



Homework 4

Lectures 11, 12, 13, 14, 15

Operating Systems

Dr. Javadi

Spring 2023



بخش زمانبندی

۱- یک سیستم تک پردازنده‌ای با صف بازخورد چند سطحی (Multi-level Feedback Queue) را در نظر بگیرید. به صف اول الگوریتم چرخشی با برش زمانی معادل ۸ میکروثانیه داده شده است. به سطح دوم الگوریتم چرخشی با برش زمانی معادل ۱۶ میکروثانیه و سطح سوم به ترتیب ورود (FCFS) زمانبندی شده است. فرض کنید ۶ کار همگی در زمان صفر به سیستم وارد می‌شوند و زمان اجرای آنها به ترتیب ۳، ۸، ۱۲، ۲۰، ۲۵، ۳۵ میکروثانیه است. متوسط Turnaround Time کارهای فوق در این سیستم چقدر خواهد بود؟ به طور کامل توضیح دهید.

۴- جدول پردازش‌های زیر را داریم:

<i>Process</i>	<i>Arrival Time</i>	<i>Burst Time</i>
<i>P1</i>	0	6
<i>P2</i>	1	2
<i>P3</i>	2	8
<i>P4</i>	3	3
<i>P5</i>	4	4

با استفاده از الگوریتم‌های زمان‌بندی زیر، این پردازش‌ها را زمان‌بندی کنید (نمودار Grantt هر زمانبندی را رسم کنید):

- First Come First Serve Non-Preemptive
- Shortest Job First Non-Preemptive
- Shortest Remaining Job First Preemptive
- Round Robin with Quantum = 1 and Context Switch = 0.5



- Round Robin with Quantum = 4 and Context Switch = 0.5

الف) برای هر الگوریتم، میانگین زمان انتظار، Turnaround Time و CPU Utilization را محاسبه کنید. عملکرد هر الگوریتم و اینکه کدام یک را برای این مجموعه از پردازشها توصیه می کنید، مقایسه و بحث کنید.

ب) در الگوریتم چرخشی اگر اندازه کوانتوم زمانی نزدیک به تعویض پردازش باشد چه اتفاقی می افتد؟ اگر از تعویض پردازش خیلی بزرگتر باشد چه اتفاقی می افتد؟ کوانتوم زمانی بهینه چه نسبتی با تعویض پردازش باید داشته باشد؟ در خصوص پاسخ خود به دو زمانبندی آخر به سوالات پاسخ دهید.

بخش ناحیه بحرانی

۳- جفت پردازشهای زیر متغیر شمارنده را به اشتراک می گذارند که قبل از شروع اجرای هر یک از پردازشها مقدار اولیه ۱۰ داده شده است:

Process A	Process B
...	...
A1: LD(counter, R0)	B1: LD(counter, R0)
ADDC(R0, 1, R0)	ADDC(R0, 2, R0)
A2: ST(R0, counter)	B2: ST(R0, counter)
...	...

الف) اگر پردازشهای A و B بر روی یک سیستم اشتراک زمانی اجرا شوند، شش ترتیب وجود دارد که در آن دستورات LD و ST ممکن است اجرا شوند. برای هر یک از ترتیب اجرا، مقدار نهایی متغیر شمارنده را بدست آورید.

در دو سوال زیر از شما خواسته می شود که برنامه های اصلی را برای پردازشهای A و B با اضافه کردن حداقل تعداد سمافورها و Signal & Wait تغییر دهید تا تضمین شود که نتیجه نهایی اجرای دو پردازش یک مقدار مشخص برای شمارنده خواهد بود. برای هر سمافوری که معرفی می کنید، مقادیر اولیه را مشخص کنید.

ب) سمافور اضافه کنید تا مقدار نهایی شمارنده ۱۲ شود.

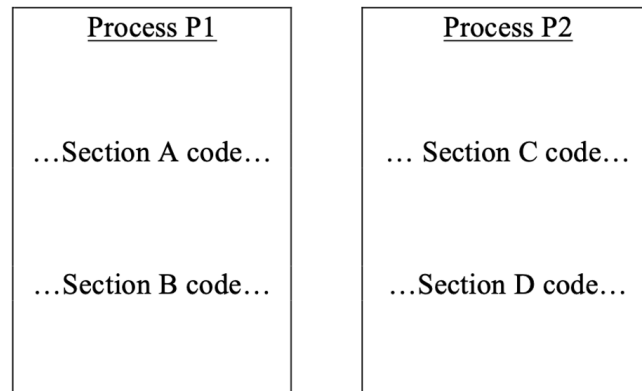
ج) سمافور اضافه کنید تا مقدار نهایی شمارنده ۱۳ نشود.



۴- $P1$ و $P2$ پردازشهایی هستند که همزمان اجرا می شوند. $P1$ دارای دو بخش از کد است که بخش A توسط بخش B دنبال می شود. به طور مشابه، $P2$ دارای دو بخش است: C و سپس D . در هر پردازش اجرای به صورت متوالی پیش می رود، بنابراین ما تضمین می کنیم که $A \leq B$ ، یعنی A قبل از B باشد. به طور مشابه، ما می دانیم که $C \leq D$. هیچ حلقه ای در پردازشها وجود ندارد. هر پردازش دقیقاً یک بار اجرا می شود.

از شما خواسته می شود که سمافورها را به برنامه ها اضافه کنید - ممکن است لازم باشد از بیش از یک سمافور استفاده کنید. مقادیر اولیه هر سمافوری که استفاده می کنید را مشخص کنید. برای نمره کامل از حداقل تعداد سمافورها استفاده کنید و هیچ گونه محدودیت اولویت غیر ضروری را معرفی نکنید.

Semaphore initial values: _____



الف) دستورات $WAIT$ (...) و $SIGNAL$ (...) را به پردازشهای زیر اضافه کنید تا محدودیت اولویت $B \leq C$ برآورده شود، یعنی اجرای $P1$ قبل از شروع اجرای $P2$ به پایان برسد.

ب) دستورات $WAIT$ (...) و $SIGNAL$ (...) را به پردازشهای زیر اضافه کنید تا $D \leq A$ یا $B \leq C$ ، به عنوان مثال، اجرای $P1$ و $P2$ نتوانند همپوشانی داشته باشند، اما اجازه داده شود به هر ترتیب انجام شوند.

ج) دستورات $WAIT$ (...) و $SIGNAL$ (...) را به پردازشهای زیر اضافه کنید تا $A \leq D$ و $C \leq B$ ، یعنی بخش اول (A و C) هر دو پردازش اجرا خود را قبل از اینکه بخش دوم (B و D) اجرای خود را آغاز کند، کامل کنند.



۵- شروط انحصار متقابل، پیشرفت و انتظار محدود را برای الگوریتم‌های زیر بررسی کرده و دلیل خود را بنویسید.

(الف)

```
do {  
    flag[i] = true;  
    turn = j;  
    while (!flag[j] || turn == j);  
        critical section  
    flag[i] = false;  
        remainder section  
} while (true);
```

(ب)

```
do {  
    flag[j] = true;  
    turn = j;  
    while (flag[i] && turn == j);  
        critical section  
    flag[j] = false;  
        remainder section  
} while (true);
```



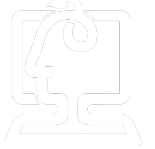
۶- الف) بدون استفاده از قفل و تنها با استفاده از دستورا `compare-and-swap` تابع زیر را به گونه‌ای کامل کنید که به صورت اتمی عملیات جمع را انجام دهد. منظور از عملیات جمع اضافه شدن مقدار `v` به حافظه‌ای است که `p` به آن اشاره دارد. سپس توضیح دهید تضمینی برای انجام شدن این عملیات وجود دارد یا خیر.

```
int add(int *p, int v)
{
    // TODO
    return *p + v;
}
```

پیاده سازی `compare-and-swap` را به صورت زیر در نظر بگیرید:

```
bool compare_and_swap(int *p, int old, int new)
{
    if(*p != old)
        return false;
    *p = new;
    return true;
}
```

ب) می‌دانید که برای پیاده‌سازی توابع `acquire()` و `release()` در قفل `mutex` باید از دستورات سخت افزاری اتمی استفاده کرد. این کار را استفاده از دستور `test-and-set` انجام دهید.



به نکات زیر توجه کنید.

- مهلت ارسال تمرین ساعت ۲۳:۵۹ روز یکشنبه ۳۱ اردیبهشت ماه می باشد.
- در صورت کشف تقلب نمره تمرین ۰ در نظر گرفته می شود.
- سوالات خود را می توانید از طریق تلگرام از تدریس‌یارهای گروه خود بپرسید.
- فایل پاسخ تمرین را تنها با قالب **HW?_StudentNumber.pdf** در کورسز بارگزاری کنید.
- نمونه: HW4_9831072