به نام خدا

آزمایشگاه سیستم عامل

پروژه چهارم آزمایشگاه سیستم عامل

(همگام سازی)

گروه ۶

اعضای گروه :

محمد سعادتی	**۸۱۰۱۹۸۴۱۰**
محمد عراقی	**۸۱۰۱۹۸۴۳۶**
سید عماد امامی	**۸۱۰۱۹۸۵۲۷**

همگام سازی در xv6

۱) علت غیرفعال کردن وقفه چیست؟ توابع ()pushcli و ()popcli به چه منظور استفاده شده و چه تفاوتی با cli و sti دارند؟

از آنجایی کد های وقفه را نمی توان مسدود کرد، پس برای آنکه اطمینان حاصل کنیم که کد هایی که قرار است اجرا شوند بصورت atomic اجرا بشوند ، باید وقفه ها برای محافظت از ناحیه critical غیر فعال شده باشند.

غیر فعال کردن وقفه توسط دو تابع pushcli و popcli صورت میگیرد. ابتدا بوسیله pushcli وقفه ها را غیر فعال می کنیم و سپس با استفاده از acquire و پس از تمام شدن ناحیه critical و release تابع فراخوانی می شود تا وقفه ها دوباره فعال بشوند.

از sti و cli در هر دو تابع pushcli و popcli استفاده شده است ولی دو تابع popcli و pushcli قابلیت ها دیگری مثل قابلیت شمارش نیز دارند یعنی معلوم است که هر کدام چند بار فراخوانی و اجرا شده اند که این مسئله در کنترل کردن می تواند کمک کننده باشد

۲) مختصری راجع به تعامل میان پردازه ها توسط دو تابع مذکور توضیح دهید. چرا در مثال تولیدکننده/مصرف کننده استفاده از قفل های چرخشی ممکن نیست.

acquiresleep یک پردازنده را روی آدرس قفلی که به آن پاس داده شده است ، تا هنگامی که شرایطی برای در دست گرفتن قفل مورد نظر پیدا نکند sleep می کند. در releasesleep ، با استفاده از ریسه ای که sleeplock را نگه داشته است، وضعیت تمام پردازه هایی که روی چنل قفل sleep کرده اند را بعد از بیدار کردن آنان از SLEEPING به RUNNABLE تغییر می دهد.

در مسئله تولیدکننده/مصرف کننده اگر فقط از spinlock استفاده کنیم یعنی از semaphore استفاده نکنیم، ممکن است حالتی رخ دهد که buffer خالی است اما تولید کننده قفل را برای مدتی در دست نمی گیرد. یعنی نمی توان مطمئن بود که بلافاصله بعد از آزاد شدن قفل توسط مصرف کننده، قفل به مصرف کننده برسد.

۳) حالات مختلف پردازه ها در xv6 را توضیح دهید. تابع ()sched چه وظیفه ای دارد؟

* UNUSED: پردازه استفاده نشده است.
* EMBRYO: هنگامی که پردازه از UNUSED تغییر حالت می دهد وارد این حالت می شود که به معنای آن است که پردازه UNUSED نمی باشد.
* SLEEPING: به معنای آن است که پردازه به منبعی نیاز دارد که هنوز آماده نیست و در cpu نمی باشد بنابراین scheduler از آن استفاده نمی کند.
* RUNNABLE: در این حالت scheduler می تواند پردازه را به cpu اختصاص دهد.
* RUNNING: وضعیتی که به پردازه cpu اختصاص داده شده و در حال اجرا می باشد.
* ZOMBIE: پردازه ای که کار آن به اتمام رسیده ولی پردازه پدر wait را صدا نکرده است و اطلاعات این پردازه همچنان در ptable موجود است.

در اتمام عملیات یک پردازه، تابع ()sched فراخوانی می شود و که در آن context فعلی ذخیره شده و context با scheduler جایگزین می شود.

۴) تغییری در توابع دسته دوم داده تا تنها پردازه صاحب قفل، قادر به آزادسازی آن باشد. قفل معادل در هسته لینوکس را به طور مختصر معرفی نمایید.

طبق فایل mutex.h در لینوکس، تنها پردازه صاحب قفل قادر به آزاد سازی آن می باشند. در نتیجه busy waiting هم رخ نمی دهد. به همین منظور در mutex تغییراتی اعمال میکنیم طوری که تنها صاحب پردازه قادر به انجام عملیات باشد. در واقع باید شرط اینکه چه پردازه ای در حال صدا کردن تابع است بررسی شود. در فایل mutex.h لینوکس یک owner برای چک کردن این مسئله تعریف شده است.

۵) یکی از روش های افزایش کارایی در بارهای کاری چندریسه ای استفاده از حافظه تراکنشی بوده که در کتاب نیز به آن اشاره شده است. به عنوان مثال این فناوری در پردازنده های جدیدتر اینتل تحت عنوان افزونه های همگام سازی تراکنشی (TSX) پشتیبانی میشود. آن را مختصراً شرح داده و نقش حذف قفل را در آن بیان کنید؟

حافظه تراکنشی برای جایگزینی بخش های حیاتی محافظت شده توسط قفل در برنامه های موازی چند رشته ای توسط تراکنش ها (بلوک های اتمی) استفاده می شود. در مقایسه با بخش های بحرانی، تراکنش ها دارای چندین مزیت هستند. اول، برنامه نویسان از استدلال در مورد درستی و عملکرد طرح قفل خود رها می شوند. ثانیاً، ساختارهای داده مشترک تضمین شده است که حتی در صورت خرابی، ثابت نگه داشته می شوند. سوم، تراکنش‌ها می‌توانند به‌طور طبیعی انجام شوند، که توسعه نرم‌افزار موازی قابل ترکیب را بسیار آسان‌تر می‌کند.

اگر تداخلی وجود نداشته باشد، سیستم های TM می توانند چندین تراکنش را به صورت موازی اجرا کنند. اگر دو تراکنش به یک آیتم حافظه دسترسی داشته باشند و حداقل یکی از آنها بنویسد، در تضاد هستند. در این صورت یکی از آنها سقط شده و دوباره راه اندازی می شود. هنگامی که یک تراکنش شروع می شود، برای ذخیره مقادیر قدیمی، که در صورت سقط شدن قابل بازیابی است، به ثبت نقاط بازرسی می پردازد. یک تراکنش نمی تواند مستقیماً در حافظه مشترک بنویسد. در عوض نتایج آن در یک undo-log یا یک بافر نوشتن که توسط سیستم نگهداری می شود ذخیره می شود. به منظور تشخیص تداخل خواندن-نوشتن یا نوشتن-نوشتن، مراجع حافظه ردیابی می شوند. اگر تراکنش بدون درگیری کامل شود، نتایج آن به حافظه مشترک متعهد می شود. اگر تداخلی بین دو تراکنش تشخیص داده شود، یکی از آنها با بازگرداندن چک پوینت ثبت به عقب برمی گردد.

افزونه‌های همگام‌سازی تراکنش‌ها (TSX)، که به آن دستورالعمل‌های جدید همگام‌سازی تراکنش‌ها (TSX-NI) نیز می‌گویند، توسعه‌ای برای معماری مجموعه دستورات x86 (ISA) است که پشتیبانی از حافظه تراکنشی سخت‌افزاری را اضافه می‌کند و اجرای نرم‌افزار چند رشته‌ای را از طریق lock elision سرعت می‌بخشد. . با توجه به معیارهای مختلف، TSX/TSX-NI می‌تواند حدود ۴۰ درصد اجرای سریع‌تر برنامه‌ها را در بارهای کاری خاص و ۴ تا ۵ برابر بیشتر تراکنش‌های پایگاه داده در ثانیه (TPS) ارائه دهد.

TSX/TSX-NI دو رابط نرم افزاری برای تعیین مناطق کد برای اجرای تراکنش ها فراهم می کند. Hardware Lock Elision (HLE) یک رابط مبتنی بر پیشوند دستورالعمل است که برای سازگاری با پردازنده‌های بدون پشتیبانی TSX/TSX-NI طراحی شده است. حافظه تراکنش محدود (RTM) یک رابط مجموعه دستورالعمل جدید است که انعطاف پذیری بیشتری را برای برنامه نویسان فراهم می کند.

Hardware Lock Elision (HLE) دو پیشوند دستورالعمل جدید XACQUIRE و XRELEASE اضافه می کند. این دو پیشوند از کدهای عملیاتی پیشوندهای REPNE / REPE موجود (F2H / F3H) دوباره استفاده می کنند. در پردازنده‌هایی که HLE را پشتیبانی نمی‌کنند، پیشوندهای REPNE/REPE در دستورالعمل‌هایی که XACQUIRE/XRELEASE برای آنها معتبر است نادیده گرفته می‌شوند، بنابراین سازگاری با عقب را ممکن می‌سازد.

حافظه تراکنش محدود (RTM) یک پیاده‌سازی جایگزین برای HLE است که به برنامه‌نویس این انعطاف‌پذیری را می‌دهد تا مسیر کد بازگشتی را مشخص کند که زمانی اجرا می‌شود که تراکنش با موفقیت اجرا نشود. برخلاف HLE، RTM با پردازنده‌هایی که از آن پشتیبانی نمی‌کنند، سازگار نیست. برای سازگاری به عقب، برنامه‌ها برای شناسایی پشتیبانی از RTM در CPU قبل از استفاده از دستورالعمل‌های جدید مورد نیاز هستند.

- برنامه نویس این توانایی را دارد که مناطق کد را برای اجرای تراکنش مشخص کند
- دو رابط نرم افزاری برای مشخص کردن مناطق ارائه می دهد:
  - قفل سخت افزاری Elision (HLE)
    - دستورالعمل های قدیمی XACQUIRE/XRELEASE
    - با سرکوب نوشتن تا قفل کردن، اجرای خوشبینانه اجازه می‌دهد تا قفل برای رشته‌های دیگر آزاد باشد.
    - تراکنش ناموفق از XACQUIRE مجدداً راه اندازی می شود
  - حافظه معاملاتی محدود (RTM)
    - رابط مجموعه دستورالعمل جدید
    - دستورالعمل های XBEGIN، XEND، XABORT
    - به برنامه نویسان اجازه می دهد تا مناطق تراکنشی را به شیوه ای انعطاف پذیرتر از HLE تعریف کنند
    - به برنامه نویس توانایی تعیین مسیر کد بازگشتی را می دهد