بسمه تعالی

تمرین شماره ۶

سیستم‌های عامل

محسن کربلائی امینی، ۹۸۲۴۲۱۲۸

دی ۱۴۰۲

سوال ۱:

- نادرست. لزوما این اتفاق نمی‌افتد. برای مثال در الگوریتم FIFO ناهنجاری Belady مشاهده می‌شود. علاوه بر آن رفتار برنامه‌ها پس از مدتی تغییر می‌کند و منجر به افزایش نقص‌ها می‌شود.

- درست. بیت ارجاع نقش هنگامی که صفحه خوانده شده باشد یک می‌شود. هنگامی که تمام بیت‌های ارجاع یک باشند به صورت FIFO رفتار می‌شود تا پیجی که باید خارج شود مشخص شود.

- درست. باید بر اساس زمان مشخص که هر page فراخوانده می‌شود مرتب‌سازی صورت بگیرد تا هر زمان که قرار باشد صفحه‌ای خارج شود به سرعت بهترین صفحه برای خروج انتخاب شود.

- نه لزوما. معمولا برنامه‌ها رفتار تناوبی در قبال این نقص‌ها دارند. یعنی در ابتدا نقص‌ها زیاد خواهد بود و به مرور این نقص‌ها کم می‌شوند. اما ممکن است در برنامه تغییر رویه‌ای برای انجام کار دیگری رخ دهد که مجددا افزایش نقص‌ها و کاهش آن‌ها در طول زمان را شاهد خواهیم بود.

- نادرست. از نظر سرعت به صورت کلی best-fit و first-fit عملکرد بهتری دارند.

- نه لزوما. Best-fit هم مانند Worst-fit باعث ایجاد fragmentation می‌شود با این تفاوت که این مورد برای Best-fit از نوع داخلی می‌باشد و نمی‌توان به صورت کلی مقدار آن را کمتر یا بیشتر از Worst-fit در نظر گرفت.

سوال ۲:

- **Fragmentation**:
  - **External**: حافظه مورد نظر برای ارضای یک درخواست حافظه موجود می‌باشد اما این فضا در دنباله هم نیست و در نتیجه قابل استفاده نمی‌باشد. این مورد در segmentation رخ می‌دهد.
  - **Internal**: حافظه ظرفیت پذیرش درخواست جدید را ندارد به این دلیل که به فرایندهای قبلی فضای بیشتری از آنچه که نیاز دارند داده شده و از آن استفاده نمی‌کنند، در حالی که یک فرایند نیاز به آن دارد. این مورد در page table رخ می‌دهد.
- **Replacement**:
  - **Local**: یک فرایند برای انتخاب قابی که باید خالی شود، از بین قاب‌هایی که به آن تخصیص داده شده انتخاب می‌کند. در این حالت ممکن است کمبود تخصیص حافظه به یک فرایند، اجرای آن را مختل کند. که در این حالت که مدام در حال جابه‌جایی صفحات برای اجرا باشد در حالت thrashing قرار می‌گیرد.
  - **Global**: یک فرایند برای انتخاب قالبی که باید خارج شود، از بین تمام قاب‌ها انتخاب می‌کند و ممکن است قابی که یک فرایند دیگر در اختیار دارد را خارج کند. توان عملیاتی بالاتری دارد اما باید مسائل اولویت و زمان اجرای فرایندها مدیریت شود.

- **Frame**: بخش‌هایی با اندازه مشخص از حافظه فیزیکی

- **Page**: بخش‌هایی با همان اندازه از حافظه منطقی. هر page که در حافظه مجازی قرار دارد ممکن است در یک frame حضور داشته باشد یا نداشته باشد. یک frame هم میتواند هر page ای را درون خود داشته باشد. در هر صورت با چک کردن یک frame به یک page مشخص می‌رسیدم.

- به این دلیل که این محاسبه تاخیر برای سیستم‌عامل کار سختی است.

- آدرسی دهی به دستورالعمل‌ها و داده‌ها به فضای حافظه در ۳ فاز ممکن است انجام شود.

- **Compile Time**: اگر موقعیت حافظه از قبل مشخص باشد کد مطلق تولید می‌شود.

- **Load Time**: اگر موقعیت حافظه در هنگام کامپایل مشخص نباشد باید کد قابل جابه‌جایی تولید شود.

- **Execution time**: آدرس‌دهی به زمان اجرا موکول می‌شود. این مورد نیاز به پشتیبانی سخت‌افزاری دارد.

- بلی. زمان بندی FCFS همروندی کارها را از بین می‌برد و ممکن است مناسب نباشد. زمان‌بندهای دیگر هم مانند SSTF، SCAN و ... هم در این فضا قابل پیاده‌سازی هستند.

سوال ۳:

Page table size $= ۸$ GB

$Page\ size = ۸KB$

$$size\ of\ logical\ address = ?$$

سوال ۴:

- **LRU**

Seq	۵	۲	۸	۴	۲	۶	۳	۵	۷	۸	۵	۰	۳	۲	۰	۳	۴
F۱	۵	۵	۵	۵	۵	۶	۶	۶	۶	۸	۸	۸	۸	۸	۸	۸	۴
F۲	-	۲	۲	۲	۲	۲	۳	۳	۳	۳	۳	۰	۰	۰	۰	۰	۰
F۳	-	-	۸	۸	۸	۸	۸	۵	۵	۵	۵	۵	۳	۳	۳	۳	۳
F۴	-	-	-	۴	۴	۴	۴	۴	۷	۷	۷	۷	۷	۲	۲	۲	۲
Page Fault	*	*	*	*		*	*	*	*	*		*	*	*			*

- قرمز: این خانه از حافظه جایگزین شده است.
- سبز: HIT

- **Second Chance**

Seq	۵	۲	۸	۴	۲	۶	۳	۵	۷	۸	۵	۰	۳	۲	۰	۳	۴
F۱	۵	۵	۵	۵	۵	۶	۶	۶	۶	۸	۸	۸	۸	۸	۸	۸	۴
F۲	-	۲	۲	۲	۲	۲	۳	۳	۳	۳	۳	۰	۰	۰	۰	۰	۰
F۳	-	-	۸	۸	۸	۸	۸	۵	۵	۵	۵	۵	۵	۲	۲	۲	۲
F۴	-	-	-	۴	۴	۴	۴	۴	۷	۷	۷	۷	۳	۳	۳	۳	۳
Page Fault	*	*	*	*		*	*	*	*	*		*	*	*			*

- آبی: بیت ارجاع =۱
- زرد: بیت ارجاع=۰

تعداد نقص‌ها در هر دو الگوریتم برابر با ۱۳ بود.

LRU	Second-chance
نیاز به پیاده‌سازی یک استک برای نگه داری شمارنده هر صفحه دارند. هر آپدیت هزینه زیادی دارد. و این مورد می‌تواند موجب کندی این الگوریتم شود.	پیاده‌سازی ساده‌تر به دلیل فرصت دوباره دادن به یک خانه می‌تواند رخداد پدیده thrashing را کاهش

عملکرد خوب در سناریوهایی که هم‌محلیت زیاد اتفاق می‌افتد.	دهد. ممکن است به اندازه LRU عملکرد خوبی نداشته باشد.

سوال ۵:

$$\boldsymbol{EAT = 10 + \left(\frac{15}{100}\right)(50 - 10) = 16}$$

سوال ۷:

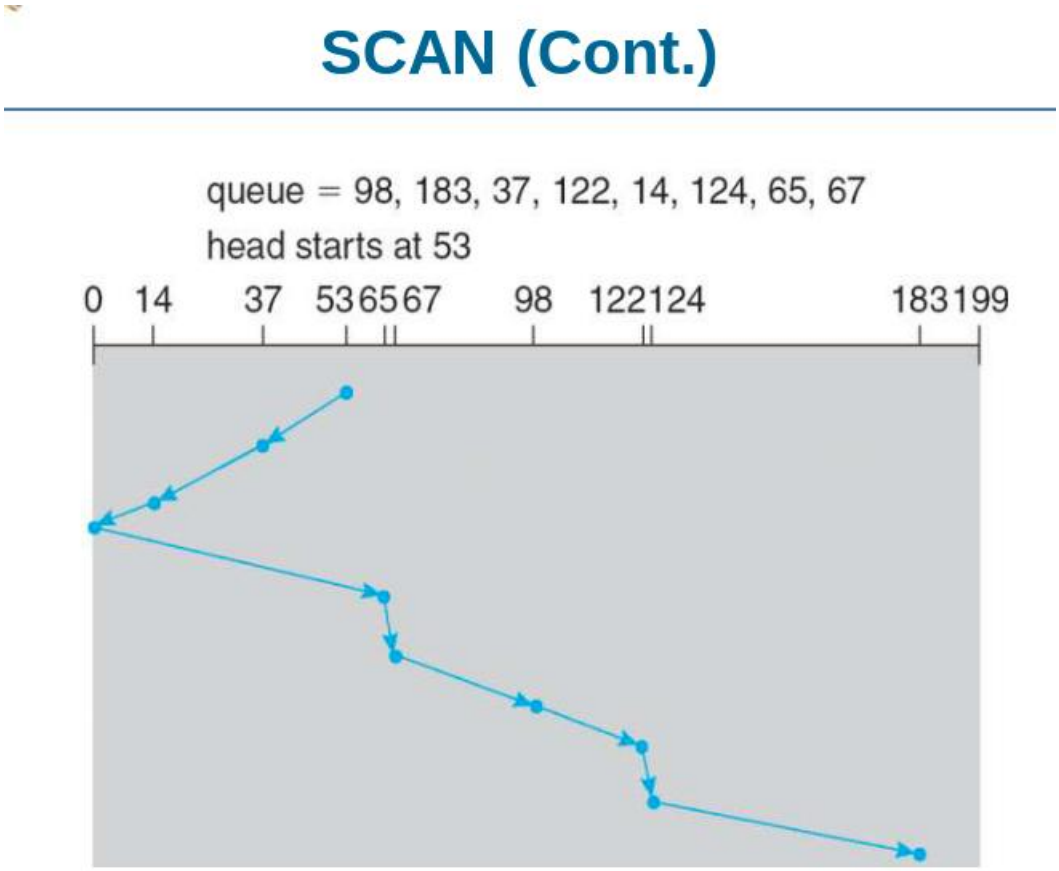
$$\boldsymbol{Serial\ TLB\ EAT = 1 + 15 + 0.1(15) = 17.5}$$

$$\boldsymbol{Parallel\ TLB\ EAT = 1 + 15 + 0.1(14) = 17.4}$$

$$\boldsymbol{Without\ TLB\ EAT = 15 + 15 = 30}$$

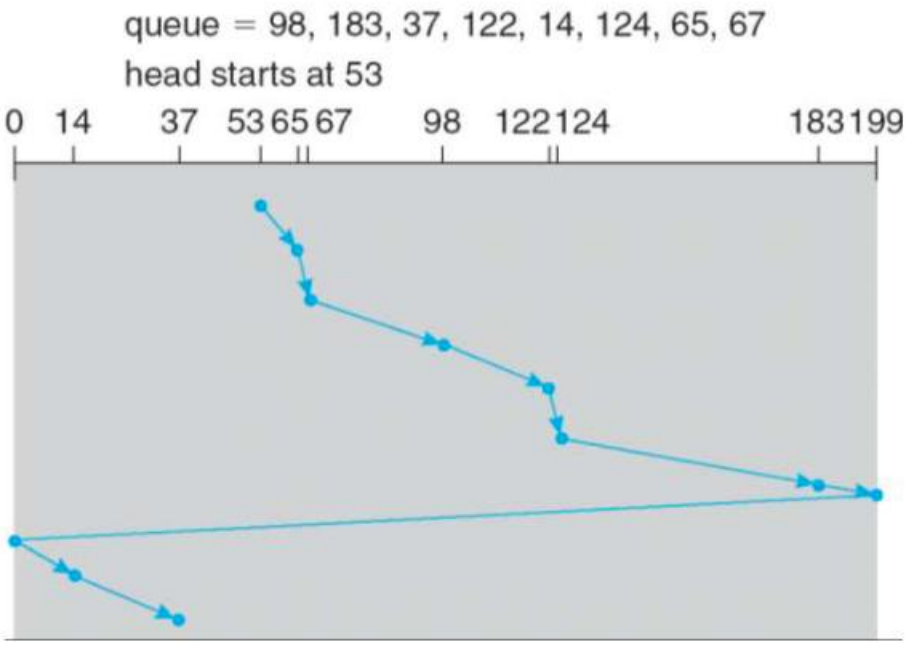
سوال ۹:

- **SCAN**: بازوی دیسک از یک انتهای دیسک شروع به حرکت می‌کند و به سمت انتهای دیگر حرکت می‌کند و درخواست‌ها را تا زمانی که به انتهای دیگر برسد، سرویس‌دهی می‌کند.سپس حرکت سر معکوس شده و مجددا در راه برگشت سرویس‌دهی انجام می‌شود.



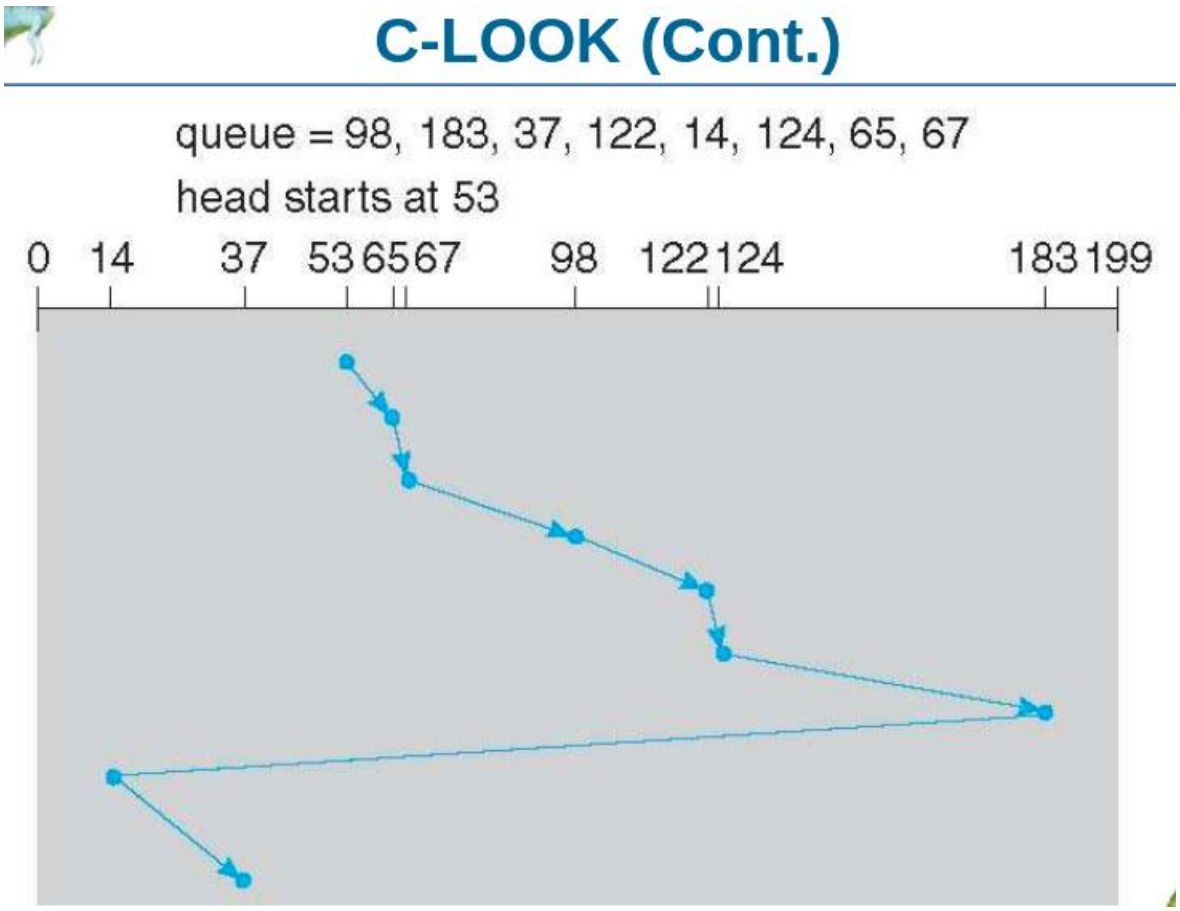
- **CSCAN**: درست مانند SCAN اما به صورت circular عمل می‌کند با این فرق کـه در راه برگشت سرویس‌دهی نمی‌کند.

C-SCAN (Cont.)



- **Look**: بازو تنها تا آخرین محل درخواست‌ها می‌رود و به انتهای صفحه نمیرسد و باز می‌گردد در جهت مخالف.

- **CLook**: درست مانند Look با این تفاوت که Circular عمل می‌کند و در راه برگشت سرویس‌دهی ندارد.



سوال ۱۰:

در این الگوریتم ممکن است در یک ناحیه فرایندهای زیادی دسترسی به حافظه بخواهند و در نتیجه یک فرایند که مدت‌هاست با seek time بالا منتظر پردازش باشد، برای همیشه منتظر بماند و در نتیجه Starve شود.