263-2800-00L
Design of Parallel and High Performance Computing

Departement Informatik
ETH Zürich

Fall 2017: February 14th, 2018 9:00am - 11:00am. Room: HG E 3

Last name, First name:______________________________
Student number:___________________________________

GENERAL GUIDELINES AND INFORMATION

1. Start this exam only after the examiner has announced that the examination can begin. You have 2 hours (120 min).
2. Be sure to provide your name. **Do this first so that you don't forget!** Print your name! Provide your student ID (first 8 digits).
3. You should write your answers directly on the test sheet. Use the space provided. If you need more space your answer is probably too long.
4. Clarity of presentation is essential and does influence the grade. **Please write or print legibly.** State all assumptions that you make in addition to those stated as part of a question.
5. Write your answers in English.
6. You are free to leave whenever you have finished the exam. But to avoid disturbing your colleagues, nobody can leave the room during the last 30 minutes.
7. With your signature below you certify that you solved these problems on your own, that you turn in your solution, and that there were no environmental or other factors that disturbed you during this exam or that diminished your performance.

Signature:______________________

Problem	Points	Score	Problem	Points	Score
1	23		4	22	
2	20		5	30	
3	10		6	15	
Total				120	

1) Caches [23 points]

a) State the difference between spatial and temporal locality. (2pt)

b) Assume a system with a 4KiB byte-addressable memory and a 2-way associative LRU cache with a total size of 256B and cache blocks of 32B. The addresses are in the (tag, set, offset) format. A program makes a sequence of accesses to an array of doubles starting at address `0x000`. The size of a double is 8 bytes. Table 1 reports the sequence of such accesses (one per row). (6pt)

- Fill the Table 1 by specifying, for each access, the tag/set/offset of the accessed memory location, and if it leads to a cache miss.
- Use Table 2 to show the cache state after the last access: write the array indices that are stored in each block.

Address	Tag	Set	Offset	Miss?
`0x050`	0	2	16	Yes
`0x028`				
`0x158`				
`0x0E0`				
`0x040`				
`0x080`				

Table 1

	Block 0				Block 1			
Set 0								
Set 1								
Set 2	8	9	**10**	11				
Set 3								

Table 2

c) State the difference between cache coherence and memory consistency. (2pt)

d) What is the difference between write-through and write-back? (2pt)

e) Assume a machine with 32bit addresses and 4 processors with directly-mapped L1 caches. Each cache is 4MiB in size, with 128B wide cache lines. MESI is used as cache-coherency mechanism. The memory is byte-addressable.

Tag/Set/Offset: How many bits of the address are used as tag-, set-, and offset-bits? Assume the offset bits are the least significant ones, while the tag bits are the most significant. (3pt)

Cache Coherence: For the following sequence of instructions, list the cacheline index (CL) that they target, the new state of the cacheline, and the bus signal. Use the provided table, filling only the fields that change their values after an instruction. Assume that initially all the cachelines are invalid (I). (8pt)

Inst.	Processor 1		Processor 2		Processor 3		Processor 4		Bus Signal
	CL	State	CL	State	CL	State	CL	State	
`P1: R(0xFF001001)`	32	E							BusRd
`P2: R(0xFF001034)`									
`P3: W(0xFF040023)`									
`P1: W(0xFF001023)`									
`P3: R(0xFF040078)`									
`P4: W(0xFF040000)`									

2) Memory Models [20 points]

a) What is a memory model? Why is it useful? (2pt)

b) For the following executions traces, either provide a sequentially consistent interleaving or show that this does not exist. *W(var, val)* indicates the write of the value *val* to the variable *var*. *R(var, val)* indicates the read of the value *var* from the variable *var*. Assume that all the variables are initially set to 0. Justify your answer! (10pt)

1) P1: R(r, 1);
   P2: W(r, 1); R(r, 2)
   P3: W(r, 2)

2) P1: R(r, 1); W(r, 2)
   P2: R(r, 2); W(r, 1)

3) P1: W(x, 2); W(y, 0)
   P2: R(y, 0); W(x, 2)
   P3: R(y, 2); R(y, 0)

4) P1: W(x, 1); W(x, 2); R(y, 2); W(x, 3)
   P2: R(x, 2); R(x, 1)
   P3: W(y, 1); R(x, 2)
   P4: W(y, 2)

5) P1: R(x, 1); W(z, 2); W(z, 1)
   P2: W(x, 1); R(z, 1)
   P3: R(z, 2); W(z, 1)

c) Give an example where write buffers invalidate sequential consistency. (2pt)

d) Consider the following instructions executed by two processors (P0, P1). The `print()` function outputs the integer passed as parameter. Provide an output of both processors that is legal in the x86 memory model but not in sequential consistency. Why is that possible? Provide the associated trace using the same format as in Exercise 2.b. (6pt)

```
P0:                 P1:
  store a, 1          store b, 1
  load b              load a
  print(b)            print(a)
```

3) Linearizability [10 points]

a) In the below history two threads `A`, `B` operate on a data structure `x`.

- Assuming that `x` is a FIFO queue, is the history linearizable?
- Assuming that `x` is a stack (LIFO), is the history linearizable?

Justify your answers. The methods `insert`/`extract` correspond to `enqueue`/`dequeue` if `x` is a FIFO queue, and to `push`/`pop` if it is a stack. (4pt)

```
A:x.insert 2
B:x.insert 1
B:void
B:x.extract
A:void
B:1
A:x.extract
A:2
```

b) Can a non-complete history be linearizable? Justify your answer. (2pt)

c) Can a history be non-linearizable but still be sequentially consistent? If yes, provide an example, otherwise justify your answer. (4pt)

4) Locks, Lock-free and Wait-free [22 points]

a) Consider the two-threads lock described by the below code. Using sequential consistency, prove or disprove that this lock provides mutual exclusion. Assume that `lock` and `flag[0], flag[1]` are initialized to 0 and `gettid()` returns 0 for the first thread and 1 for the second. (6pt)

```
volatile int lock;
volatile int flag[2];

int me = gettid(); //my thread ID (0 or 1)
int peer = 1-me;

flag[me] = 1
while (lock==1 && flag[peer]==1){;}
lock=1;

/* critical section */

lock=0;
flag[me] = 0;
```

b) Define the starvation- and deadlock-free properties for locks. Does the above lock satisfy these properties? Justify your answer. (4pt)

c) Briefly describe the Lamport's Bakery Algorithm. Is it starvation free? Justify your answer. (3pt)

d) Can some threads overtake others in the Filter lock for an arbitrary number of times? If yes, which property is this lock violating? Justify your answer. (3pt)

e) Assume a linked list `L` is shared among $n > 2$ threads.

- State one advantage and one disadvantage of using coarse- and fine-grained locking schemes. (3pt)

	Advantage	**Disadvantage**
Coarse-grained		
Fine-grained		

- The `L.contains(x)` method returns true if `x` is in the linked list, false otherwise. Can this method be implemented in a wait-free manner? If so, how? (3pt)

5) Performance Models [30 points]

a) Assume a search engine receives a search query every 0.1s. It takes 0.4s to serve a query.

- If a query needs 8KiB of memory to be stored, how much memory does the search engine need to buffer the incoming search queries? (3pt)

- To serve a query, the search engine spends the 40% in a serial computation. Now assume that we can increase the parallelism degree of our system by a factor of 6. Use the Amdhal's law to give an upper bound on the expected speedup of the parallelized engine. (3pt)

- Is it conceivable that the bound you computed above is exceeded? If so, explain why is that possible. (2pt)

b) Assume a single-core system with an LRU data cache, a peak performance of $\pi = 4$ single precision floating point operations/cycle, and a memory bandwidth of $\beta = 8$ bytes/cycle.

- What is the ridge point in the roofline point of the above described system? (2pt)

- Consider the following function operating on a matrix `A` of n^2 floats. `A` is stored in row-major order. Assume that the cache size γ is much smaller than n ($\gamma \ll n$) and that a cache block has size equal to 8 floats (a float is 8 bytes). No elements of A are initially in cache (i.e., cold cache). What is the operational intensity of the following code? Is it compute or memory bound on this system? Justify your answer. (4pt)

```
void foo(float A[n][n]){
  for (int j=0; j<n; j++){
    for (int i=1; i<n; i++){
      A[0][j] = A[0][j] + A[i][j];
    }
  }
}
```

c) Assume a program with an operational intensity of $I = \Theta(\sqrt{\gamma})$ that is balanced with respect to a given architecture (single-core). If the peak performance (π) doubles every 2 years and the memory bandwidth (β) doubles every 4 years, with which yearly rate does the cache size need to increase in order to keep the balance? (4pt)

d) A process $i = 0$ needs to broadcast k messages of s bytes to the remaining $n - 1$ processes (i.e., n is the total number of processes). The processes communicate in a linear fashion: i.e., process $i = 0$ sends to $i + 1$, processes $1 < i < n - 1$ receive from $i - 1$ and send to $i + 1$. and process $n - 1$ only receives from $n - 2$. Express the total cost of this broadcast as function of the latency α and the cost per byte β. Justify your answer. (4pt)

e) Given a binary string (i.e., made by 0s and 1s), provide a CRCW PRAM algorithm that computes the longest sequence of 1s in O(1) time. Assume that multiple processes can accumulate (e.g., min, max, sum) values in a memory location at the same time. (8pt)

6) Sampler [15 points]

a) What is the difference between strong and weak scalability? (2pt)

b) When does false sharing occur? How can it be limited? (2pt)

c) What is the difference between program order and visibility order? (2pt)

d) What is the difference between the shared memory model and the message passing model? (2pt)

e) Can a test-and-set lock result in poor performance? If yes, how can this be improved? (2pt)

f) What is the semantic of a memory barrier? (2pt)

g) What does it mean for a class to solve the n-thread consensus? State one requirement that a consensus protocol has to satisfy. (3pt)

This page is intentionally left blank