پایانترم

Started on	Saturday, 18 Dey 1400, 9:00 AM
State	Finished
Completed on	Saturday, 18 Dey 1400, 11:26 AM
Time taken	2 hours 25 mins
Grade	**36.00** out of 60.00 (**60**%)

Question 1
Complete
Mark 2.00 out of 10.00
Flag question

راه حل زیر برای مسیله خوانندگان نویسندگان (اولویت با نویسندگان) ارایه شده است. با ذکر دلیل توضیح دهید آیا انحصار متقابل لازم برای مسیله خوانندگان نویسندگان رعایت شده است؟ همچنین با ذکر دلیل بیان کنید آیا امکان ایجاد بن بست و گرسنگی در این راه حل وجود دارد یا نه.

```
int writer_cnt = 0;
semaphore  reader = 1;
semaphore  cnt_mutex = 1 ,  w_mutex = 1;
```

reader	writer

If (writer_cnt>=1)
wait(reader)
/* CS: reading is performed */
	wait(cnt_mutex);
	writer_cnt ++;
	if(writer_cnt==1)
	wait(reader)
	signal(cnt_mutex);
	wait(w_mutex);
	/* CS: writing is performed */
	signal(w_mutex);
	
	wait(cnt_mutex);
	writer_cnt --;
	if(writer_cnt==0)
	signal(reader);
	signal(cnt_mutex);

انحصار متقابل برقرار است، هیچ یک از نویسندگان یا خوانندگان نمیتوانند وارد CS شوند اگر دیگری از قبل در حال اجرای CS باشد

امکان ایجاد بن بست وجود ندارد، زیرا شرط circular wait برقرار نیست

امکان ایجاد گرسنگی وجود دارد، زیرا اگر یک ریدر وارد سیستم شود، سمافور reader قفل میشود و دیگر باز نخواهد شد، در نتیجه دیگر پروسه ها نمیتوانند به CS دسترسی پیدا کنند (کد ریدر نیاز به یک signal پس از CS دارد)

و از wait درون if هم میگذرد زیرا سمافور reader به صورت پیش‌فرض ۱ است و هنوز کم نشده است. بنابراین نویسنده خطوط بعدی را هم اجرا کرده و وارد CS میشود. پس همزمان هم خواننده و هم نویسنده در CS هستند. **پس انحصار متقابل خواننده نویسنده رعایت نشده است.** اما انحصار متقابل بین هر دو نویسنده‌ای رعایت شده به دلیل قرار گرفتن CSنویسنده درون قفل w_mutex.

(۳ نمره)بن بست: بن‌بست میتواند اتفاق بیفتد. اگر یک یا بیش از یک نویسنده اقدام به نوشتن بکنند با اقدام اولین نویسنده سمافور reader صفر میشود، پس از آن اگر چند خواننده وارد شوند به ازای اجرای هر یک، سمافور reader یک بار wait می‌شود و همگی در خط مربوطه بلاک میشوند. با توجه به خط if(writer_cnt==0) signal(reader) در کد نویسنده، زمانی سمافور reader سیگنال می‌شود که همه نویسنده‌ها نوشتنشان تمام شود و نویسنده دیگری هم اقدام به نوشتن نکند. در این صورت یکی از خواننده‌ها میتواند از خط wait گذر کند و بقیه همچنان در این نقطه میمانند و اگر نویسنده جدیدی وارد شودآن نویسنده هم وارد if(writer_cnt==1) می‌شود و آن هم در فراخوانی wait(reader) بلاک می‌شود و بدین ترتیب همه خواننده‌ها و اولین نویسنده در بن‌بست میمانند و اگر نویسندگان دیگری هم وارد شوند آن‌ها هم در wait(cnt_mutex) بلاک می‌شوند و لذا همه یکی پس از دیگری در بن‌بست می‌افتند

(۳ نمره)گرسنگی امکان‌پذیر است حتی اگر اجرای خوانندگان و نویسندگان به ترتیبی انجام شود که بن‌بست هم اتفاق نیفتد، به دلیل اولویت نویسندگان، اگر نویسندگان یکی پس از دیگری وارد شوند و فقط یک خواننده هم منتظر ورود باشد آن یک خواننده دچار گرسنگی می‌شود اگر نویسندگان هیچ وقت تمام نشوند

Comment:

فقط مورد آخر درست است و آن هم دلیلش درست نیست

Question 2

Complete

Mark 6.00 out of 10.00

Flag question

در یک سیستم آموزش مجازی کمک درسی، هر معلم میتواند در صورت تمایل، اعلام تشکیل کلاس مجازی دهد، از طرفی دانش آموزان متقاضی تشکیل کلاس هم می‌توانند در کلاسها شرکت کنند. در صورتی یک کلاس تشکیل می‌شود که یک معلم متقاضی موجود باشد و دانش‌اموزان متقاضی هم دقیقاً ۱۰ نفر باشند. هر معلم متقاضی تابع createClass و هر دانش‌آموز تابع joinClass را در threadهای مختلف اجرا می‌کند. اما تا قبل از اینکه معلمی کلاس تشکیل نداده باشد یا تعداد دانش‌آموزان به ۱۰ نرسیده باشد کلاس آغاز نمی‌گردد (startClass) و معلم یا دانش‌آموز تا فراهم‌شدن شرایط به حالت انتظار می‌روند. اگر تعداد دانش‌اموزان بیش از ۱۰ باشد، بقیه دانش‌آموزان در کلاس‌های بعدی(در صورت وجود معلم) قرار می‌گیرند. توابع createClass و joinClass رابا استفاده از سمافور و یا متغیر شرطی تکمیل کنید (در سودوکدهایی که می‌نویسید startClass در محل مناسب فراخوانی شود متغیر و سمافورهای لازم به درستی **تعریف و مقداردهی اولیه** شوند)

joinClass	createClass

//your code
startClass()
//yourCode
	//your code
	startClass()
	//your code

2.c

```
sem class_created = 0;
sem std_mutex = 1;
int std_num = 0;
cond_t teacher;
```

joinClass	createClass

wait(std_mutex);
std_num++;
if(std_num>=10)
  signal(teacher);
signal(std_mutex);
wait(class_created)
startClass()
	wait(std_mutex);
	while(std_num<10)
	  wait(teacher, std_mutex);
	std_num = std_num -10;
	signal(std_mutex);
	for(i=0;i<10;i++)
	  signal(class_created)
	startClass()

Comment: همه کلاسها را یک معلم مجبور نیست برگزار کند و بیش از یک معلم هم میتواند وجود داشته باشد
چرا joined بیدلیل سیگنال شده؟

Question 3
Complete
Mark 8.00 out of 10.00
Flag question

کدهای دو تابع f1 وf2 در زیر آمده این دو تابع به صورت مشترک از دو منبع استفاده می‌کنند هر یک از این دو منبع در هر لحظه فقط توسط یکی از توابع می‌تواند استفاده شود. با دو روش مختلف <نقض circular wait> و <نقض no preemption>، کدهای زیر را به نوعی **تغییر دهید یا بازنویسی کنید** که از ایجاد بن‌بست (deadlock) جلوگیری شود.

semaphore s1=1, s2=1;

f1	f2

wait(s1)
wait(s2)
//CS
signal(s1)
signal(s2)
	wait(s2)
	wait(s1)
	//CS
	signal(s2)
	signal(s1)

نقض circular wait
f1:
wait(s1)
wait(s2)
//CS
signal(s2)
signal(s1)

f2:
wait(s1)
wait(s2)
//CS
signal(s2)
signal(s1)

نقض no preemption
f1{
wait(s1)
wait(s2)
//CS
signal(s1)
signal(s2)
}
f2{
start:
wait(s2)
if(not_available(s1)){
  signal(s2) //preemptively release s2
  goto start
}
wait(s1)
//CS
signal(s2)
signal(s1)
}

جهت نقض circular wait کنند بنابراین برای هر دو تابع باید بنویسیمwait کافیست هر دو تابع با یک ترتیب سمافورها را

f1	f2

wait(s1)
wait(s2)
//CS
signal(s1)
signal(s2)
	wait(s1)
	wait(s2)
	//CS
	signal(s1)
	signal(s2)

می‌توان در صورتی که یک پروسس منبعی را در دست دارد و سپس منبع دیگری را میخواهد که آزاد نیست، منبع قبلی را هم از او گرفت و تا وقتی هر دو منبع آزاد شوند صبر کرد و پس از آن هر دو را به اون ، no preemption جهت نقض
داد این کار را در مورد این سوال میتوان با کمک متغیر شرطی به صورت زیر انجام داد.

f1	f2

wait(s1)
while (s2_count>0)
wait(cond_s2, s1);
	wait(s2)
	lock(mutex2)
	s2_count++;

wait(cond_s2, s1);	s2_count ++;
wait(s2)	unlock(mutex2)
//CS	wait(s1)
signal(s1)	//CS
signal(s2)	signal(cond_s2)
	lock(mutex2)
	s2_count --;
	unlock(mutex2)
	signal(s2)
	signal(s1)

Comment:

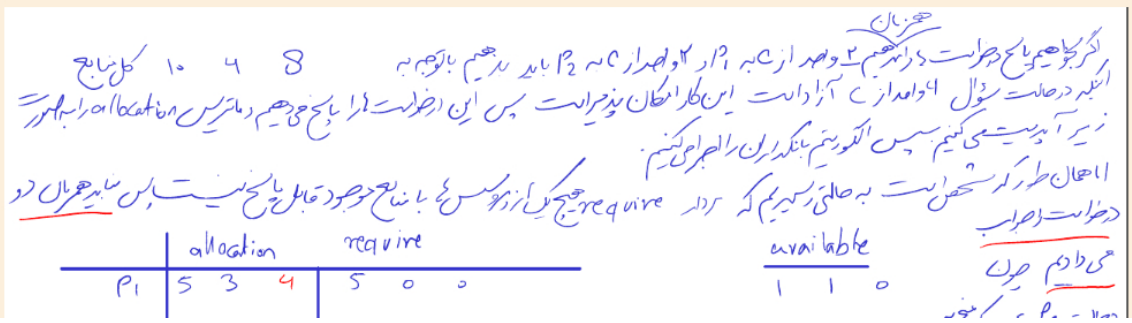
چک کردن مستقیم مقدار سمافور کار درستی نیست
کد شما مشکل busy waiting هم دارد

Question 4
Complete
Mark 10.00 out of 10.00
Flag question

در یک سیستم، موجودیهای ۳ منبع به صورت A=10, B=4, C=8 است. اگر وضعیت تخصیص منابع در یک حالت سیستم به صورت زیر باشد، سپس همزمان p1 و p2 هر کدام به صورت جداگانه ۲ واحد C درخواست دهند، آیا سیستم با توجه به الگوریتم بانکداران می‌تواند درخواست آن‌ها را پاسخ دهد؟ (پاسخ با ذکر کامل روند حل مسیله با الگوریتم بانکداران)

process	Max need			allocation		
	A	B	C	A	B	C
p1	10	3	4	5	3	2
p2	0	2	3	0	0	1
p3	5	0	3	2	0	0
p4	3	0	3	2	0	1

4.png



P_2	0	0	3	0	2	0
P_3	2	0	0	3	0	3
P_4	2	0	1	1	0	2

3 1 5
5 1 5

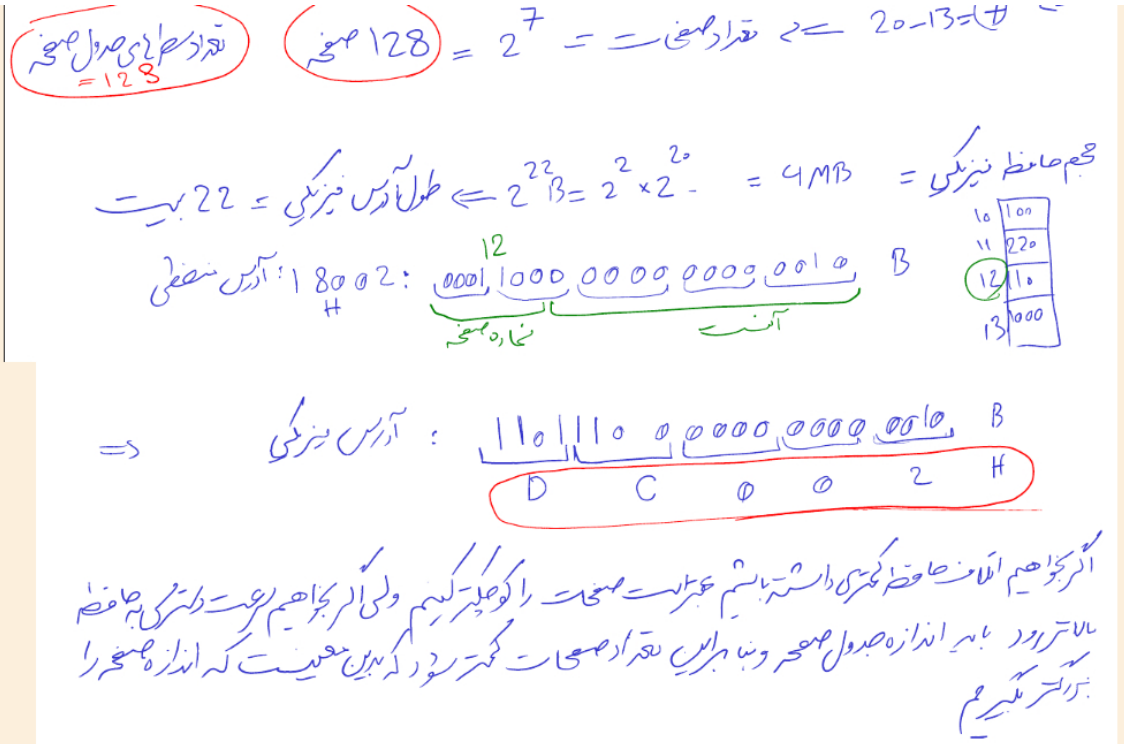
Comment:

Question 5
Complete
Mark 3.00 out of 10.00
Flag question

در یک سیستم مدیریت حافظه از صفحه‌بندی (paging) با اندازه صفحه (page) 8 کیلوبایت استفاده می‌شود و فضای آدرس منطقی(virtual) هر پروسس، یک مگابایت و حجم حافظه فیزیکی موجود 4 مگابایت است. (۱) جدول صفحه (page table) در این سیستم دارای چند سطر است؟ (۲) با توجه به سطرهای ۱۰ تا۱۳ جدول صفحه که در شکل با اعداد دسیمال (ده دهی) آمده است، آدرس منطقی 18002**Hex**، معادل چه آدرس فیزیکی در فرمت Hex است؟ (مراحل بدست آوردن جوابها را بنویسید.) (۳) اگر بخواهیم حافظه را بهینه‌تر استفاده کنیم و یا سرعت دسترسی به حافظه بالاتری داشته باشیم اندازه صفحه را بزرگتر کنیم یا کوچکتر؟ شرح دهید

100
220
110
500

5.png



اگر بخواهیم اتلاف حافظه کمتری داشته باشیم بهتر است صفحات را کوچکتر کنیم ولی اگر بخواهیم سرعت دسترسی به حافظه بالاتر برود باید اندازه جدول صفحه و بنابراین تعداد صفحات کمتر شود که بدین معنیست که اندازه صفحه را بزرگتر بگیریم

Comment:

برای قسمت الف باید تعداد آدرسهای مجازی را برای اندازه جدول صفحه درمیاوردید. شما تعداد آدرسهای فیزیکی را محاسبه کردید

قسمت ب صحیح است که ۳ نمره داشته

Question 6

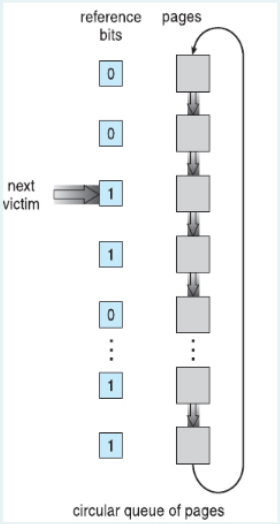
Complete

Mark 7.00 out of 10.00

Flag question

یک روش page replacement به صورت زیر پیشنهاد شده است:

هر فریم، دارای یک بیت رفرنس است. به صورت پیش‌فرض این بیت برای همه فریمها صفر است. هرگاه به یک صفحه رجوع می‌شود، بیت رفرنس فریم متناظرش یک می‌شود. یک اشاره گر به صورت چرخشی روی فریمها حرکت می‌کند. هرگاه بخواهیم صفحه‌ای را برای جایگزینی (replacement) انتخاب کنیم بیت رفرنس فریمی که اشاره‌گر بدان اشاره می‌کند را چک میکنیم اگر این بیت صفر بود، صفحه مربوطه برای جایگزینی انتخاب می‌شود. اما اگر این بیت یک بود بیت تبدیل به صفر می‌شود و اشاره‌گر جلو می‌رود و به همین ترتیب صفحات بعدی برای جایگزینی بررسی می‌شوند تا به اولین فریمی که بیت رفرنسش صفر است برسیم.

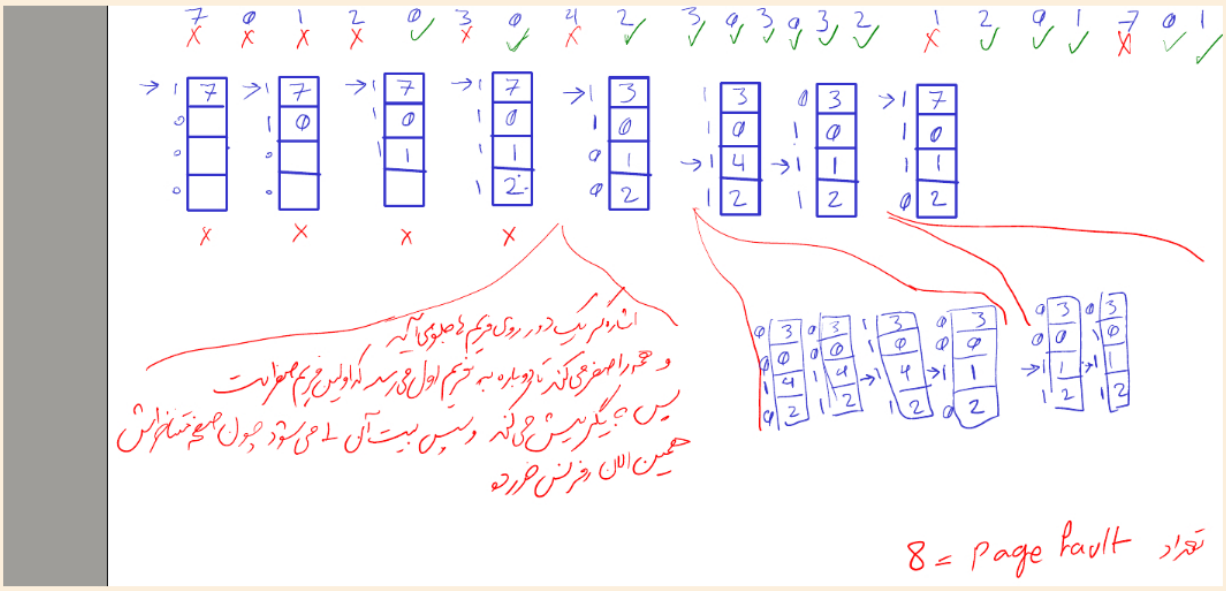


تعداد page fault را برای این الگوریتم روی رشته رفرنس زیر (reference string) از چپ به راست با ۴ فریم بدست آورید:

7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

این الگوریتم را از نظر تعداد page fault، سربار زمانی و حافظه ای اجرا با الگوریتم LRU مقایسه کنید (متناسب با جواب، علت را توضیح دهید)

6.png



استفاده کند بدین ترتیب همیشه دقیق عمل نمیکند زیرا اشاره گر LRU است که سعی شده فقط از یک بیت برای پیدا کردن LRU عملکرد بهتری دارد. زیرا الگوریتم شرح داده شده تخمینی از page fault از نظر تعداد LRU به طور متوسط به صورت چرخشی جلو میرود و به دنبال صفحه ای که در دیرترین زمان نسبت به حال رفرنس خورده میگردد اما سرچ کامل انجام نمیدهد و اولین صفحه ای که بیتش صفر بود را جایگزین می‌کند.
نگهداری کند (طول بیش از یک بیت) timestamp است از نظر سربار زمانی بهتر است از نظر سربار حافظه هم بهتر است زیرا به جای اینکه به ازای هر صفحه یک LRU به همین دلیل که سرچ کامل صورت نمیگیرد این الگوریتم سریعتر از فقط یک بیت نگه داری میکند.

Comment: ۳- تحلیل بخش دوم

Finish review