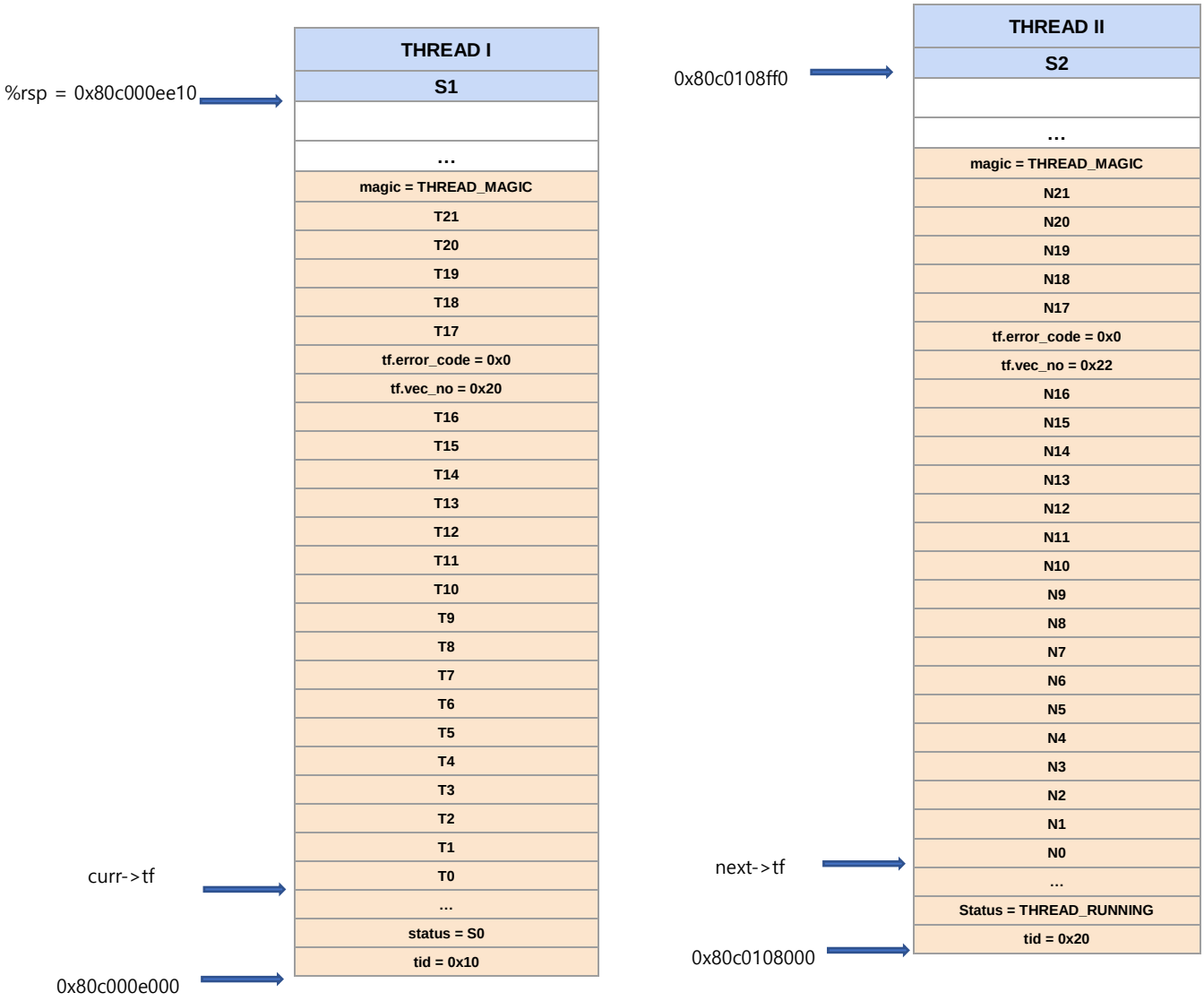
Operating Systems and Lab Midterm Homework 1

20200130 Yujun Kim

<Part 1 Context Switch>

Question 1) Show details of trap frames. What are values of %rip, %rsp?

1. Current



2. Location A

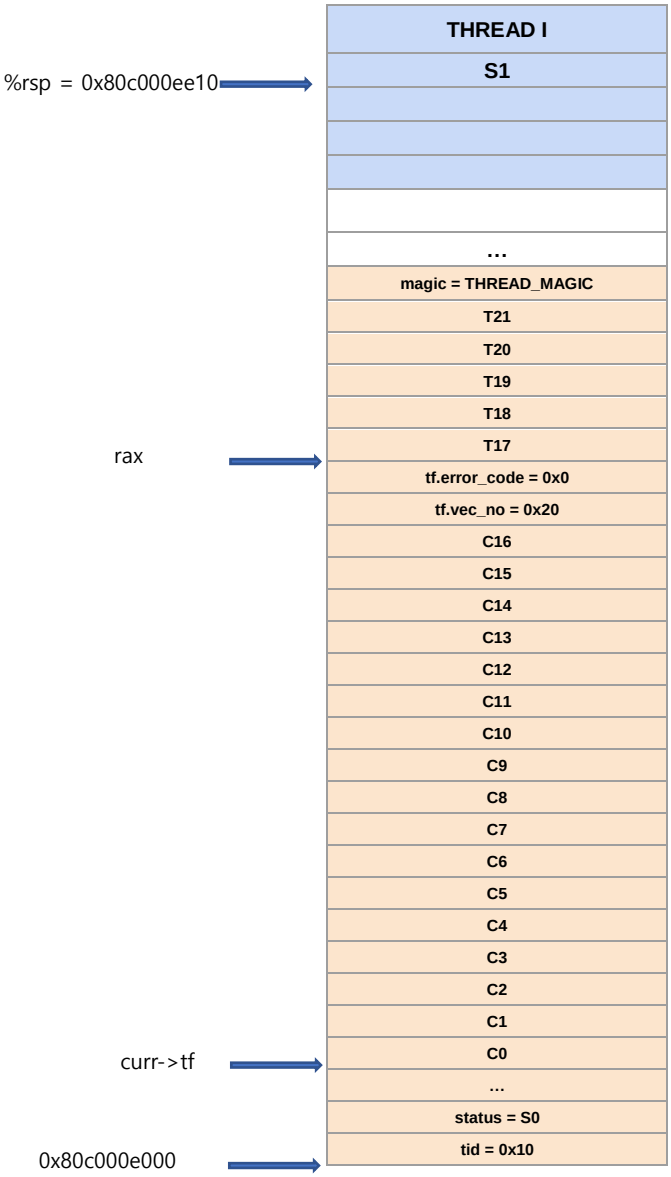
%rip = 0x800420743a

Thread2 same as current

3. Location B

%rip = 0x800420748d

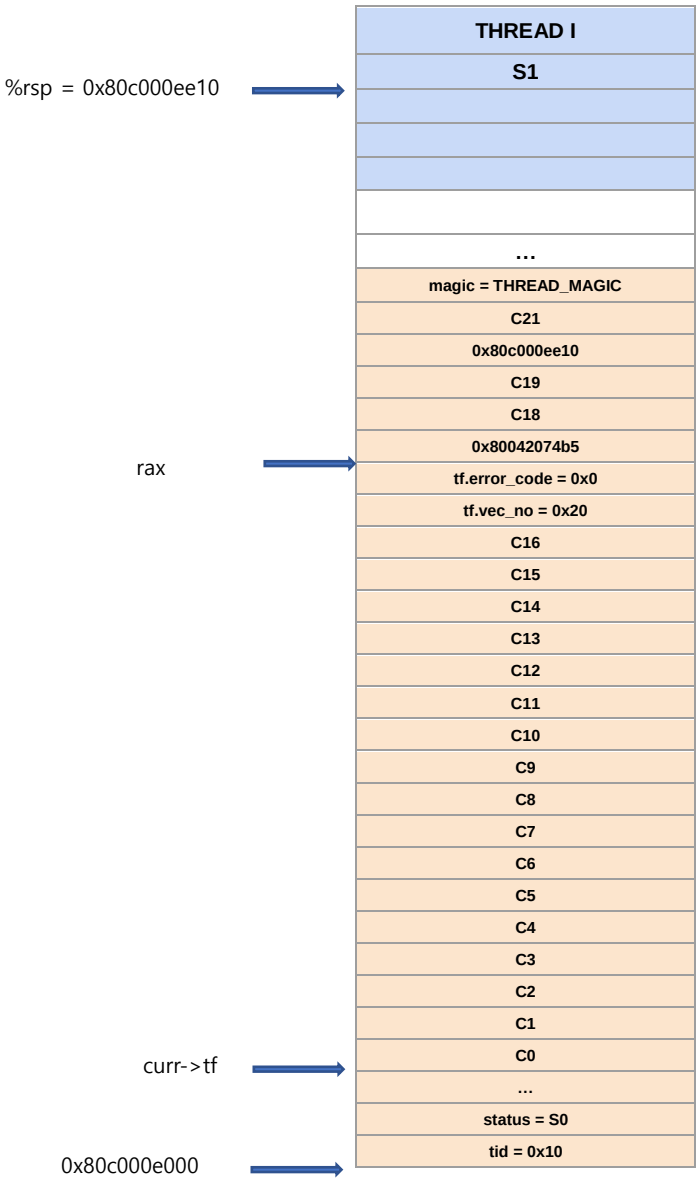
Thread2 same as current



4. Location C

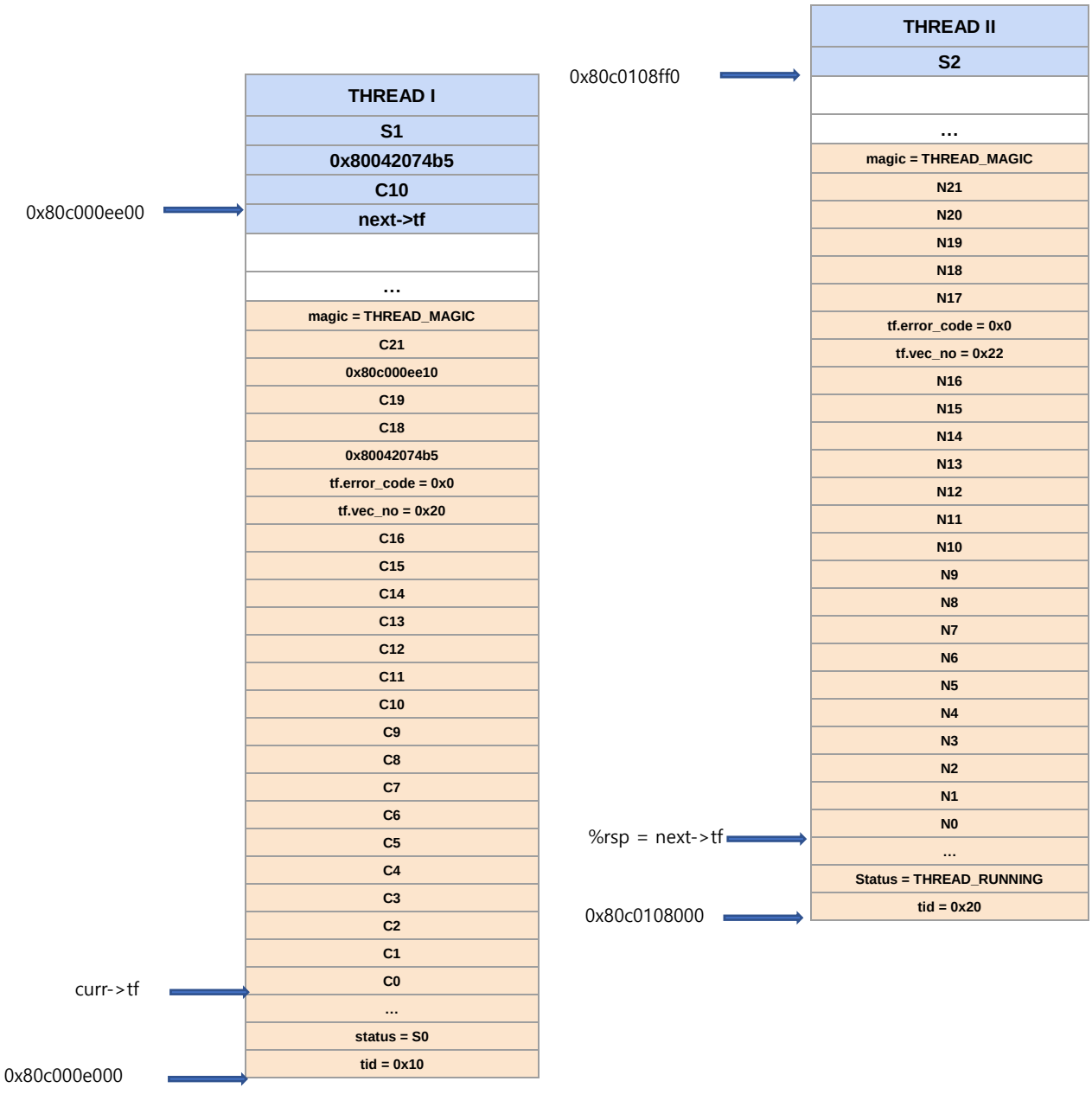
%rip = 0x80042074b0

Thread2 same as current



5. Location D

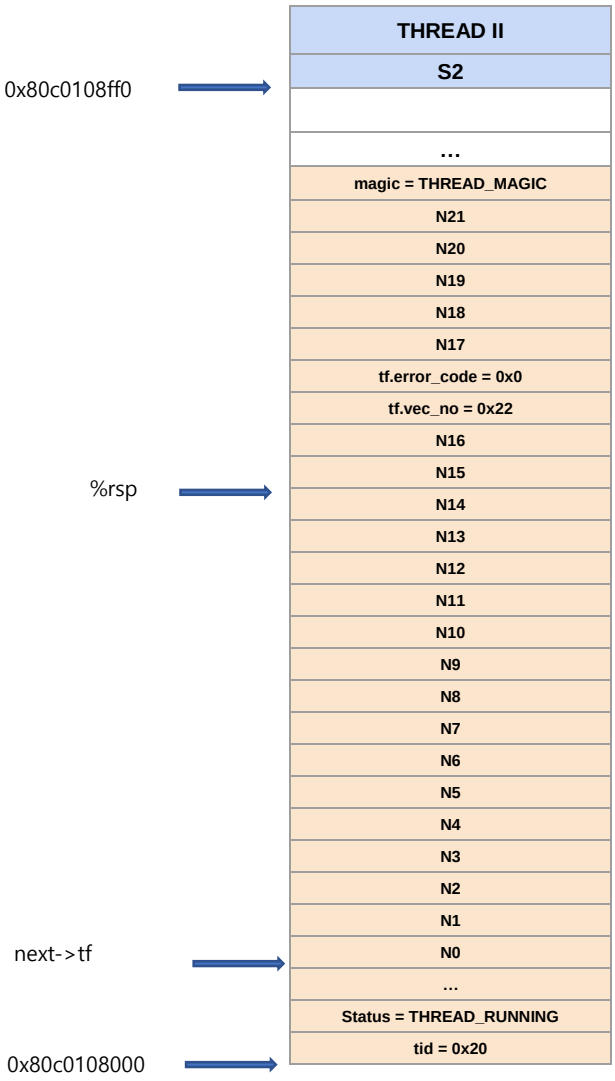
%rip = 0x80042073a3



6. Location E

%rip = 0x80042073ae

Thread1 same as D



Question 2) Explain the use of thread_ticks

thread_ticks count the ticks since the last yield happened. If the running thread does not yield for more than TIME_SLICE(= 4ticks), then thread_tick() which is called every tick sets yield_on_return to true. If so, interrupt handler makes thread to yield on the interrupt.

Then, thread_yield() calls schedule() and resets thread_ticks to 0, starting count for newly preempted thread.

<Part 2>

Question 1) Explain how to fill rest of `do_schedule()`

For thread in THREAD_DYING, we have to free the page given to that thread. Thus, when we call the `schedule()`, we make list of threads that are going to be destructed, and when the next `do_schedule()` is called free the corresponding pages. By doing so, we can keep only the pages that we should track. The pseudocode is

```
While list of thread that requested destruction is not empty
        Pop the front element in the list
        Palloc_free_page the element
```

Question2) Do the same for `schedule()`

Where `schedule()` has to check the status of current running thread and push back to the list that requested destruction if the state is THREAD_DYING. The pseudocode is

```
If current thread exists and its status is THREAD_DYING and current thread is not
the main thread,
        Assert that the current thread is not the thread to run
        Push back the current thread to the list of thread that requested
        destruction
```

Note the sequential flow in do_schedule() where we destroy the requested threads and then add the threads to be destroyed on the next do_schedule() call. This to prevent the current thread removing page of itself and get out of control.