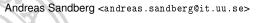
Introduction to Lab 2



Division of Computer Systems Dept. of Information Technology Uppsala University

2010-09-28

What is lab 2? ... or what is consistency, and who cares anyway?

The purpose of this assignment is to give insights into:
how to program muli-processors
why synchronization is needed
how synchronization may be implemented
how memory consistency affects program behavior
how heavy-weight synchronization can be avoided with atomic instructions

2 Avdark'10 | Introduction to Lab 2

What is a process?

A process contains the following:

- A set of memory mappings (heap, code, etc)
- Environment variables
- Signal handlers
- A list of open file descriptors (files, devices, network connections, etc)
- UID/GID/PID and some more TLAs¹
- One or more *threads*.

¹Three Letter Abbreviations ③ Avdark'10 Introduction to Lab 2

What is a thread?

A thread is an independent flow of control within a process

4 Avdark'10 | Introduction to Lab 2

What is a thread?

Why do we need synchronization?

A thread contains:

- A set of registers. Including:
 - Program Counter
 - Stack Pointer
- A scheduling priority

if (balance > amount) balance = balance - amount;

What happens if multiple threads execute the code above at the same time?

5 Avdark'10 Introduction to Lab 2

6 Avdark'10 Introduction to Lab 2

How do we update shared state correctly? Bringing order to chaos

Two common approaches:

- Use critical sections
 - Heavy-weight approach.
- · Operating systems usually provide an API to do this. Atomic instructions
- - Relatively light-weight compared to above method.
 - Serializes memory accesses on the system. May need to write assembler or use compiler
 - pragmas/intrinsics.

x86 memory ordering

Consistency in the x86

- Defined in Volume 3A (System Programming guide) of the Intel® 64 and IA-32 Architectures Software Developer's Manual.
- Memory ordering depends on access type: Processor Ordering for "normal" memory operations. Very
 - similar to Total Store Order. Total Lock Order for instructions with the lock prefix.
 - Atomic instructions behave as if the system implemented Sequential Consistency.

(7) Avdark'10 | Introduction to Lab 2

(8) Avdark'10 | Introduction to Lab 2

What is Processor Ordering? An incomplete description

In an individual processor:

- Writes are not reordered with other writes.
- Reads may be reordered with older writes to different locations.

In a multi-processor system:

- Writes by a single processor are observed in the same order by all processors.
- Writes from an individual processor are not ordered with respects to writes from other processors.
- Memory ordering obeys causality.
- Any two stores are seen in a consistent order by processors other than those performing the store.

9 Avdark'10 | Introduction to Lab 2

Forcing memory order

It is possible to force memory ordering using memory fences.

Atomic instructions in the x86

Assembler:

mfence

GCC intrinsics:

__builtin_ia32_mfence();

10 Avdark'10 | Introduction to Lab 2

What is an atomic instruction?

 Atomic instructions perform their action as one unit without exposing intermediate state

Atomic instructions in the x86

- Naturally aligned loads and stores (up to 64 bits) are generally atomic, i.e. it's impossible to read a half-updated word.²
- Most instructions accessing memory can be turned into atomic instructions by adding a lock prefix.

² They still adhere to	Processor	Ordering an	d not	Total Lock	Order.
11 Avdark'10 Introduction to Lab 2					

Simple examples

Incrementing a number: Iock inc 0x0(%eax)

Decrementing a number: **lock dec** 0x0(%**eax**)

12 Avdark'10 | Introduction to Lab 2

Introduction	Consistency in the x86	Atomic instructions in the x86	Dekker's algorithm	Summary
Excha	nge			
xchg	g %eax , 0x0(%e	ebx)		
	-	lue in memory location	n 0x0(%ebx) wit	h
	the value in %eax Always atomic, the	e lock prefix is option	al	
3 Avdark'10	Introduction to Lab 2			

lock cmpxchg %ebx, 0x0(%ecx)

Compare and exchange

tomic instructions in the x86

Uses %eax as an implicit operand

Is %eax is equal to 0x0(%ecx)? true Write %ebx into 0x0(%ecx) false Write 0x0(%ecx) into %eax

14 Avdark'10 | Introduction to Lab 2

oduction Consistency in the x86 Atomic instructions in the x86

Background ... or who is this Dekker guy anyway?

- Dekker's algorithm solves the critical section problem for 2 threads without fancy hardware support.
- Attributed to the Dutch mathematician Theodorus J. Dekker in a manuscript from 1965 by Edsger W. Dijkstra.

Introduction Consistency in the x86 Atomic instructions in the x86 Dekker's algorithm Summary

The algorithm

```
\begin{array}{l} flag_i \leftarrow True \\ \textbf{while } flag_j \ \textbf{do} \\ \textbf{if } turn \neq i \ \textbf{then} \\ flag_i \leftarrow False \\ \textbf{while } turn \neq i \ \textbf{do} \\ Do nothing or sleep \\ \textbf{end while} \\ flag_i \leftarrow True \\ \textbf{end while} \\ \textbf{Do critical work} \\ turn \leftarrow j \\ flag_i \leftarrow False \end{array}
```

15 Avdark'10 | Introduction to Lab 2

16 Avdark'10 | Introduction to Lab 2

Introduction	Consistency in the x86	Atomic instructions in the x86	Dekker's algorithm	Summary
Limitat	ions			

· Requires memory barriers to force the processor to order

In the lab What you'll be doing (hopefully)

You will:

- · Implement Dekker's algorithm and use memory barriers to make it run correctly on x86.
- · Implement a simple algorithm using different types of atomic instructions instead of critical sections
- · Do performance studies for different types of
- implementation strategies

Bonus Implement queue locks using atomic instructions

Complete lab manual on the course homepage³

³http://www.it.uu.se/edu/course/homepage/avdark/ht10 (18) Avdark'10 Introduction to Lab 2

(17) Avdark'10 Introduction to Lab 2

Only works for two threads.

• ... but we don't care.

accesses.

Does not work with weak consistency models.



Groups:

- Prep. Room 1549, now-17:00
 - A 2010-09-30, Room 1549, 08:15–12:00
 - B 2010-09-30, Room 1549, 13:15–17:00
- C 2010-10-01, Room 1549, 08:15-12:00
- Deadline: See course homepage



Summary And remember.

Thou shalt make thy program's purpose and structure clear to thy fellow man by using the One True Brace Style, even if thou likest it not, for thy creativity is better used in solving problems than in creating beautiful new impediments to understanding.⁴

⁴http://www.lysator.liu.se/c/ten-commandments.html (20) Avdark'10 Introduction to Lab 2

(19) Avdark'10 Introduction to Lab 2