تمرین چهارم سیستم‌های عامل

اشکان شکیبا (۹۹۳۱۰۳۰)

سوال اول

ابتدا در سطح اول، پردازه‌ها با برش زمانی ۸ میکروثانیه اجرا می‌شوند:

P1	P2	P3	P4	P5	P6

0 3 11 19 27 35 43

سپس پردازه‌هایی که اجرای آنها به پایان نرسیده، وارد سطح بعد با برش زمانی ۱۶ میکروثانیه می‌شوند:

P3	P4	P5	P6

0 4 16 32 48

سپس پردازه‌های باقی‌مانده وارد آخرین صف می‌شوند و تا زمانی که اجرایشان به پایان نرسد متوقف نمی‌شوند:

P5	P6

0 1 12

average turnaround time =

((3)+(11)+(43+4)+(43+16)+(43+48+1)+(43+48+12)) / 6 = 52.5

سوال دوم

الف)

الگوریتم First Come First Serve Non-Preemptive

P1	P2	P3	P4	P5

0 6 8 16 19 23

average waiting time = (0+5+6+13+15) / 5 = 7.8

average turnaround time = (6+7+14+16+19) / 5 = 12.4

CPU utilization = 100%

الگوریتم Shortest Job First Non-Preemptive

P1	P2	P4	P5	P3

0 6 8 11 15 23

average waiting time = (0+5+13+5+7) / 5 = 6

average turnaround time = (6+7+21+8+11) / 5 = 10.6

CPU utilization = 100%

الگوریتم Shortest Remaining Job First Preemptive

P1	P2	P4	P5	P1	P3

0 1 3 6 10 15 23

average waiting time = (9+0+13+0+2) / 5 = 4.8

average turnaround time = (15+2+21+3+6) / 5 = 9.4

CPU utilization = 100%

الگوریتم Round Robin with Quantum = 1 and Context Switch = 0.5

P1	↻	P2	↻	P1	↻	P3	↻	P2	↻	P4	↻	P5	↻	P1

0 1 1.5 2.5 3 4 4.5 5.5 6 7 7.5 8.5 9 10 10.5

↻	P3	↻	P4	↻	P5	↻	P1	↻	P3	↻	P4	↻	P5	↻

11.5 12 13 13.5 14.5 15 16 16.5 17.5 18 19 19.5 20.5 21 22 22.5

P1	↻	P3	↻	P5	↻	P1	↻	P3

23.5 24 25 25.5 26.5 27 28 28.5 32.5

average waiting time = (22+4+22.5+13.5+19.5) / 5 = 16.3

average turnaround time = (28+6+30.5+17.5+22.5) / 5 = 20.9

CPU utilization = (23 / 32.5) * 100 = 71%

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

P1	↻	P2	↻	P3	↻	P4	↻	P5	↻	P1	↻	P3

0 4 4.5 6.5 7 11 11.5 14.5 15 19 19.5 21.5 22 26

average waiting time = (15.5+3.5+16+8.5+11) / 5 = 10.9

average turnaround time = (21.5+5.5+24+11.5+15) / 5 = 15.5

CPU utilization = (23 / 26) * 100 = 88%

ب) اگر اندازه کوانتوم زمانی نزدیک به تعویض پردازه باشد، overhead بیشتری داشته و CPU utilization کاهش می‌یابد. از سوی دیگر اگر اندازه کوانتوم زمانی خیلی بزرگ باشد، الگوریتمی تقریبا مشابه FCFS خواهیم داشت که می‌تواند دچار starvation شود. از این رو باید اندازه کوانتوم زمانی مقداری مناسب داشته باشد که بهینه‌ترین حالت رخ دهد. یک دیدگاه این است که اندازه کوانتوم زمانی باید طوری انتخاب شود که حدودا چهار پنجم از CPU burstها از آن کوچک‌تر باشند.

سوال سوم

الف) ۶ ترتیب متفاوت داریم:

LD_A, LD_B, ST_A, ST_B

مقدار نهایی متغیر شمارنده: 12

LD_B, LD_A, ST_B, ST_A

مقدار نهایی متغیر شمارنده: 11

LD_A, LD_B, ST_B, ST_A

مقدار نهایی متغیر شمارنده: 11

LD_B, LD_A, ST_A, ST_B

مقدار نهایی متغیر شمارنده: 12

LD_A, ST_A, LD_B, ST_B

مقدار نهایی متغیر شمارنده: 13

LD_B, ST_B, LD_A, ST_A

مقدار نهایی متغیر شمارنده: 13

ب) سمافورهای P و Q را با مقدار اولیه صفر تعریف می‌کنیم.

```
Process A:

LD(counter, R0)

ADDC(R0, 1, R0)

WAIT(P)

ST(R0, counter)

SIGNAL(Q)


Process B:

LD(counter, R0)
```

```
ADDC(R0, 2, R0)
SIGNAL(P)
WAIT(Q)
ST(R0, counter)
```

ج)

Process A:

```
LD(counter, R0)
ADDC(R0, 1, R0)
WAIT(P)
SIGNAL(Q)
ST(R0, counter)
```

Process B:

```
LD(counter, R0)
ADDC(R0, 2, R0)
SIGNAL(P)
WAIT(Q)
ST(R0, counter)
```

سوال چهارم

الف) سمافور P را با مقدار اولیه صفر تعریف می‌کنیم.

```
P1:

Section A code

Section B code

SIGNAL(P)


P2:

WAIT(P)

Section C code

Section D code
```

ب) سمافور P را با مقدار اولیه یک تعریف می‌کنیم.

```
P1:

WAIT(P)

Section A code

Section B code

SIGNAL(P)
```

P2:

WAIT(P)

Section C code

Section D code

SIGNAL(P)

ج) سمافورهای P و Q را با مقدار اولیه صفر تعریف می‌کنیم.

P1:

Section A code

WAIT(P)

SIGNAL(Q)

Section B code

P2:

Section C code

SIGNAL(P)

WAIT(Q)

Section D code

سوال پنجم

الف) انحصار متقابل وجود دارد، چون با توجه فلگ turn، هیچ گاه دو پردازه همزمان وارد ناحیه خطرناک نمی‌شوند.

پیشرفت وجود ندارد، چون اگر یک پردازه به ابتدای حلقه برنگردد و turn را به پردازه دیگر ندهد، پردازه دیگر نمی‌تواند وارد ناحیه بحرانی شود.

انتظار محدود وجود دارد، زیرا در هر دو پردازه هر بار turn به پردازه دیگر داده می‌شود.

ب) انحصار متقابل وجود دارد، چون مشابه بخش الف، زمانی پردازه‌ای وارد ناحیه بحرانی می‌شود که وضعیت فلگ‌ها در پردازه دیگر از این موضوع جلوگیری می‌کند.

پیشرفت وجود دارد، چرا که هر پردازه بلافاصله پس از خروج از ناحیه بحرانی فلگ پردازه دیگر را تغییر می‌دهد و با توجه به تغییر turn به پردازه دیگر در ابتدای حلقه، پردازه‌ای که خارج از ناحیه بحرانی است نمی‌تواند مانع ورود پردازه دیگر به آن شود.

انتظار محدود وجود دارد، چون مشابه بخش الف هر بار turn به پردازه دیگر داده می‌شود.

سوال ششم

الف)

```
int add(int *p, int v) {
	int lock = 1;
	while(!compare_and_swap(&lock, 1, 0));
	int r = *p + v;
	lock = 1;
	return r;
}
```

خیر وجود ندارد، چون در این روش انتظار محدود نداریم و ممکن است هیچ گاه lock به پردازه ما نرسد.

ب)

```
bool lock = false;
void acquire() {
	while(test_and_set(&lock));
}
void release() {
	lock = false;
}
```