پاسخ نامه تمرین دوم درس سیستم‌های عامل

استاد درس: دکتر زرندی

پاییز ۹۹

سوال ۱

الف)

داشتن چند واحد محاسباتی می تواند سرعت اجرای برنامه هارا به طور چشم گیری افزایش دهد. اما انجام این کار مشکلات خاص خودش را دارد. در وهله اول باید برنامه ها را به نحوی نوشت که استفاده از هسته های متعدد به صورت بهینه باشد و تا حد ممکن محاسبات میان همه هسته ها پخش شود و هیچ کدام بیکار نباشد. این با این فرض است که محاسبات یک برنامه یا چند برنامه مستقل از هم باشد.

در صورتی که بخش های مختلف یک برنامه ، یا چند برنامه که همزمان بر روی واحد های پردازشی در حال اجرا است به هم وابستگی داشته باشند ، مشکلات بزرگی به وجود می آید.

یکی از مشکلات عدم سازگاری داده های حافظه نهان است. به فرضی که واحد های پردازنده حافظه های نهان مخصوص خود را داشته باشند ، اگر داده ای در حافظه اصلی یا حافظه نهان واحد های دیگر تغییر کند، این تغییر باید به حافظه نهان واحد های دیگر منتقل شود.

> راه حل: ساده ترین راه حل این است که امکان تغییر یک خانه از حافظه اصلی را به بیشتر از یک واحد ندهیم. غیر از آن می توان یک راه ارتباطی میان حافظه های نهان واحد ها ایجاد کرد تا در صورت تغییر یک سطر ، این تغییر به قسمت های دیگر هم منتقل شود. همچنین می توان حافظه اصلی همه واحد های پردازشی را یکی کرد.

یکی دیگر از مشکلات ممکن ، دسترسی همزمان چند واحد پردازنده به یک فایل یا قطعه کد یا یک آدرس است ، برای این مشکل که بعدا در درس به تفصیل به آن می پردازیم (تحت عنوان dining philosophers) راه حل های گوناگونی وجود دارد. آنچه شهودی و منطقا به ذهن می رسد استفاده از یک مکانیزم قفل کردن آن قسمت از داده ای است که نمیخواهیم بیش از یک واحد به آن دسترسی داشته باشد. همچنین می توان همه تغییرات لازم را در یک حافظه محلی انجام داد و در زمان مناسب به محل اصلی منتقل کرد.

نکته : موارد ذکر شده تنها مشکلات قابل رخ دادن نیست و مشکلات دیگری هم مطرح هستند. لذا هرکدام که درست اشاره شده باشند و راه حل منطقی برای حل آن ارائه شده باشد قابل قبول است.

سوال ۱

ب)

یک پردازنده که در داخل خود بیش از یک هسته دارد را پردازنده چند هسته‌ای می نامند.

یک کامپیوتر که بیش از یک پردازنده دارد که از یک bus و حافظه اصلی و دستگاه های i/o استفاده می کنند را سیستم چندپردازنده ای می نامند.

از لحاظ معماری :

یک پردازنده چند هسته ساختار ساده تری از سیستم چندپردازنده ای دارد. اتصالات کمتر و ممکن است ساختار cache هم ساده تر باشد.

عموما هسته ها در یک چیپ پیاده سازی می شوند اما در چندپردازنده‌ای ، چیپ ها جدای از هم هستند.

مصرف برق در چند پردازنده‌ای بسیار بیشتر از چند هسته ای است. (و لذا گران تر است)

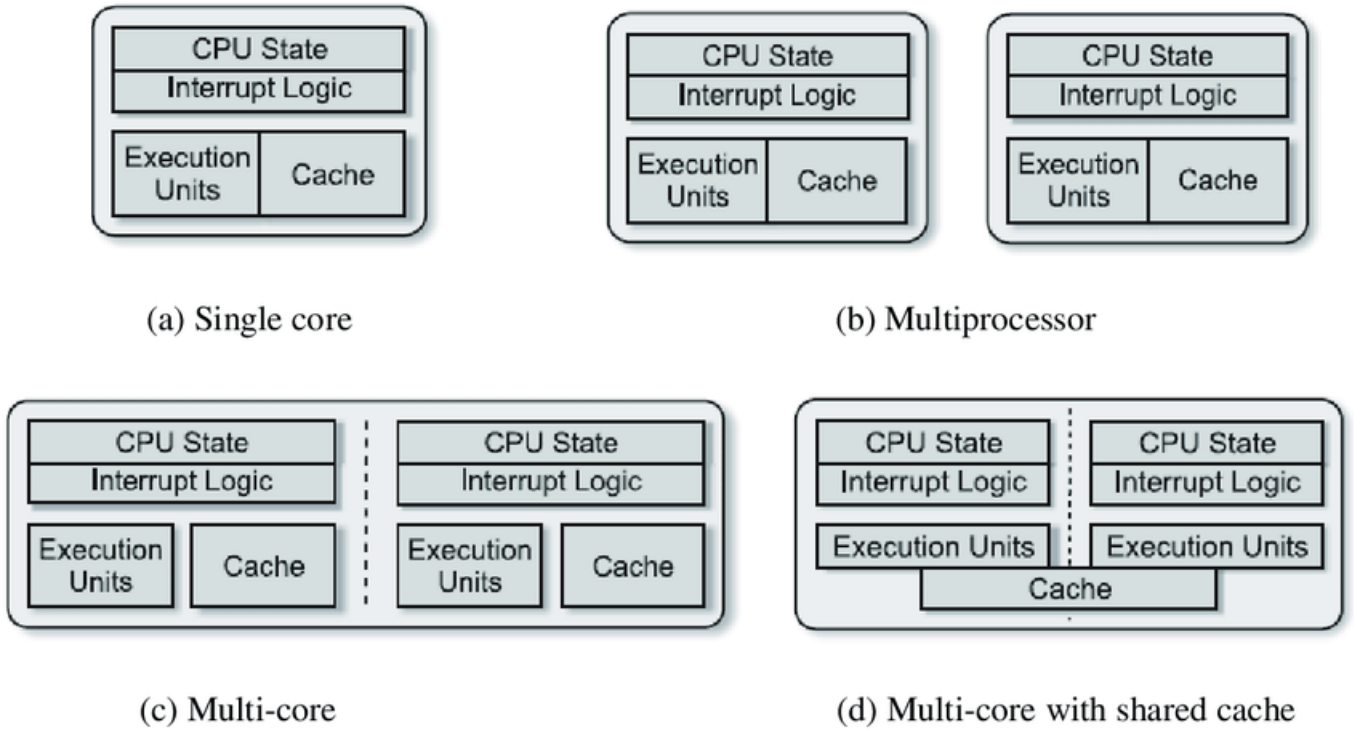
از لحاظ عملکرد :

یک پردازنده چند هسته ای می تواند یک برنامه را سریع تر اجرا کند ، اما چند پردازنده‌ای می تواند چند برنامه را سریع تر اجرا کند.

تحمل خطا در چند پردازنده‌ای بیشتر است چرا که اگر یک پردازنده خراب شود ، سیستم از کار نمی افتد اما در چند هسته‌ای چنین نیست.

پیکر بندی برنامه نویسی با چند هسته‌ای ساده است اما در مورد چند هسته ای باید آن را متناسب با شرایط تنظیم کرد.

مقایسه انواع معماری تک هسته ای ، چند هسته ای و چند پردازنده ای



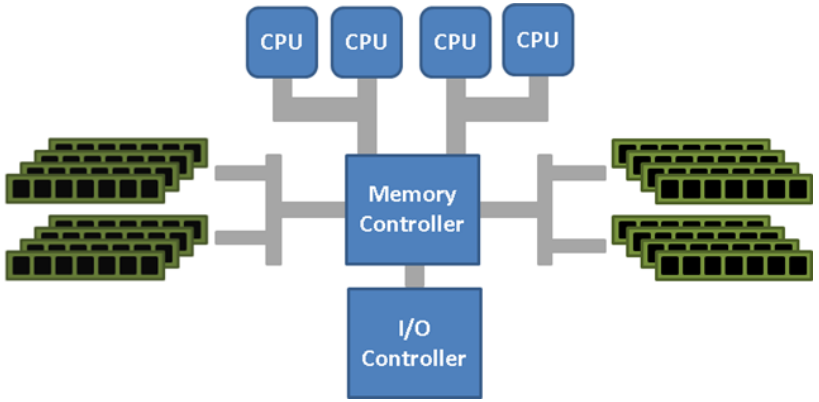
سوال ۲

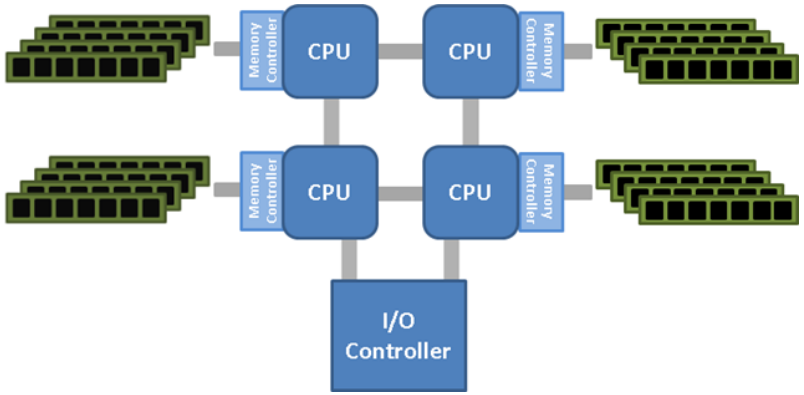
به طور خلاصه UMA و NUMA دو معماری برای پردازنده هایی هستند که به صورت حافظه شراکتی کار می کنند. تفاوت آنها در نحوه توضیع شدن حافظه و سخت افزار است.

در مدل اول (UMA) یک حافظه اصلی میان یک یا چند پردازنده به اشتراک گذاشته می شود. زمان دسترسی همه پردازنده ها به حافظه اصلی یکی است و مقدار حافظه در دست همه آنها یکی است. همچنین حافظه اصلی تنها یک کنترلر دارد که آن را به همه پردازنده ها متصل می کند. سرعت این معماری در حد معمولی می باشد و برای فعالیت های عمومی و همه روزه مناسب می باشد.

در مدل دوم (NUMA) هر پردازنده حافظ اصلی خودش را دارد و لذا هر کدام ازین حافظه ها کنترلر مخصوص خود را دارند. زمان دسترسی پردازنده ها به حافظه خودشان سریع تر از حالت قبلی است (هرچند دسترسی پردازنده ها به حافظه پردازنده دیگر می تواند کند تر باشد. برای همین نحوه ذخیره داده ها اهمیت خیلی زیادی دارد). در این حالت سرعت بیشتر از حالت قبلی است و برای کارها حساس و real-time مناسب تر می باشد. همچنین این طراحی عموما گران تر است.

تصویر سمت راست معماری NUMA و سمت چپ معماری UMA می باشد.





سوال ۳

امروزه همه سیستم ها از دو حالت کلی user و kernel پشتیبانی می کنند. اما بسیاری از آنها برای تعیین حالت اجرای کد بیش از یک بیت دارند که در نتیجه تعداد حالات (به اصطلاح ring) بیشتری را ایجاد می کند که دستورات تحت آن قابلیت اجرا داشته باشند.

برای مجازی سازی یک سیستم عامل ، روی یک سیستم عامل دیگر لازم است که اجرای دستورات حساس تحت نظارت سیستم عامل میزبان رخ دهد به همین منظور همه این دستورات در یک ring بالاتر از صفر (که حالت kernel است) انجام می شوند. و به این شکل دسترسی سیستم عامل مهمان محدود تر خواهد بود. این فرایند برای همه سیستم عامل هایی که ممکن است به صورت موازی در حال اجرا باشند قابل انجام است (تا جایی که امکان تعریف حالات جدید باشد)

پس هر دستوری که برای اجرا نیاز به دسترسی ring 0 یا kernel داشته باشد باید با api های تعریف شده درخواست اجرای درخواست خود را کند.

سوال ۴

الف)

تقسیم بر صفر: از نوع وقفه برنامه است.

مراجعه به آدرس غیرمجاز : از نوع وقفه برنامه است.

خطای سرریز حافظه : از نوع وفقه برنامه است.

اتمام کار DMA: از نوع وقفه i/o

ب)

- خیر ، HLT یک دستور در زبان اسمبلی است که پردازنده را به حالت بیکار یا idle می برد.
- اگر سیستم کاری برای انجام نداشته باشد یا منتظر عملیات i/o باشد ، برای آنکه ناخواسته دستوری اجرا نشود و همچنین مصرف برق سیستم کاهش یابد ،سیستم عامل فعالت پردازنده را موقتا متوقف می کند.
- در صورتی که یک وفقه رخ دهد ،پردازنده از حالت بیکار خارج می شود.

سوال ۵

اگر همیشه به درخواست با اهمیت بالاتر برسیم ، احتمال آن وجود دارد که هرگز درخواست های دیگر انجام نشوند.

برای سیستم تک هسته ای: می توان یک قاعده تعریف کرد که سیستم بازه های زمانی مشخصی (متناسب با اهمیت درخواست ها) برای رسیدگی به هر یک اختیار کند. یعنی مثلا به ازای هر ۳۰ میلی ثانیه ای که درخواست های نوع A را انجام داد ، ۲۰ میلی ثانیه درخواست نوع B و ۱۰ میلی ثانیه درخواست های نوع C را انجام دهد. (در صورت وجود) به این ترتیب به این مشکل که هیچ وقت نوبت به درخواست B یا C نرسد هم نمی خوریم.

برای سیستم چند هسته‌ای : می توان متناسب با تعداد هسته ها ، آنها را میان درخواست ها پخش کرد. مثلا اگر ۸ هسته بود ، ۴ هسته برای درخواست های نوع A و ۳ هسته برای درخواست نوع B و ۱ هسته برای درخواست نوع C . یا اگر ۴ هسته بود ،دو هسته برای A و یک هسته برای B و یک هسته به صورت اشتراکی برای B و C.

به این ترتیب باز هم مطمئن می شویم با مشکل بالا رو به رو نخواهیم شد.