دانشکده مهندسی کامپیوتر

دکتر رضا انتظاری ملکی

پاییز ۱۴۰۰

سیستم‌های عامل

پاسخنامه تمرین سری سوم

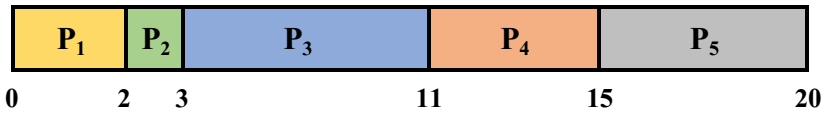
دانیال بازمانده – ملیکا احمدی رنجبر

سوال اول

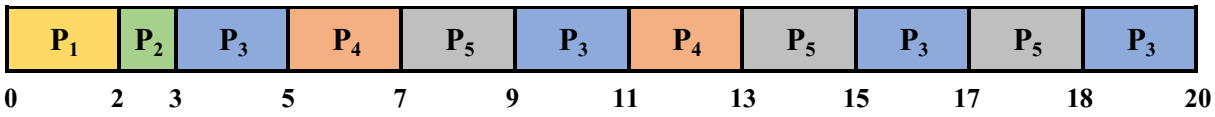
پاسخ:

آ) گانت چارت های مربوط به هریک از الگوریتم‌های زمان‌بندی خواسته‌شده به شکل زیر می‌باشند:

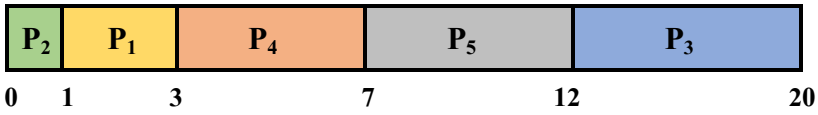
FCFS:



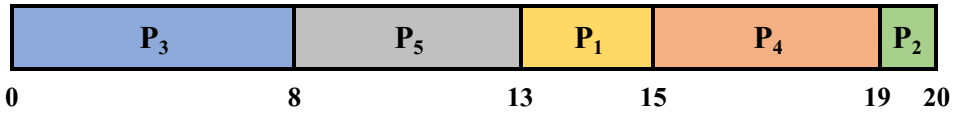
RR (Quantum=2):



SJF:



Non-Preemptive Priority:



ب) مقدار زمان برگشت (turnaround time) برای هریک از الگوریتم‌های فوق به صورت زیر است:

	P_1	P_2	P_3	P_4	P_5
FCFS	2	3	11	15	20
RR(q=2)	2	3	20	13	18
SJF	3	1	20	7	12
Non-Preemptive Priority	15	20	8	19	13

ج) مقدار زمان انتظار (waiting time) برای هریک از الگوریتم‌های فوق به صورت زیر است:

	P_1	P_2	P_3	P_4	P_5
FCFS	0	2	3	11	15
RR(Q=2)	0	2	12	9	13
SJF	1	0	12	3	7
Non-Preemptive Priority	13	19	0	15	8

د) با محاسبه‌ی میانگین زمان انتظار برای هریک از الگوریتم‌های مربوطه داریم:

$$\text{FCFS}: \frac{2+3+11+15}{5} = 6.2 \qquad \text{RR(Q=2)}: \frac{2+12+9+13}{5} = 7.2$$

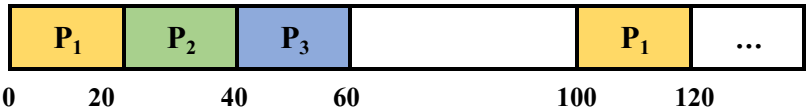
$$\text{SJF}: \frac{1+12+3+7}{5} = 4.6 \qquad \text{Non-Preemptive Priority}: \frac{13+19+15+8}{5} = 11$$

با توجه به مقادیر محاسبه‌شده واضح است که الگوریتم SJF میانگین زمان انتظار کمتری نسبت به سایر الگوریتم‌ها دارد.

سوال دوم

پاسخ:

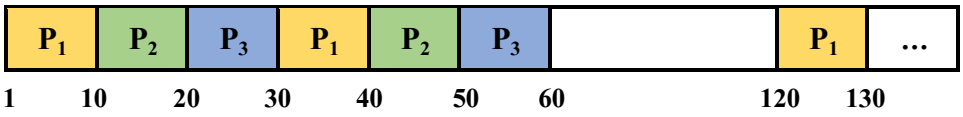
۱) استفاده از الگوریتم FCFS: در این حالت، هرکدام از پردازه‌ها به ترتیب، به مدت ۲۰ میلی ثانیه، از CPU استفاده می‌کنند. پردازه‌ی P_1 بعد از ۲۰ میلی ثانیه‌ی اول، به سراغ I/O می‌رود که ۸۰ میلی ثانیه است و این یعنی در ۱۰۰ میلی ثانیه تمام می‌شود و پس از آن، دوباره P_1 از CPU استفاده می‌کند.



با توجه به اینکه CPU در فاصله‌ی بین ۶۰ تا ۱۰۰ بیکار است، داریم:

CPU Utilization = 60 / 100 = 60%

۲) استفاده از الگوریتم Round Robin(Q=10): در این حالت، ابتدا هریک از پردازه‌های P_1، P_2 و P_3 به مدت ۱۰ میلی ثانیه از CPU استفاده می‌کنند. بنابراین بعد از ۴۰ میلی ثانیه، P_1 با I/O شروع می‌شود و بعد از ۵۰ میلی ثانیه P_2 نیز با I/O و پس از ۶۰ میلی ثانیه P_3 شروع می‌شود. از آنجایی که عملیات I/O را می‌توان به صورت موازی انجام داد، عملیات I/O مربوط به P_1 در ۱۲۰ (۸۰+۴۰)، عملیات مربوط به P_2 در ۱۳۰ (۸۰+۵۰) و عملیات مربوط به P_3 در ۱۴۰ (۸۰+۶۰) میلی ثانیه به پایان می‌رسد.



با توجه به اینکه CPU در فاصله‌ی بین ۶۰ تا ۱۲۰ بیکار است، داریم:

CPU Utilization = 60 / 120 = 50%

سوال سوم

پاسخ:

آ) معمولا اگر کوانتوم زمانی از حدود ۸۰ درصد CPU Burst ها بزرگتر باشد، الگوریتم Round Robin کارایی بهتری دارد.

ب) در این صورت، الگوریتم Round Robin مشابه با الگوریتم FCFS عمل می‌کند.

ج) با توجه به اینکه بخشی از زمان هر کوانتوم صرف context-switching بین پردازه‌ها می‌شود، اگر بازه زمانی کوانتوم بسیار کوتاه باشد، باعث افزایش تعداد context-switch ها و کاهش throughput خواهد شد.

د) خیر. اول اینکه الگوریتم Round Robin غیر انحصاری و الگوریتم SRTF انحصاری است. تفاوت بعدی این است که RR براساس زمان ورود به پردازه‌ها اولویت می‌دهد اما در SRTF زمان باقی‌مانده از اجرا، اولویت اجرای پردازه‌ها را تعیین می‌کند. بنابراین، این دو الگوریتم در حالت کلی نمی‌توانند عملکرد یکسانی داشته باشند. (مگر در حالت خاصی که مقدار کوانتوم بزرگتر از مقدار CPU Burst ها باشد و تمامی پردازه‌ها به ترتیب اندازه از کوچک به بزرگ وارد شده باشند.)

سوال چهارم

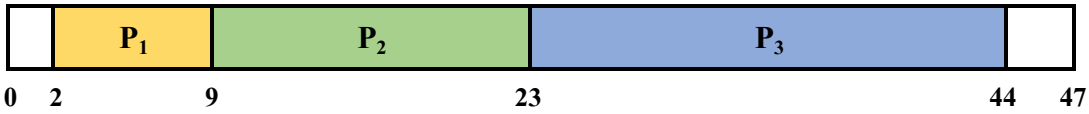
پاسخ: وضعیت اجرای هریک از پردازه‌ها به صورت زیر می‌باشد:

۱) پردازه‌ی P_1 ابتدا ۲ واحد زمانی را در I/O، ۷ واحد را از زمان CPU و در نهایت ۱ واحد را در I/O مصرف می‌کند.

۲) پردازه‌ی P_2 ابتدا ۴ واحد زمانی را در I/O، ۱۴ واحد را از زمان CPU و در نهایت ۲ واحد را در I/O مصرف می‌کند.

۳) پردازه‌ی P_3 ابتدا ۶ واحد زمانی را در I/O، ۲۱ واحد را از زمان CPU و در نهایت ۳ واحد را در I/O مصرف می‌کند.

نمودار گانت چارت اجرای این پردازه‌ها به صورت زیر است:



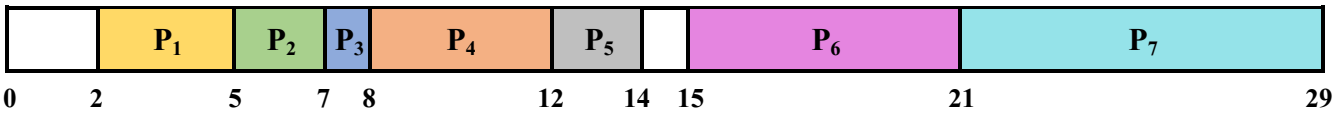
با توجه به اینکه CPU در بازه‌های (۲-۰) و (۴۷-۴۴) بیکار است، داریم:

CPU Idle Percentage = 5 / 47 * 100 = 10.6%

سوال پنجم

پاسخ:

آ) در این حالت الگوریتم FCFS اجرا می‌شود.



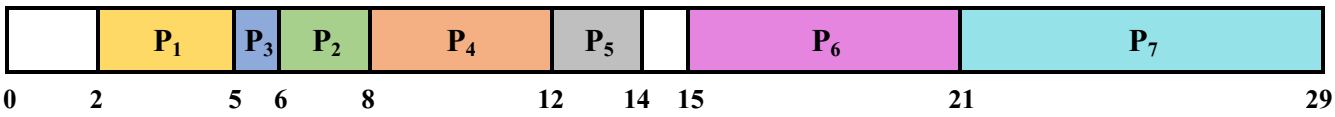
دقت شود برای محاسبه‌ی مقدار توان عملیاتی (throughput) دو واحد زمانی اول که CPU هنوز شروع به انجام کاری نکرده است، نباید درنظر بگیریم.

$$\text{throughput} = \frac{\text{number of completed processes}}{\text{the whole time}} = \frac{7}{27} \simeq 0.25$$

$$\text{Average waiting time} = \frac{0 + (5-4) + (7-5) + (8-7) + (12-9) + 0 + (21-16)}{7} = \frac{12}{7} \simeq 1.71$$

$$\text{Average turnaround time} = \frac{(5-2) + (7-4) + (8-5) + (12-7) + (14-9) + (21-15) + (29-16)}{7} = \frac{38}{7} \simeq 5.42$$

ب) در این حالت الگوریتم SJF (انحصاری) اجرا می‌شود.



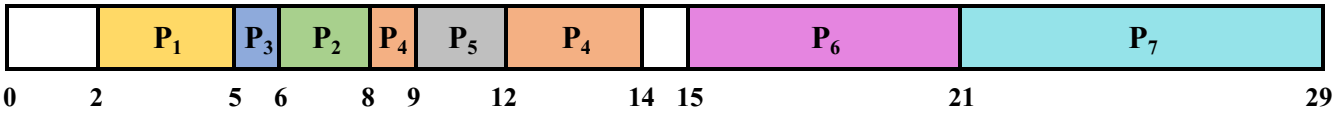
$$\text{throughput} = \frac{\text{number of completed processes}}{\text{the whole time}} = \frac{7}{27} \simeq 0.25$$

$$\text{Average waiting time} = \frac{0 + 0 + (6-4) + (8-7) + (12-9) + 0 + (21-16)}{7} = \frac{11}{7} \simeq 1.57$$

$$\text{Average turnaround time} = \frac{(5-2) + (6-5) + (8-4) + (12-7) + (14-9) + (21-15) + (29-16)}{7} = \frac{37}{7} \simeq 5.28$$

ج) مقدار توان عملیاتی (throughput) در هر دو مورد برابر است. چون تعداد پردازه‌ی برابر در زمان کل برابر به صورت کامل اجرا شده اند. اما همانطور که واضح است، میانگین زمان انتظار و میانگین زمان برگشت در SJF نسبت به FCFS کمتر است.

د) در این حالت الگوریتم SRTF اجرا می‌شود که همان حالت قبضه‌ای (غیر انحصاری) الگوریتم SJF است.



$$\text{throughput} = \frac{\text{number of completed processes}}{\text{the whole time}} = \frac{7}{27} \simeq 0.25$$

$$\text{Average waiting time} = \frac{0+0+(6-4)+0+(8-7)+2+0+(21-16)}{7} = \frac{10}{7} \simeq 1.42$$

$$\text{Average turnaround time} = \frac{(5-2)+(6-5)+(8-4)+(11-9)+(14-7)+(21-15)+(29-16)}{7} = \frac{36}{7} \simeq 5.14$$

ه) مقدار توان عملیاتی (throughput) در هر دو حالت برابر است. چون تعداد پردازه‌ی برابر در زمان کل برابر به صورت کامل اجرا شده اند. اما همانطور که واضح است، میانگین زمان انتظار و میانگین زمان برگشت در SRTF نسبت به SJF کمتر است.

سوال ششم

پاسخ:

آ)

Disk Check	Batch Processing	Time Sharing	Stack Operation
Scan	FIFO	Round Robin	LIFO

ب) برنامه‌ی I/O Bound با احتمال بیشتری context-switch را بصورت داوطلبانه انجام می‌دهد. زیرا برنامه‌ی I/O Bound نیاز به زمان CPU Burst زیادی ندارد و درعوض، دائماً باید درخواست I/O بدهد. پس خودش بصورت داوطلبانه زمان CPU را تحویل می‌دهد و context-switch می‌کند تا پس از برطرف شدن I/O دوباره زمان CPU به آن برنامه اختصاص داده شود. اما برنامه‌ی CPU Bound برای اتمام عملیات خود به بازه‌ی طولانی زمان CPU نیاز دارد و برای حفظ کارایی سیستم نیاز داریم با یک الگوریتم غیر انحصاری، زمان CPU را از آن گرفته و به برنامه‌ی دیگری اختصاص دهیم. پس درمورد CPU Bound با احتمال بیشتری context-switch را بصورت غیرداوطلبانه خواهیم داشت.

ج) وقتی یک پردازه روی CPU اجرا می‌شود، درحین اجرای آن، آخرین اطلاعات مورد استفاده پردازه به حافظه‌ی cache در آن CPU انتقال می‌یابد. درنتیجه، زمانی که پردازه دوباره درخواست دسترسی به این اطلاعات را دارد، به احتمال زیاد این اطلاعات درون cache قرار دارند. اگر بخاطر مکانیزم Load Balance یا دلایل دیگر، این پردازه به پردازنده‌ی جدیدی مهاجرت کند، آنگاه اطلاعات موردنیاز در cache پردازنده‌ی جدید نیستند و این اتفاق باعث صرف هزینه‌ی اضافی جهت انتقال اطلاعات به cache پردازنده‌ی جدید دارد. مفهوم Processor Affinity درواقع به این معنی است که بخاطر دلایل مطرح شده، اجرای یک پردازه را دائما بین پردازنده‌ها جابجا نکنیم تا از موجود بودن اطلاعات پردازه در حافظه‌ی cache حداکثر استفاده را ببریم و هزینه اضافی ندهیم.

دو رویکرد برای رعایت آن به شکل زیر هستند:

۱) Soft Affinity: سیستم عامل تلاش می‌کند تا حد امکان پردازه روی یک پردازنده بماند و اجرا شود. اما هیچ تضمینی نمی‌کند که تحت هر شرایطی این اتفاق بیفتد و ممکن است بنا به شرایط خاص، پردازه را به پردازنده دیگری منتقل کند.

۲) Hard Affinity: در این رویکرد، به پردازه این امکان را می‌دهیم تا زیرمجموعه‌ای از پردازنده‌ها را انتخاب کند و فقط روی همان پردازنده‌ها اجرا شود.

د) برای انواع مختلف پردازه‌ها باید از ۴ صف با اولویت‌های زیر استفاده کنیم: (به ترتیب از بیشترین اولویت به کمترین اولویت)

۱) Real-time Processes: پردازه‌هایی که زمان انجام آنها اهمیت بالایی دارد و ددلاین دارند.

۲) System Processes: پردازه‌هایی که مربوط به سیستم هستند و متعلق به کاربر نیستند.

۳) Interactive Processes: پردازه‌هایی که مربوط به کاربر هستند و response time آنها اهمیت زیادی دارد.

۴) Batch Processes: پردازه‌هایی که interactive نیستند و turnaround time آنها اهمیت زیادی دارد.