دانشکده ی مهندسی کامپیوتر

دانشگاه صنعتی امیرکبیر

تمرین هشتم درس سیستم عامل

مهلت تحویل ساعت ۲۳:۵۹ روز جمعه ۱۲ آذر ۱۴۰۰

تمرینات را انفرادی حل کرده و در سایت مودل (courses.aut.ac.ir) با قالب زیر بارگذاری نمایید:

StudentID_Name_Last Name

درصورت داشتن سوال درمورد این تمرین، سوال خود را با موضوع تمرین ۸ با ایمیل زیر در میان بگذارید:

OsFall1400@gmail. com

۱- اجرای دستورات *acquire()* و *release()* در قفل *mutex* باید اتمیک باشد، لذا معمولا با استفاده از دستورات سخت‌افزاری (که همگی اتمیک هستند) پیاده‌سازی می‌شوند. نشان دهید چطور با استفاده از دستور *compare_and_swap()* می‌توان آنها را پیاده‌سازی کرد. دقت نمایید پیاده‌سازی نباید منجر به بن‌بست (*deadlock*) شود (رفتار دستور فوق در زیر به زبان *C* آمده است).

```
int compare_and_swap(int *value, int expected, int new_value) {
    int temp = *value;
    if (*value == expected)
    *value = new_value;
    return temp;     /* old value */
}
```

۲- سه فرآیند همروند زیر را در نظر بگیرید که دارای سه سمافور باینری هستند. سمافورها به صورت S1=1، S0=0، S2=0 مقداردهی اولیه شده‌اند. فرایند *P1* چند بار عبارت *"Hello world"* را چاپ خواهد کرد؟

Process P0	*Process P1*	*Process P2*

`wait(S0);`
`release (S1);`
	`while(True)`
	`{`
	`  wait (S1);`
	`  print ' Hello world ';`
	`  release (S0);`
	`  release (S2);`
	`}`
		`wait(S2);`
		`release (S1);`

۳- در مورد دستورات سخت‌افزاری حل ناحیه بحرانی *Test_and_Set* و *Compare_and_Swap* به سوالات زیر پاسخ دهید:

الف) چطور دستور *Test_and_Set* را می‌توان با استفاده از *Compare_and_Swap* پیاده‌سازی کرد؟

ب) هنگام استفاده از هر یک در زبان‌های سطح بالا (مثل *C*) داخل حلقه *while* قبل از ناحیه بحرانی، آیا اتمیک بودن دستورات *while* و دیگر دستورات سطح بالا مهم نیست؟ چرا؟

ج) چطور در سیستم‌های تک هسته‌ای (تک *program counter*)، اتمیک بودن دستورات فوق، مشکل ناحیه بحرانی را حل می‌کند (به عبارتی توضیح دهید چرا حتی با تعویض متن، شرط مسابقه و ناسازگاری پیش نخواهد آمد و چطور انحصار متقابل تامین می‌شود)؟

د) در سیستم‌های چند هسته‌ای یا چند پردازنده‌ای که دارای چندین سخت‌افزار موازی من جمله *program counter* هستند، آیا دستورات فوق مشکل‌ساز نیستند؟ آیا شرایطی پیش نمی‌آید که دو فرآیند در دو هسته بصورت موازی (مثالی از همروندی) اجرا شوند طوریکه هر دو در حین استفاده از دستور *Test_and_Set* (یا *Compare_and_Swap*) روی یک متغیر مشترک از حافظه باشند و در این حالت، شرط مسابقه و ناسازگاری پیش آید یا انحصار متقابل تامین نشود؟ بحث کنید.