+-------------------------------------------------------------+
|             CS39001: Operating Systems Lab                  |
|   Introduction to PintOS and improving threads in PintOS    |
|                      DESIGN DOCUMENT                        |
+-------------------------------------------------------------+

---- GROUP 31 ----

Animesh Jain  18CS10004 <animeshjain0019@gmail.com>
Abhinav Bohra 18CS30049 <abhinavbohra09@gmail.com>

---- PRELIMINARIES ----

>> If you have any preliminary comments on your submission, notes for the
>> TAs, or extra credit, please give them here.

In the test given in the build directory (make test command runs it), the simulation was not powering off properly (and hence giving timeout) because in the latest versions of qemu the shutdown port has been changed. Thus we had to add a new line in src/devices/shutdown.c. -> " outw (0xB004, 0x2000); "

>> 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.

Reference :
https://stackoverflow.com/questions/39805784/timeout-in-tests-when-running-pintos/45276093#45276093

## 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.

In pintos/src/threads/thread.h we added,

```
struct thread {
  /* some stuff */

  struct list_elem waitelem;     /* To be stored in the waiting_list */
```

```
  int64_t sleep_endtick;          /* The tick after which the thread should
awake (if the thread is in sleep) */

  /* more stuff */
};


---- ALGORITHMS ----

>> A2: Briefly describe what happens in a call to timer_sleep(),
>> including the effects of the timer interrupt handler.

In pintos/src/devices/timer.c we implemented the following algorithm,
(Pseudo-code)

timer_sleep (time_t x):
  Save waitelem to waiting_list
  Set current_thread->sleep_endtick to ( current_time + x )
  current_thread->status = BLOCKED

In pintos/src/threads/threads.c we implemented the following algorithm,
(Pseudo-code)

thread_tick (time_t x):
  /* do some stuff */
  traverse the waiting_queue and unblock all threads which have slept
enough


---- RATIONALE ----

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

1. Changing thread status to blocked prevents it from scheduling.
2. thread_tick is timer interrupt driven
3. Threrefore, it will unblock threads periodically
```