امیرعلی فرازمند

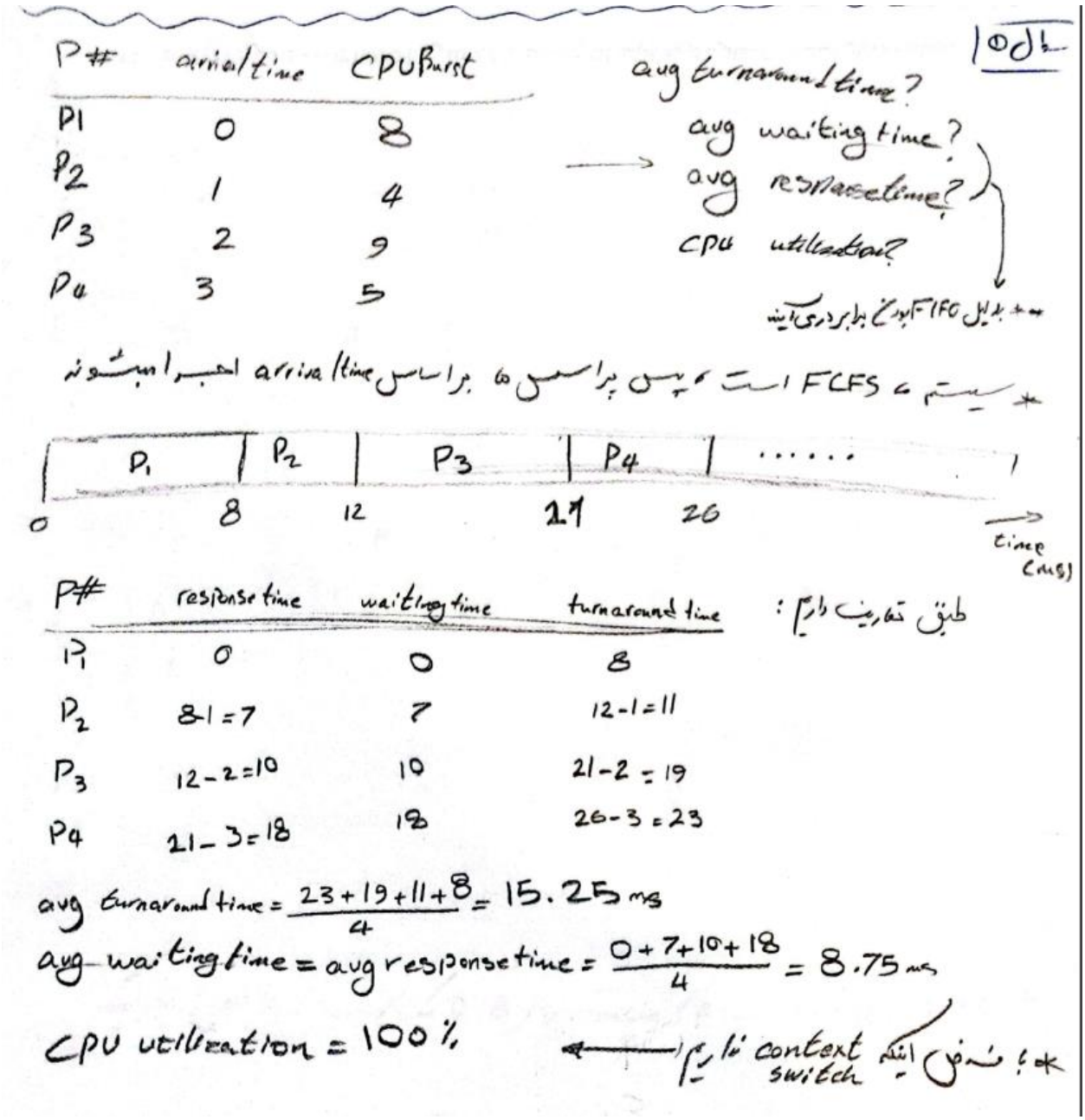
99522329

تمرین فصل 5 O.S

سوال 1

۱. سیستمی شامل ۴ فرآیند هر یک با زمان‌های ورود و اجرای زیر را در نظر بگیرید. اگر در این سیستم از الگوریتم FCFS برای اجرای فرآیندها استفاده شود، پس از تعیین ترتیب اجرای فرآیندها، پارامترهای بهره‌وری CPU، میانگین زمان برگشت، میانگین زمان انتظار و میانگین زمان پاسخ را مشخص کنید. (زمان‌ها بر حسب میلی ثانیه هستند.)

فرآیند	زمان ورود	زمان اجرا
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



در صورت داشتن context switch داریم :

CPU Utilization = 26 / (26 + 3*context switch)

سوال 2

۲. با در نظر گرفتن سیستم سوال قبل و با استفاده از از الگوریتم نوبت گردشی (RR) با دو برش زمانی:

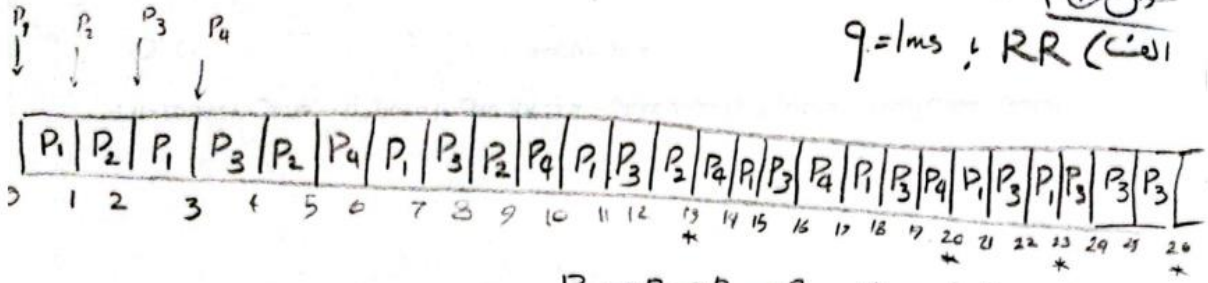
الف) ۱ میلی ثانیه ، ب) ۲ میلی ثانیه

ترتیب اجرای فرآیندها را مشخص نموده و متوسط زمان برگشت، انتظار و پاسخ را محاسبه نمایید.

ج) اگر طول بازه‌ی زمانی برابر با بیشترین CPU Burst درخواست شده در نظر گرفته شود، در واقع کدام الگوریتم اجرا خواهد شد؟

سوال ④

الف) RR ؛ $q = 1ms$



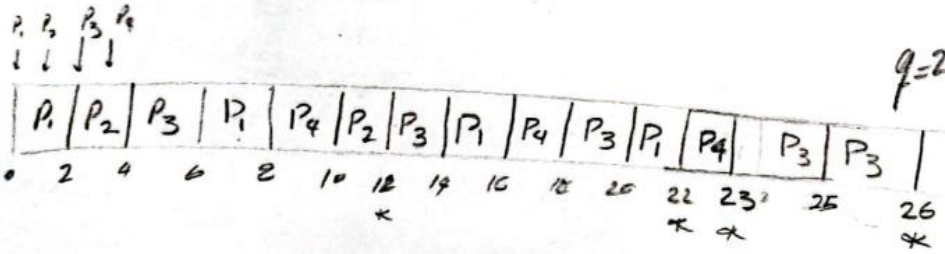
ترتیب اتمام پراسس ها: $P_2 \rightarrow P_4 \rightarrow P_1 \rightarrow P_3$

$$\text{avg waiting time} = \frac{(23-8-0)+(13-4-1)+(26-9-2)+(20-5-3)}{4} = 12.5\,ms$$

$$\text{avg turnaround time} = \frac{(23-0)+(13-1)+(26-2)+(20-3)}{4} = 19\,ms$$

$$\text{avg response time} = \text{avg turnaround time} = 19\,ms$$

ب) RR ؛ $q = 2ms$



ترتیب اتمام پراسس ها: $P_2 \rightarrow P_1 \rightarrow P_4 \rightarrow P_3$

$$\text{avg waiting time} = \frac{(22-0-8)+(12-4-1)+(26-9-2)+(23-5-3)}{4} = 12.75\,ms$$

$$\text{avg turnaround time} = \frac{(22-0)+(12-1)+(26-2)+(23-3)}{4} = 19.25\,ms$$

$$\text{avg response time} = \text{avg turnaround time} = 19.25\,ms$$

ج) اگر q از بیشترین CPU Burst بزرگتر شود مثل این است که ما الگوریتم FCFS را اجرا کردیم؛ پراسس ها به ترتیب ورودشان اجرا میشوند.

سوال 3

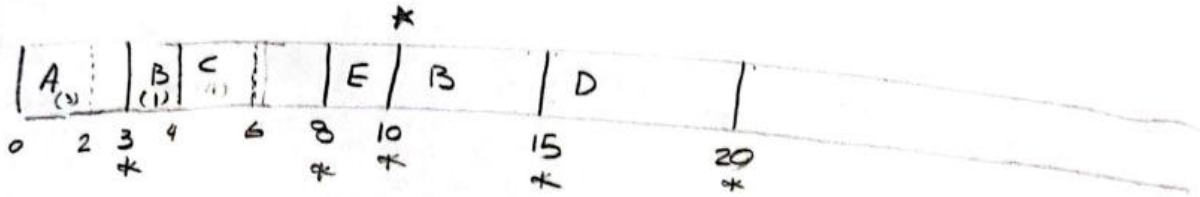
۳. با توجه به زمان ورود و زمان سرویس فرآیندهای زیر، اگر از الگوریتم زمان‌بندی SRTF استفاده شود، متوسط زمان برگشت چقدر خواهد بود؟

فرآیند	زمان ورود	زمان سرویس
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

سوال ۴

P#	arrival time	CPU Burst	
A	0	3	✓
B	2	6 → 5	✓✓✓✓
C	4	4	✓✓
D	6	5	✓✓✓✓✓
E	8	2	✓✓✓

گانت چارت:



$$\text{avg turnaround time} = \frac{(3-0)+(15-2)+(8-4)+(20-6)+(10-8)}{5}$$

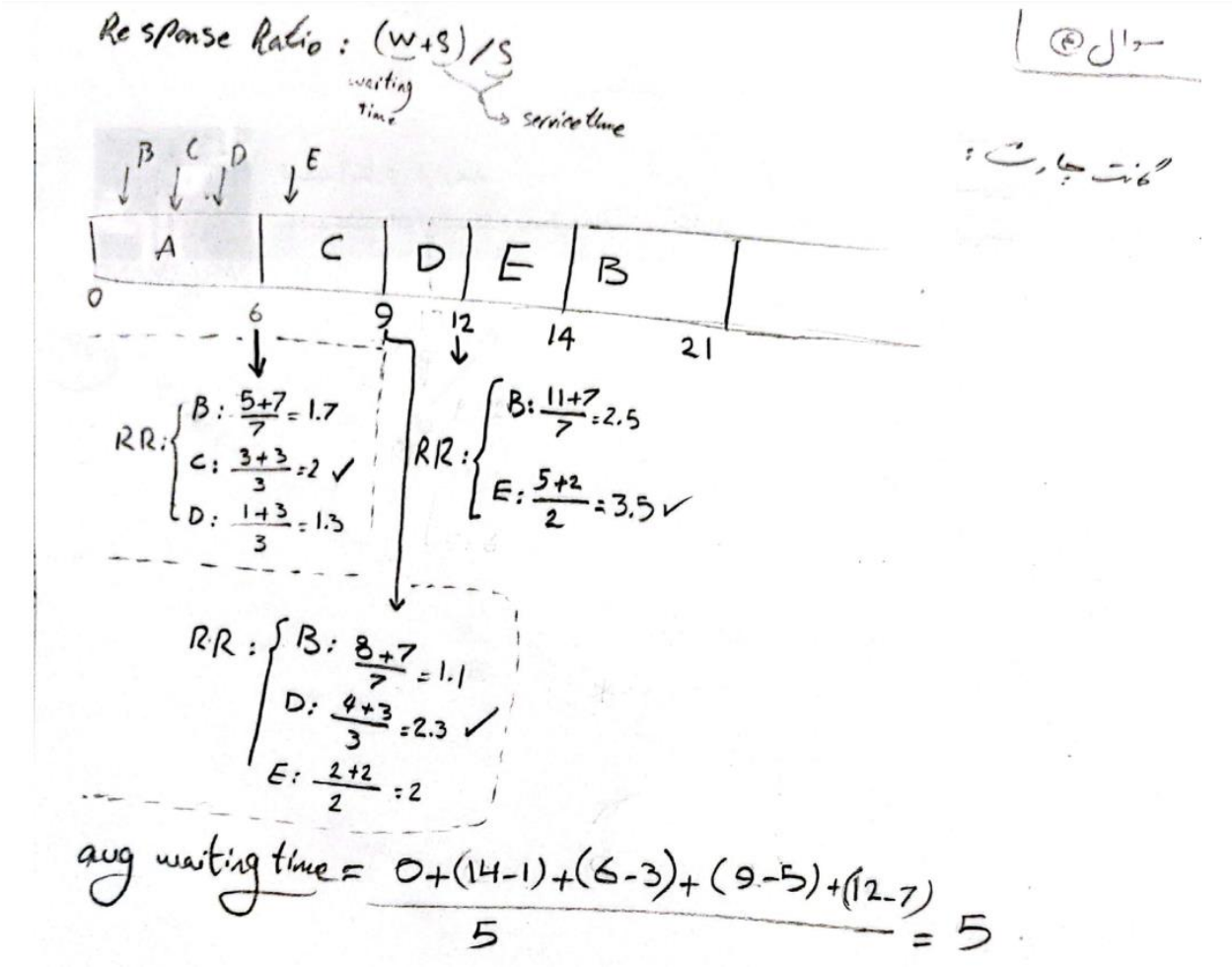
$$= 7.2$$

چون، مشروط براین شده که در لحظه ★ که remaining برای B، D یکی است، B را اول اجرا میکنیم

سوال 4

۴. الگوریتم زمان‌بندی HRRN روی اطلاعات جدول زیر پیاده کرده و گانت چارت مربوطه را رسم نمایید. سپس میانگین زمان انتظار را محاسبه کنید.

فرآیند	زمان ورود	زمان سرویس
A	0	6
B	1	7
C	3	3
D	5	3
E	7	2



[تعریف و مثال از HRRN]

سوال 5

۵. از بین الگوریتم‌های زمان‌بندی گفته شده در کلاس، کدام‌ها باعث starvation می‌شوند؟ توضیح دهید.

در SFJ , SRTF احتمال رخ دادن starvation (گرسنگی) وجود دارد هنگامی که ما در صف پراسس هایمان پراسس(هایی) را داشته باشیم که زمان طولانی ای برای CPU Burst بخواهند.

همچنین در Priority Scheduling هم اگر پراسس ما عدد priority اش بزرگ باشد (اولویت کمتری برای اجرا داشته باشد) ممکن است برایش این اتفاق رخ دهد.

این اتفاق (starvation) هنگامی رخ میدهد که پراسس در صف باشد اما شاید اصلا اجرا نشود.

راه حل این مشکل هم میتوان از Aging استفاده کرد. برای مثال در Priority Scheduling میتوان طی سیکل های مشخص چک کنیم اگر پراسسی باقی مانده بود آنرا یک مرتبه عدد Priority اش را کم کنیم (مثال سر کلاس).

پ.ن : SFJ هم نوعی Priority Scheduling است که اولویت عکس CPU Burst است پس میتوان از پاراگراف اول صرف نظر کرد و گفت جواب سوال اول Priority Scheduling هست .

متن کتاب :

and other, often political, factors.

Priority scheduling can be either preemptive or nonpreemptive. When a process arrives at the ready queue, its priority is compared with the priority of the currently running process. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process. A nonpreemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.

A major problem with priority scheduling algorithms is **indefinit blocking**, or **starvation**. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low-priority processes waiting indefinitely. In a heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU. Generally, one of two things will happen. Either the process will eventually be run (at 2 A.M. Sunday, when the system is finally lightly loaded), or the computer system will eventually crash and lose all unfinished low-priority processes. (Rumor has it that when they shut down the IBM 7094 at MIT in 1973, they found a low-priority process that had been submitted in 1967 and had not yet been run.)

A solution to the problem of indefinite blockage of low-priority processes is **aging**. Aging involves gradually increasing the priority of processes that wait in the system for a long time. For example, if priorities range from 127 (low) to 0 (high), we could periodically (say, every second) increase the priority of a waiting process by 1. Eventually, even a process with an initial priority of 127 would have the highest priority in the system and would be executed. In fact, it would take a little over 2 minutes for a priority-127 process to age to a priority-0 process.

Another option is to combine round-robin and priority scheduling in such a way that the system executes the highest-priority process and runs processes with the same priority using round-robin scheduling. Let's illustrate with an example using the following set of processes, with the burst time in milliseconds:

سوال 6

۶. جدول زیر اطلاعات سه فرآیند را در سیستم نشان می‌دهد. زمان ورود این فرآیندها نشان‌دهنده‌ی زمانی است که فرآیند در صف آماده (ready queue) قرار گرفته است. زمان CPU Burst نیز نشان‌دهنده‌ی زمانی است که هر فرآیند به CPU نیاز دارد. به همین ترتیب I/O Burst نشان‌دهنده‌ی نیاز فرآیند به وسیله‌ی I/O می‌باشد. فرض کنید که یک پردازنده‌ی تک‌هسته‌ای داشته باشیم و مکانیزم سرویس‌دهی در آن RR با برش زمانی ۶ میلی‌ثانیه باشد. سرویس دهی در وسیله I/O هم بر اساس FIFO است. هر فرآیند نیاز دارد برای مدتی از CPU سرویس گرفته و بعد از اتمام سرویس از وسیله I/O استفاده کرده و سپس برای مدتی دیگر از CPU سرویس بگیرد. بعد از سرویس دوم از CPU فرآیند تکمیل شده و از سیستم خارج می‌شود. با توجه به زمان‌های داده شده در جدول زیر، میانگین زمان بازگشت را محاسبه نمایید.

فرآیند	زمان ورود	CPU Burst (ms)	I/O Burst (ms)	CPU Burst (ms)
P_1	3	6	12	15
P_2	8	5	6	3
P_3	0	10	15	6