«گزارش کار پروژه 4 آزمایشگاه»

«Operating system»

کسری کاشانی نژاد **810101490**

البرز محمودیان **810101514**

نرگس بابالار **810101557**

سوال**1-** علت غیرفعال کردن وقفه چیست؟ توابع **popcli** و **pushcli** به چه منظور استفاده شده و چه تفاوتی با **cli** و **sti** دارند؟

پاسخ:

غیرفعال کردن وقفه‌ها به منظور جلوگیری از ایجاد شرایط رقابتی **(Race Conditions)**و اطمینان از اجرای ایمن و همگام کدهای حساس در محیط چندپردازشی یا چندنخی انجام می‌شود. در سیستم عامل**XV6** ، این امر برای محافظت از منابع مشترک یا تغییر وضعیت پردازنده مورد استفاده قرار می‌گیرد.

تفاوت توابع **pushcli** و **popcli** با **cli** و**sti**:

- توابع **cli**و **sti**به ترتیب وقفه‌ها را به صورت مستقیم غیرفعال و فعال می‌کنند. این توابع ساده هستند ولی برای مدیریت پیچیده‌تر کافی نیستند.
- توابع **pushcli**و **popcli**به طور خاص برای مدیریت پشته وقفه‌ها طراحی شده‌اند. این توابع می‌توانند وضعیت قبلی وقفه‌ها را ذخیره **(pushcli)** و بازگردانند **(popcli)**و از این طریق در مدیریت وقفه‌ها هنگام ورود و خروج از بخش‌های بحرانی کمک می‌کنند.

این ویژگی باعث می‌شود وقفه‌ها در شرایط پیچیده‌تر به درستی مدیریت شوند و رفتار سیستم پیش‌بینی‌پذیر باقی بماند.

سوال2- حالات مختلف پردازه ها در XV6 را توضیح دهید. تابع sched چه وظیفه ای دارد؟

پاسخ:

حالات مختلف پردازه‌ها در:XV6

1. **UNUSED(غیرفعال):** پردازه‌ای که در حال حاضر استفاده نمی‌شود و منابع آن آزاد شده‌اند.
2. **EMBRYO(جنین):** پردازه در حال ایجاد شدن است و هنوز کامل نشده است.
3. **SLEEPING(خواب):** پردازه در حال انتظار برای یک رخداد یا منبع است.
4. **RUNNABLE(قابل اجرا):** پردازه آماده اجرا است و منتظر پردازنده می‌باشد.
5. **RUNNING(در حال اجرا):** پردازه در حال اجرای دستورات خود روی پردازنده است.
6. **ZOMBIE(زامبی):** پردازه‌ای که خاتمه یافته ولی هنوز والد آن نتایج را دریافت نکرده است.

وظیفه تابع sched:

تابع **sched**مسئول برنامه‌ریزی و تغییر بین پردازه‌ها است. این تابع پردازنده را از پردازه فعلی گرفته و به پردازه دیگری که در حالت قابل اجرا **(RUNNABLE)** است اختصاص می‌دهد. همچنین وظیفه اطمینان از ذخیره وضعیت پردازه فعلی **(Context Switching)** و بارگذاری وضعیت پردازه جدید بر عهده این تابع است. این عملکرد بخشی از زمان‌بند سیستم **(Scheduler)** است که تضمین می‌کند پردازه‌ها به ترتیب و بر اساس سیاست‌های زمان‌بندی اجرا شوند.

سوال3- یکی از روش‌های سینک کردن این حافظه‌های نهان با یکدیگر روش **Modified-Shared-Invalid**است. آن را به اختصار توضیح دهید. (اسلایدهای موجود در منبع اول کمک کننده شما خواهند بود).

پاسخ:

این روش برای مدیریت همگام‌سازی داده‌ها بین حافظه‌های نهان **(Cache Coherency)** در سیستم‌های چندپردازنده‌ای استفاده می‌شود. سه وضعیت اصلی در این روش وجود دارد:

- **Modified(تغییر یافته):** بلوک حافظه فقط در یک کش وجود دارد و مقدار آن تغییر کرده است. این مقدار هنوز به حافظه اصلی نوشته نشده است.
- **Shared(مشترک):** بلوک حافظه بین چند کش مشترک است و هیچ تغییری در آن صورت نگرفته است.

- **Invalid(نامعتبر)** بلوک حافظه نامعتبر است و نیاز به به‌روزرسانی از حافظه اصلی یا کش دیگری دارد.

این پروتکل تضمین می‌کند که در هر زمان تنها یک نسخه صحیح از داده وجود دارد و عملیات‌های خواندن و نوشتن بین پردازنده‌ها به صورت همگام انجام می‌شود.

سوال4- یکی از روش‌های همگام‌سازی استفاده از قفل‌هایی معروف به قفل بلیت است. این قفل‌ها را از منظر مشکل مذکور در بالا بررسی نمایید.

پاسخ:

قفل بلیت یکی از روش‌های کارآمد برای مدیریت همگام‌سازی بین پردازنده‌ها در شرایطی است که داده‌های مشترک بین کش‌های مختلف پردازنده استفاده می‌شوند. در این روش:

1. **نوبت‌دهی به پردازنده‌ها :قفل بلیت تضمین می‌کند که فقط یک پردازنده در هر لحظه می‌تواند به داده‌های مشترک دسترسی داشته باشد. این کار با شماره‌گذاری بلیت‌ها انجام می‌شود.**
2. **پیشگیری از ناسازگاری حافظه نهان :در شرایطی که پردازنده‌ای داده‌ای را تغییر می‌دهد مثل مقدار 101 در کش(CPU1) ، قفل بلیت دسترسی دیگر پردازنده‌ها مثل (CPU2) را تا زمان به‌روزرسانی کامل حافظه اصلی و کش‌های دیگر محدود می‌کند.**

مزایا برای این مشکل:

- **تضمین همگام‌سازی: با استفاده از قفل بلیت، قبل از این‌که پردازنده دیگر به داده دسترسی پیدا کند، حافظه اصلی و کش‌های دیگر به‌روزرسانی می‌شوند.**
- **جلوگیری از شرایط رقابتی: این قفل مانع از دسترسی همزمان پردازنده‌ها به داده‌های مشترک می‌شود و رفتار سیستم را پیش‌بینی‌پذیر می‌کند.**

نتیجه :استفاده از قفل بلیت مشکل ناسازگاری کش‌ها را کاهش می‌دهد، زیرا تا زمانی که پردازنده تغییرات را کامل نکرده، دیگر پردازنده‌ها نمی‌توانند به داده دسترسی پیدا کنند. این روش در شرایط چندپردازشی با کش‌های جداگانه بسیار مفید است.

سوال5- دو مورد از معایب استفاده از قفل با امکان ورود مجدد را بیان نمایید.

پاسخ:

1 - **پیچیدگی در پیاده‌سازی و مدیریت :**قفل‌های با امکان ورود مجدد **(Reentrant Lock)** نیازمند نگهداری اطلاعات مربوط به مالک قفل **(Thread/Process ID)** و تعداد دفعات ورود هستند. این موضوع باعث افزایش پیچیدگی در پیاده‌سازی و سربار پردازشی می‌شود.

2- **احتمال بروز بن‌بست :**اگر طراحی و مدیریت این نوع قفل‌ها به درستی انجام نشود، ممکن است شرایطی ایجاد شود که یک پردازه قفل را آزاد نکند و دیگر پردازه‌ها منتظر بمانند، که به بن‌بست **(Deadlock)** منجر می‌شود.

سوال6- یکی از ابزارهای همگام‌سازی قفل **Read-Write lock** است. نحوه کارکرد این قفل را توضیح دهید. و در چه مواردی این قفل نسبت به قفل با امکان ورود مجدد برتری دارد.

پاسخ:

قفل **Read-Write Lock** به دو دسته قفل برای خواندن و نوشتن تقسیم می‌شود:

- **حالت خواندن :(Read Lock)** چندین پردازه می‌توانند به‌طور همزمان داده‌ها را بخوانند، به شرطی که هیچ پردازه‌ای در حال نوشتن نباشد.
- **حالت نوشتن :(Write Lock)** فقط یک پردازه می‌تواند داده‌ها را بنویسد و هیچ پردازه‌ای اجازه خواندن یا نوشتن به‌طور همزمان ندارد.

مزایا نسبت به قفل با امکان ورود مجدد:

- **افزایش همزمانی :**در مواقعی که فقط عملیات خواندن انجام می‌شود، چندین پردازه می‌توانند به صورت موازی عمل کنند، که منجر به بهبود عملکرد می‌شود.
- **بهینه برای سناریوهای خواندن زیاد و نوشتن کم :**این قفل برای کاربردهایی که نیاز به خواندن مکرر و نوشتن کمتر دارند، بسیار مناسب‌تر از قفل با امکان ورود مجدد است، زیرا به کاهش سربار پردازشی کمک می‌کند.