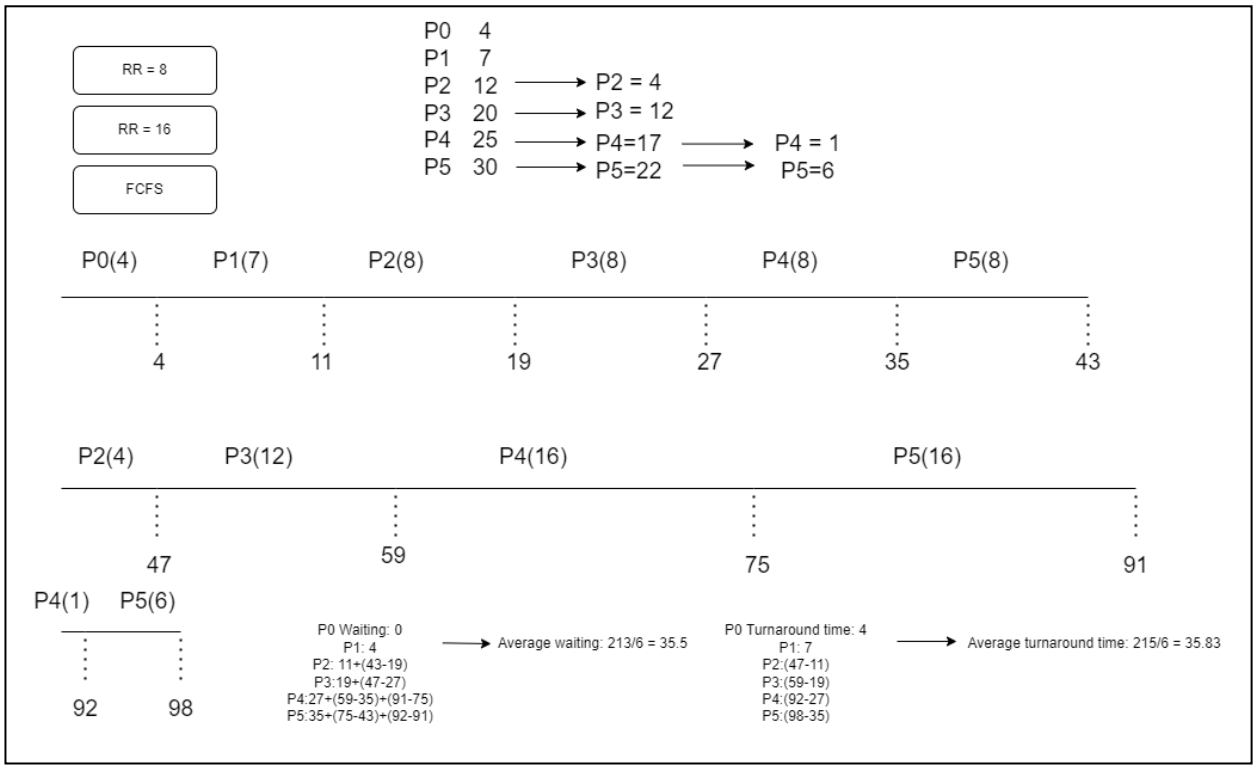
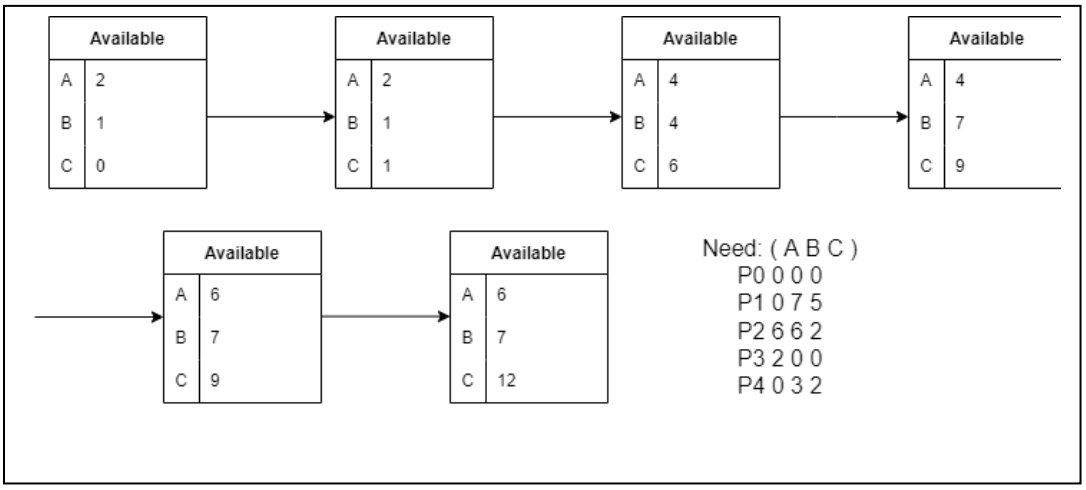
تمرین سری آخر سیستم عامل

1.ساختار جدول TLB عینا مانند Page table است، با این تفاوت که یک فیلد اضافه به نام Page Number نیز دارد؛ زیرا TLB کوچک است و نمی توان از Page number به عنوان Indexایی به جدول استفاده کرد (مانند Page table)
پردازنده دارای قابلیت بازجویی همزمان است که می تواند همزمان چندین سطح از TLBرا جست و جو کرده تا Page number مد نظر را پیدا کند. در واقع روش جست و جو در TLB، چون برخلاف Page table که دارای Direct-Mapping است، دارای Associative mapping است، باید دونه دونه سطر های آن بررسی شود.

.2



.3



الف) مرحله 1: P0 اجرا می شود.
مرحله 2: P3 اجرا می شود.
مرحله 3: P4 اجرا می شود.
مرحله 4: P1 اجرا می شود.
مرحله 5: P2 اجرا می شود.

بنابراین سیستم در حالت امن قرار دارد.

ب) بله، زیرا همانطور که مشخص است در مرحله ای که میخواهیم به P2 سرویس بدهیم، 7 تا از B موجود است.

4. در Paging حافظه به بخش هایی با سایز های مساوی به نام Page frame تقسیم می شود. به تبع از آن، فرآیند ها نیز به بخش هایی به همان سایز تقسیم می شوند و هر یک از این بخش ها، یک Page نام دارد. در هر زمان لیستی از صفحات خالی توسط سیستم عامل نگهداری می شود. زمانی که فرآیند جدیدی میخواهد اجرا شود، سیستم عامل Page های آن را در این Frame ها قرار می دهد. نکته حائز اهمیت این است که نیاز نیست این Page ها در حافظه به صورت پیوسته کنار هم باشند.

در Segmentation هر برنامه به چندین Segment تقسیم می شود. لزومی ندارد که Segment ها یک اندازه باشند اما یک ماکسیمم اندازه ای برای آنها وجود دارد. در هر اجرای برنامه، Segment ها باید به حافظه اصلی آورده شوند اما نکته حائز اهمین این است که نیاز نیست پیوسته در کنار هم باشند.

مقایسه:
صفحه بندی، برای برنامه نویس قابل رویت نیست اما در Segmentation وظیفه برنامه نویس / کامپایلر است که برنامه / داده را به Segment ها تخصیص دهد.

در Segmentation رویه و داده ها را میتوان از هم جدا و برای هر یک حفاظت خاصی در نظر گرفت یا آن ها را به اشتراک گذاشت. اما در صفحه بندی چنین قابلیتی وجود ندارد.

در Paging حافظه بخش بندی می شود اما در Segmentation نه.

Paging باعث تکه تکه شدن داخلی و Segmentation باعث تکه تکه شدن خارجی می شود.

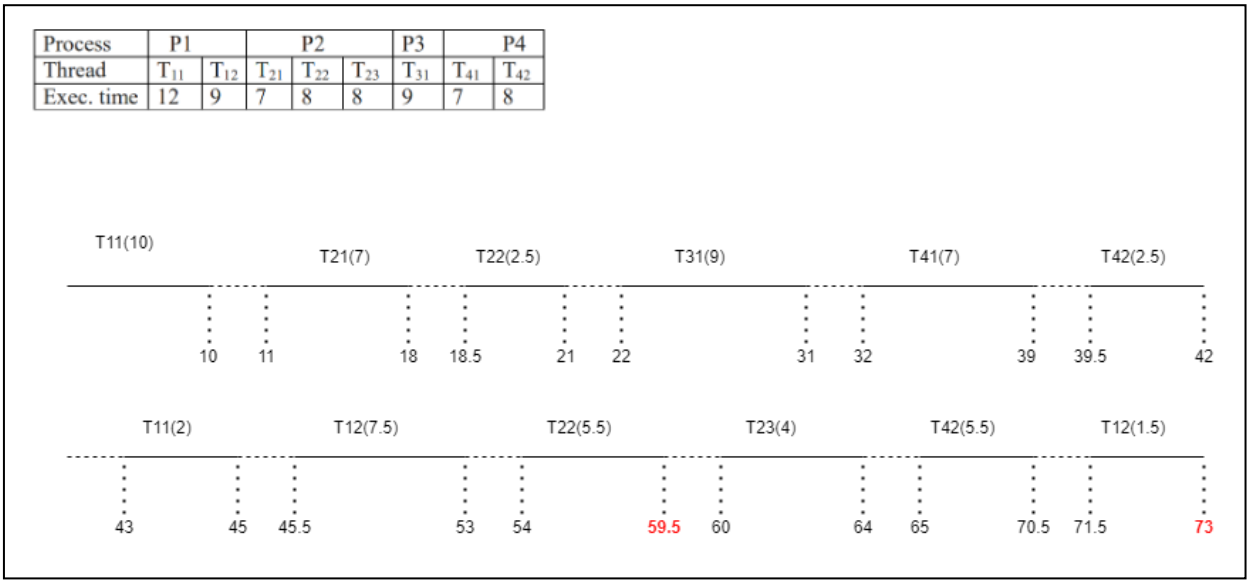
.5

```
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 0 0 2 2 2 2 2 2 2 2 7 7 7
  0 0 0 1 1 1 4 4 4 0 0 0 1 1 1 1 1 1 1
    1 1 0 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0
F F F F   F F F F   F   F   F   F   F
```

12 تا

.6

Process	P1		P2			P3	P4	
Thread	T_{11}	T_{12}	T_{21}	T_{22}	T_{23}	T_{31}	T_{41}	T_{42}
Exec. time	12	9	7	8	8	9	7	8



زمان پایان دو نخ، در شکل بالا مشخص است.

7. آدرس منطقی: 8 صفحه 1024 بایتی
اندازه حافظه فیزیک: 32 قاب.

8 page = 2^3 => 3 Bit for index
1024 byte = 2^10 => 10 bit for offset

طول آدرس منطقی: 13 بیت

32 Frame => 2^5 => 5 bit for index
2^10 => 10 bit for offset

طول آدرس فیزیکی: 15 بیت

.8

Overhead = Se/p + p/2

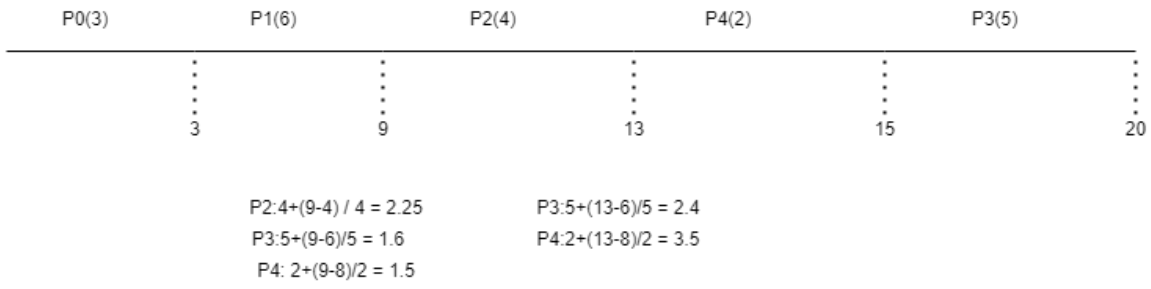
میخواهیم p به گونه ای باشد که Overhead کمینه شود. مشتق میگیریم:

-(Se/p^2) + 1/2 = 0
=> p = sqrt(2se)

بنابراین به راحتی با قوانین ریاضی، فرمول را ثابت کردیم.

.9

	زمان ورود (ms)	زمان سرویس (ms)
P0	0	3
P1	2	6
P2	4	4
P3	6	5
P4	8	2



P2:4+(9-4) / 4 = 2.25
P3:5+(9-6)/5 = 1.6
P4: 2+(9-8)/2 = 1.5

P3:5+(13-6)/5 = 2.4
P4:2+(13-8)/2 = 3.5

همانطور که مشخص است، P3 آخر از همه اجرا می شود.

10. فضای آدرس 2mb است.

8KB = 2^3 * 2^10 = 2^13
2MB = 2 * 2^20 = 2^21

2MB/8KB = 2^8

8 بیتی خواهد بود.

11.هنگامی که یک صفحه را پس از انتقال به دیسک، مجبور می شویم تقریبا بلافاصله دوباره به حافظه اصلی برگردانیم. رخ دادن مکرر این رویداد وضعیتی را به وجود میاورد که به آن کوبندگی می گوییم.

روش های جلوگیری:
محدود کردن چند برنامگی به یک سطح بی خطر به طوری که حاصل جمع مجموعه کارها از اندازه حافظه کمتر باشد. یعنی فرآیندی که نقص صفحه زیادی دارد، معلق کرد. از طرف دیگر اگر تعداد نقص صفحه از یک حدی کمتر است، سطح چند برنامگی بالا برده می شود

استفاده از الگوریتم مجموعه کاری با بسامد خطای صفحه: فقط فرآیند هایی که مجموعه مقیم آنها به اندازه کافی بزرگ است، مجاز به اجرا هستند.

برنامه ها بر حسب اولویت منظم شوند و به همراه هر صفحه نشانه ای از اولویت صاحب صفحه نگه داشته شود. به برنامه ها اجازه داده می شود تا صفحات برنامه های کم اولویت تر را از حافظه بیرون کند.

استفاده از رویکرد L=S:
سطح چند برنامگی بگونه ای تنظیم می شود که میانگین زمان بین خطا ها برابر میانگین زمان لازم برای پردازش یک خطای صفحه باشد.

استفاده از معیار 50:
این روش سعی می کند استفاده از دستگاه صفحه بندی را حدود 50 % نگه دارد. در این نقطه استفاده از پردازنده به حداکثر می رسد.

.12

در هنگام استفاده از TLB:

0.8*(x+100)+0.2*(x+100+100)

بدون استفاده از TLB:

100+100

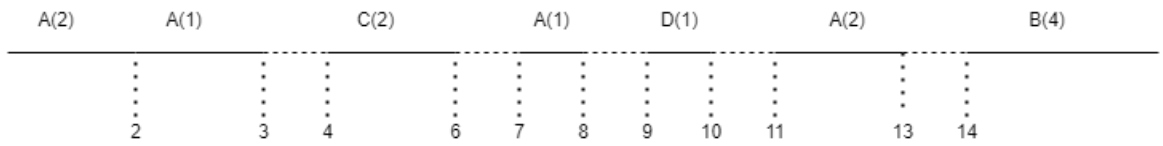
$$\frac{0.8 * (x + 100) + 0.2 * (x + 100 + 100)}{100 + 100} = \frac{80}{100}$$

$$200 * 80 = 80 * (x + 100) + 20 * (x + 100 + 100) => X = 40$$

زمان خواندن TLB برابر است با 40 واحد زمانی.

.13

	زمان ورود (ms)	زمان سرویس (ms)
A	0	6
B	2	4
C	3	2
D	8	1



A Waiting Time: (7-3)+(11-8) = 7
B: (14-2) = 12
C: (4-3) = 1
D: (9-8) = 1

→ Average waiting time: 7+12+1+1 / 4 = 5.25

.14

تکه تکه شدن داخلی: هنگامی که فرآیندی یا صفحه ای به یک بخش از حافظه منتسب می شود به طوری که اندازه آن فرآیند یا صفحه از آن بخش از حافظه کوچکتر است، در این صورت، قسمتی از آن بخش خالی و بلااستفاده می ماند که به آن تکه تکه شدن داخلی می گوییم.

تکه تکه شدن خارجی: زمانی که در میان بخش های تخصیص یافته، بخش هایی (یا Hole هایی) به وجود می آید. هنگامی که فرآیند جدیدی میخواهد اجرا شود که اندازه از از تک تک این Hole ها بزرگتر است اما از مجموع آنها کوچکتر است، نمیتواند اجرا شود.