**+--------------------+**

**| CSE 421/521 |**

**| PROJECT 1: THREADS |**

**| DESIGN DOCUMENT |**

**+--------------------+**

**---- GROUP ----**

>> Fill in the names and email addresses of your group members.

ARUN SURESH <arunsure@buffalo.edu>

HARIPRASTH PARTHASARATHY <hparthas@buffalo.edu>

PRAJAL JAIN <@buffalo.edu>

**---- PRELIMINARIES ----**

>> If you have any preliminary comments on your submission, notes for the

>> TAs, or extra credit, please give them here.

>> Please cite any offline or online sources you consulted while

>> preparing your submission, other than the Pintos documentation, course

>> text, lecture notes, and course staff.

**ALARM CLOCK**

**===========**

**---- DATA STRUCTURES ----**

>> A1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

**---- ALGORITHMS ----**

>> A2: Briefly describe what happens in a call to timer\_sleep(),

>> including the effects of the timer interrupt handler.

Inside the timer\_sleep() method, we calculate the time at which the thread needs to be woken up and push the thread to a wait list. Then we call sema\_down() to block the thread.

>> A3: What steps are taken to minimize the amount of time spent in

>> the timer interrupt handler?

The wait list where all the threads are put to sleep is sorted on based on the time to be woken up. The first entry in the list is the thread that needs to be woken up. This saves amount of time sent by timer interrupt handler searching for the thread to be woken up.

**---- SYNCHRONIZATION ----**

>> A4: How are race conditions avoided when multiple threads call

>> timer\_sleep() simultaneously?

>> A5: How are race conditions avoided when a timer interrupt occurs

>> during a call to timer\_sleep()?

**---- RATIONALE ----**

>> A6: Why did you choose this design? In what ways is it superior to

>> another design you considered?

The design we have chosen has many advantages. The amount of time spent by timer interrupt handler is very minimal as the list is already sorted based on the time to be woken up. The timer\_sleep () method is synchronized void of race conditions when multiple threads call timer\_sleep (). We chose to have a separate list for waiting threads sorted based on the time to be woken up.

**PRIORITY SCHEDULING**

**===================**

**---- DATA STRUCTURES ----**

>> B1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

>> B2: Explain the data structure used to track priority donation.

>> Use ASCII art to diagram a nested donation. (Alternately, submit a

>> .png file.)

**---- ALGORITHMS ----**

>> B3: How do you ensure that the highest priority thread waiting for

>> a lock, semaphore, or condition variable wakes up first?

>> B4: Describe the sequence of events when a call to lock\_acquire()

>> causes a priority donation. How is nested donation handled?

>> B5: Describe the sequence of events when lock\_release() is called

>> on a lock that a higher-priority thread is waiting for.

**---- SYNCHRONIZATION ----**

>> B6: Describe a potential race in thread\_set\_priority() and explain

>> how your implementation avoids it. Can you use a lock to avoid

>> this race?

**---- RATIONALE ----**

>> B7: Why did you choose this design? In what ways is it superior to

>> another design you considered?

**ADVANCED SCHEDULER**

**==================**

**---- DATA STRUCTURES ----**

>> C1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

**---- ALGORITHMS ----**

>> C2: Suppose threads A, B, and C have nice values 0, 1, and 2. Each

>> has a recent\_cpu value of 0. Fill in the table below showing the

>> scheduling decision and the priority and recent\_cpu values for each

>> thread after each given number of timer ticks:

timer recent\_cpu priority thread

ticks A B C A B C to run

----- -- -- -- -- -- -- ------

0

4

8

12

16

20

24

28

32

36

>> C3: Did any ambiguities in the scheduler specification make values

>> in the table uncertain? If so, what rule did you use to resolve

>> them? Does this match the behavior of your scheduler?

>> C4: How is the way you divided the cost of scheduling between code

>> inside and outside interrupt context likely to affect performance?

**---- RATIONALE ----**

>> C5: Briefly critique your design, pointing out advantages and

>> disadvantages in your design choices. If you were to have extra

>> time to work on this part of the project, how might you choose to

>> refine or improve your design?

>> C6: The assignment explains arithmetic for fixed-point math in

>> detail, but it leaves it open to you to implement it. Why did you

>> decide to implement it the way you did? If you created an

>> abstraction layer for fixed-point math, that is, an abstract data

>> type and/or a set of functions or macros to manipulate fixed-point

>> numbers, why did you do so? If not, why not?

**SURVEY QUESTIONS**

**================**

Answering these questions is **optional**, but it will help us improve the

course in future quarters. Feel free to tell us anything you

want--these questions are just to spur your thoughts. You may also

choose to respond anonymously in the course evaluations at the end of

the quarter.

>> In your opinion, was this assignment, or any one of the three problems

>> in it, too easy or too hard? Did it take too long or too little time?

>> Did you find that working on a particular part of the assignment gave

>> you greater insight into some aspect of OS design?

>> Is there some particular fact or hint we should give students in

>> future quarters to help them solve the problems? Conversely, did you

>> find any of our guidance to be misleading?

>> Do you have any suggestions for the TAs to more effectively assist

>> students, either for future quarters or the remaining projects?

>> Any other comments?