \leftarrow x[0] + x[1]$;  $y[1] \leftarrow x[0] - x[1]$;
    else
        allocate temporary arrays $u, v, r, s$
            of $n/2$ elements each;
        for $l$ in { 0.. n/2-1} do
            $u[l] \leftarrow x[l] + x[l + n/2]$;
            $v[l] \leftarrow w^l * (x[l] - x[l + n/2])$;
        od
        $r \leftarrow FFT$ ( $u[0..n/2-1]$ );
        $s \leftarrow FFT$ ( $v[0..n/2-1]$ );
        for $i$ in { 0.. n-1} do
            if $i$ is even then $y[i] \leftarrow r[i/2]$ fi
            if $i$ is odd then $y[i] \leftarrow s[(i-1)/2]$ fi
        od
    fi
    return $y[0..n-1]$
}
```



FFT(n):
x_0 x_1 ... $x_{n/2-1}$ $x_{n/2}$ $x_{n/2+1}$ ... x_{n-1}
+
−
*
u_0 u_1 ... $u_{n/2-1}$ v_0 v_1 ... $v_{n/2-1}$
FFT(n/2)
FFT(n/2)
y_0 y_1 y_2 ... $y_{n/2-1}$ $y_{n/2}$ $y_{n/2+1}$ ... y_{n-2} y_{n-1}

Figure 1: The sequential FFT algorithm.

(b) The Fast-Fourier-Transform (FFT) is a (sequential) algorithm for computing the Discrete Fourier Transform of an array x of n elements (usually, complex numbers) that might represent sampled input signal values, and a special complex number ω that is a nth root of unit, i.e., $w^n = 1$. The result y is again an array of n elements, now representing amplitude coefficients in the frequency domain for the input signal x. Assume for simplicity that n is a power of 2. A single complex addition, subtraction, multiplication and copy operation each take constant time.

Figure 1 shows the pseudocode of a recursive formulation of the FFT algorithm and gives a graphical illustration of the data flow in the algorithm.

i. Which fundamental algorithmic design pattern is used in the FFT algorithm? (0.5p)

ii. Identify which calculations could be executed in parallel, and sketch a parallel FFT algorithm for n processors in pseudocode (shared memory). (0.5p)

iii. Analyze your parallel FFT algorithm for its *parallel execution time*, *parallel work* and *parallel cost* (each as a function in n, using big-O notation) for a problem size n using n processors. *(A solid derivation of the formulas is expected; just guessing the right answer gives no points.)* (2.5p)

iv. Is your parallel FFT algorithm *work-optimal*? Justify your answer (formal argument). (1p)

v. How would you adapt the algorithm to work for a fixed number $p < n$ of processors? What will then be its parallel time with p processors? (1.5p)
How would you choose p ideally to make the adapted algorithm asymptotically cost-optimal? (1p)

[In case of questions about the following 3 assignments, ask I. Ragnemalm in the first hand.]

4. (5 p.) **GPU Algorithms and Coding**

   Describe in pseudocode and figures how an optimized matrix multiplication works on the GPU.

5. (5 p.) **GPU Conceptual Questions**

   (a) List three different kinds of GPU memory and describe for each their characteristics in terms of performance, usage and accessibility. CUDA terminology is assumed, please note if you use OpenCL terminology. (3p)

   (b) If you have a modern GPU with 512 cores, how much speedup can you expect to get? Yes, it depends on the algorithms, but in what way? Make a reasonable assessment and back that with hardware and algorithm based arguments. (2p)

6. (5 p.) **GPU Quickies**

   (a) In graphics, data is always input as geometrical shapes. What geometry is usually used for shader-based GPU computing?

   (b) What concept in CUDA corresponds to a thread processor in the GPU architecture? (1p)

   (c) How can pinned (page-locked) CPU memory improve performance? (1p)

   (d) List three different kinds of hardware that OpenCL runs on. (Similar systems by different vendors count as one.) (1p)

   (e) For what kind of problems are shader-based GPU computing most suitable? Give one specific example. (1p)

7. (2 p.) **Optimization and Parallelization**

   (a) Given its data dependence graph, formulate a sufficient (safe) condition whether a sequential loop can be executed in parallel. (1p)

   (b) Why is it, in general, so hard for C/C++ compilers to statically analyze a given sequential legacy program and parallelize it automatically? (1p)

8. (2 p.) **Parallel algorithmic design patterns and High-level parallel programming**

   Give two main advantages and two main drawbacks of *skeleton programming*.

Good luck!