



باسمه تعالی
سیستم‌های عامل
تمرین شماره ۵



۱. race condition را در قالب یک قطعه کد توضیح دهید.
۲. یک راه‌حل برای critical section چه نیازمندی‌هایی (شروطی) را باید برآورده کند؟ هر یک از آن‌ها را توضیح دهید.
۳. راه‌حل peterson برای مسئله‌ی critical section را توضیح داده و بررسی کنید از بین نیازمندی‌های حل این مسئله کدام‌ها را برآورده می‌کند.
۴. چرا disable کردن interruptها به منظور حفاظت از critical section در ماشین‌های multi processor ایده‌ی مناسبی نیست؟
۵. starvation و deadlock را توضیح دهید. تفاوت اصلی این دو با یکدیگر در چیست؟
۶. aging را توضیح دهید و بیان کنید که چگونه برای جلوگیری از starvation از آن استفاده می‌کنند.
۷. انواع کرنل را از نظر preemptive بودن بررسی کنید.
۸. توضیح دهید که busy waiting به چه معنی است و چگونه در یک سیستم با تنها یک پردازنده می‌تواند مشکل ساز باشد.
۹. توضیح دهید که در پیاده‌سازی semaphore ها با چه روشی اقدام به از بین بردن مشکل busy waiting شده‌است.
۱۰. بررسی کنید که اگر دستورات wait و signal برای semaphore ها atomic نباشند چه مشکلی ممکن است به وجود بیاید.
۱۱. الف) مشکل priority inversion را در قالب یک مثال توضیح دهید.
ب) یک راه‌حل برای حل مشکل priority inversion ارائه کنید.

۱۲. در مورد راه حل های سخت افزاری مسئله ی critical section به سوالات زیر پاسخ دهید:

الف) منظور از دستورات atomic چیست؟

ب) دستور test_and_set .atomic را مطابق زیر در نظر بگیرید:

```
boolean test_and_set(boolean *target) {  
    boolean rv = *target;  
    *target = true;  
  
    return rv;  
}
```

حال بررسی کنید که راه حل زیر که با استفاده از این دستور پیاده سازی شده کدام یک از شروط مورد نیاز برای حل یک مسئله ی critical section را برآورده می کند.

در این راه حل از دو data structure زیر استفاده شده که هر دوی آنها در ابتدا false مقدار دهی شدند.

boolean waiting[n];

boolean lock;

```
do {  
    waiting[i] = true;  
    key = true;  
    while (waiting[i] && key)  
        key = test_and_set(&lock);  
    waiting[i] = false;  
  
    /* critical section */  
  
    j = (i + 1) % n;  
    while ((j != i) && !waiting[j])  
        j = (j + 1) % n;  
  
    if (j == i)  
        lock = false;  
    else  
        waiting[j] = false;  
  
    /* remainder section */  
} while (true);
```

۱۳. یک سیستم با چند هسته‌ی پردازشی را در نظر بگیرید. در هر یک از سناریوهای زیر توضیح دهید استفاده از **spinlock** بهتر است یا **mutex lock**. در این حالت فرض کنید که پردازش‌هایی که متقاضی گرفتن **lock** در حالت **waiting** هستند تا زمان گرفتن قفل به حالت **sleep** در می‌آیند.

الف) قفل قرار است برای بازه‌های زمانی کوتاه گرفته شود.

ب) قفل قرار است برای بازه‌های زمانی بلند گرفته شود.

۱۴. نحوه‌ی پیاده‌سازی **monitor** با استفاده از **semaphore** را توضیح دهید.

۱۵. الف) مسئله‌ی **Dining-Philosophers** را با استفاده از **semaphore** حل کنید.

ب) دو راه حل ارائه دهید که در مسئله‌ی بالا **deadlock** رخ ندهد.