Homework 9

Graded

2 Days, 16 Hours Late

Student

Vidit Dharmendra Pokharna

Total Points

90 / 100 pts

Question 1

(no title) **90** / 100 pts

1.1 **(no title)** **20** / 20 pts

- ✔ **+ 10 pts** Mention shared data structure used by multiple threads
- ✔ **+ 10 pts** Adopt any synchronization primitives (e.g. mutex lock, barrier)
- **– 5 pts** Insufficient explanation but on the right track
- **+ 0 pts** Incorrect

1.2 **(no title)** **15** / 20 pts

- ✔ **+ 10 pts** Pthread can call the OS to do the synchronization and scheduling but there isn't an abstraction layer below the kernel to handle synchronization for it
- ✔ **+ 10 pts** Need an atomic read&write instruction (e.g. Test-And-Set)
- ✔ **– 5 pts** Insufficient explanation but on the right track
- **+ 0 pts** Incorrect

1.3 **(no title)** **20** / 20 pts

- ✔ **+ 10 pts** TLB entries must reflect the corresponding page table entries, and all processors' TLBs should be consistent
- ✔ **+ 10 pts** Partnership of hardware and software, evict the entry in TLB (hardware), let the OS running on other CPUs evict the corresponding entry (if present) using software interrupts (software)
- **– 5 pts** Insufficient explanation but on the right track
- **+ 0 pts** Incorrect

1.4 **(no title)** **20** / 20 pts

- ✔ **+ 20 pts** Implement some cache coherence protocol (e.g. write-invalidate, write-update)
- **– 5 pts** Insufficient explanation but on the right track
- **+ 0 pts** Incorrect

1.5 **(no title)** **15** / 20 pts

- ✔ **+ 20 pts** Private stack for each thread; shared heap, global, code among threads
- ✔ **– 5 pts** Insufficient explanation but on the right track
- **+ 0 pts** Incorrect

Q1
100 Points

Give a brief explanation to answer each of these questions. Two- or three-sentence answers are quite adequate.

Q1.1
20 Points

What might make a C library function not thread-safe? How might one make such a function thread-safe?

If a function uses global or static data, it can lead to race conditions when accessed by multiple threads concurrently. The solution is to have thread-safe wrappers for library calls.

Q1.2
20 Points

Why can't we implement mutual exclusion in the kernel the same way we implement it in pthreads; i.e., why can't we use the scheduler to block a thread in the kernel while it waits for access to a critical section?

Using the scheduler to block a thread in the kernel while waiting for access to a critical section is generally avoided because it can lead to significant inefficiencies and complexity. Blocking a thread at the kernel level involves context switching and scheduler overhead, which can be resource-intensive, especially for short critical sections. Additionally, kernel-level operations must account for concerns like interrupt handling and hardware interactions, which can make simple blocking strategies used in user-space (like in pthreads) inadequate or overly simplistic for kernel-level synchronization.

Q1.3
20 Points

Explain the reasons for a TLB Shootdown. How is it implemented?

A TLB Shootdown is necessary in SMP to maintain consistency between the caches of different CPUs when a page is replaced. When the operating system replaces a page in memory, it must invalidate the corresponding TLB entry on the CPU where the replacement occurs (a standard practice even in uniprocessor systems) and also inform other CPUs to evict their TLB entries for that page. This is typically implemented through an inter-processor interrupt, which is sent to remote cores, prompting their interrupt handlers to invalidate and then update the TLB entries for the affected memory regions or pages. This process involves waiting for these handlers not only to invalidate their TLBs but also to acknowledge completion before reloading the TLB with correct mappings.

Q1.4
20 Points

How do we ensure that all threads have an identical view of memory in an SMP in which each CPU has its own cache?

SMP systems use hardware-based cache coherence protocols to ensure consistency between the caches of different CPUs. The operating system worries about the consistency of TLBs to ensure that all threads have the same view of the shared process address space.

Q1.5
20 Points

What is a "cactus stack"?

A cactus stack is the stack layout of a multithreaded process.