به نام خدا

محمدمهدی آقاجانی

تمرین ششم سیستم عامل

استاد طاهری جوان

تمرین 1:

دربارهٔ مدیریت حافظه مجازی در ویندوز تحقیق کنید

مدیر حافظه مجازی (W2K) windows 2000 چگونگی تخصیص حافظه و اجرای صفحه بندی ها را کنترل می کند.مدیر حافظه طوری طراحی شده است که بتواند روی کامپیوترهای مختلف اجرا گردد و از صفحه های ۴ تا ۴ کیلوبایت استفاده کند.کامپیوترهای intel وpoweraPC و MIPS دارای ۴ کیلو بایت در هر صفحه و کامپیوترهای DEC Alpha دارای ۸ کیلو بایت در صفحه می باشند.

ترجمه آدرس مجازی W2K

هر فرآیند کاربر در W2K فضای آدرس ۳۲ بیتی مربوط به خود را را مشاهده میکند که تا ۴ گیگابایت حافظه را اجازه دسترسی میدهد.به طور پیش فرض قسمتی از این حافظه برای سیستم عامل در نظر گرفته شده است. پس در واقع هر کاربر ۲گیگابایت از فضای آدرس مجازی موجود را در اختیار دارد و همه فرآیندها در ۲ گیگابایت فضای سیستم عامل شریک هستند. این انتخاب وجود دارد که فضای کاربر به ۳ گیگابایت افزایش یابد و یک گیگابایت برای سیتم باقی بماند.مستندات W2K این خصوصیت را برای حمایت از کاربردهای خواهان حافظه زیاد ذکر می‌کنند. کاربردهایی چون "حمایت از تصمیم‌گیری " یا "کاوش داده‌ها" که بر روی سرویس دهندگانی با چند گیگابایت حافظه RAM اجرا شده و استفاده از فضای آدرس بزرگتر می‌تواند به طور قابل توجهی کارآمدی آنها را بهبود ببخشند.

فضای آدرس پیش فرض که توسط کاربر دیده میشود شامل بخش های زیر است:

0X00000000 تا 0x0000FFFF : جهت کمک به برنامه ساز برای بدست آوردن تخصیص های اشاره گر تهی ، کنار گذاشته است.

- 0X00010000 تا 0X7FFEFFFF : فضای آدرس کاربر. این فضا به صفحه هایی تقسیم شده که ممکن است به داخل حافظه اصلی بار شوند.
- 0X7FFF0000 تا 0X7FFFFFFF : صفحه حفاظت شده ای که در دسترس کاربر نیست.این صفحه ، بررسی اشاره‌گرهای خارج از محدوده را برای سیستم عامل آسان تر می‌سازد.
- 0X80000000 تا 0XFFFFFFFF : فضای آدرس سیستم. این فضای ۲ گیگابایتی برای مجری W2K ، ریزهسته و گرداننده‌های دستگاه‌ها به‌کار می‌رود.

صفحه بندی W2K :

کلیات آن شبیه به صفحه بندی عمومی است و هر برنامه میتواند ۲ گیگابایت حافظه را در اختیار داشته باشد و هر صفحه یکی از حالات زیر را داراست :

- موجود: صفحه‌هایی که در حال حاضر به‌وسیله این فرآیند، مصرف نشده‌اند.
- رزرو: مجموعه‌ای از صفحه‌های متوالی که مدیر حافظه مجازی آنها را برای فرآیند کنار می‌گدارد ولی تا زمانی که مورد استفاده قرار نگرفته‌اند از سهمیه حافظه آن فرآیند کم نمی‌کند.هنگامی‌که فرآیندی نیاز به نوشتن در حافظه داشته‌باشد ، برخی از صفحه‌های رزرو شده به آن فرآیند داده می‌شود.
- متعهد: صفحه‌هایی که مدیر حافظه مجازی برای آنها صفحه هایی از پرونده صفحه‌بندی را کنار گذاشته است.(به عنوان مثال پرونده صفحه‌بندی پرونده‌های دیسکی که مدیریت حافظه مجازی صفحه‌هایی را که از حافظه اصلی برمی‌دارد، در آن می‌نویسد.)

تمایز بین حافظه رزرو و حافظه متعهد مفید است زیرا اول مقدار فضای دیسک کنارگذاشته شده برای فرآیند خاص را حداقل و آن را برای سایر فرآیندها آزاد می‌سازد ؛ ثانیا به یک نخ یا فرآیند این توان را می‌دهد که مقدار حافظه موردنیاز خود را اعلام کند و به فوریت به آن تخصیص یابد.
طرح مدیریت مجموعه مقیم که توسط W2K به کار می‌رود دارای خصوصیات تخصیص متغیر و دیدگاه محلی است. زمانی که فرآیندی برای اولین بار فعال می‌شود ، تعداد مشخصی از قاب صفحه‌های حافظه اصلی به عنوان مجموعه کاری به آن تخصیص می‌یابد. اگر فرآیندی به صفحه ای مراجعه کند که در حافظه نباشد، یکی از صفحه‌های مقیم آن فرآیند به خارج مبادله شده وصفحه جدید به داخل آورده می شود. مجموعه‌های کاری فرآیند فعال توسط قراردادهای کلی زیر تنظیم می‌شوند:

تمرین 2:

الگوریتم جایگزینی صفحه «ساعت دو عقربه ای» که در سیستم عامل یونیکسSVR4 استفاده می شود را بررسی کنید

الگوریتم جایگزینی صفحه مورد استفاده در SVR4 بهبودیافته الگوریتم سیاست ساعت است که به نام الگوریتم ساعت دوعقربه ای شناخته می شود.الگوریتم مزبور، بیت مراجعه در مدخل جدول صفحه را برای هر صفحه ای که در داخل حافظه واجد شرایط مبادله به خارج باشد (قفل نشده باشد) به کار می برد . زمانی که برای اولین بار صفحه به داخل حافظه بار می شود این بیت صفر می شود و هر گاه صفحه برای خواندن یا نوشتن مورد مراجعه قرار گیرد، یک می گردد .یک عقربه در الگوریتم ساعت (عقربه جلو) لیست صفحه های واجد شرایط را مرور کرده و بیت مراجعه هر صفحه را صفر می کند . بعدا عقربه عقبی همان لیست را مرور کرده و بیت های مراجعه را آزمایش می کند ، اگر این بیت یک باشد به این معنی است که صفحه مزبور از زمان مرور عقربه جلویی مورد مراجعه واقع شده است. اط اینگونه قاب ها صرفنظر می شود . اگر بیت مزبور هنوز صفر باشد به این معنی است که در فاصله زمانی بین ملاقات عقربه جلویی و عقربه عقبی ، به صفحه مزبور مراجعه نشده است. این صفحه ها در لیست مبادله به خارج قرار می گیرند. دو پارامتر عملکرد این الگوریتم را تعیین می کنند:

- نرخ مرور: نرخی که بر اساس آن دو عقربه لیست صفحه ها را برحسب صفحه در ثانیه مرور می کنند.
- فاصله عقربه ها: فاصله بین عقربه جلویی و عقربه عقبی

مقادیر پیش فرض این دو پارامتر بر اساس مقدار حافظه فیزیکی در زمان راه اندازی تعیین می گردد. می توان پارامتر نرخ مرور را با تغییر شرایط ، عوض نمود. این پارامتر به طور خطی بین مقادیر slowscan و fastscan با تغییر مقدار حافظه آزاد، که بین دو مقدار lotsfree و minfree است، تغییر می کند. به عبارت دیگر، با کوچک شدن مقدار حافظه آزاد عقربه های ساعت با سرعت بیشتری حرکت می کنند تا صفحه های بیشتری را آزاد نمایند. پارامتر فاصله عقربه ها همراه با پارامتر نرخ مرور چهارچوب فرصت استفاده از یک صفحه را مشخص می سازد.

تمرین ۳:

درباره روال مدریتی حافظه و مبتنی بر معماری پینتوم ازشرکت اینتل تحقیق کنید.

تا آمدن سیستم x86-64 سیستم حافظه‌ی مجازیx86 در بسیاری از جهات از جمله وجود هردوی تقسیم‌بندی و صفحه‌بندی در آن شبیه مولتیکس بود در حالی که در مولتیکس دارای 256K قطعه‌ی مجزا که هر یک شامل64K کلمات 36 بیتی است وx86 دارای 16K قطعه‌ی مجزاست که هر یک شامل 1میلیارد کلمه‌ی 32 بیتی است می‌باشد.

اگرچه در x86 تعداد قطعات کمتر است ولی این بزرگتر بودن قطعات خود نقطه‌ی قوتی برای این سیستم بوجود می‌آورد.

بسیاری از برنامه‌ها نیاز به بیش از ۱۰۰۰ قسمت دارند در حالی که برخی دیگر نیازمند بخش‌هایی با اندازه‌ی بزرگتر هستند.

دو شکل زیر به معرفی بخش‌هایی از سیستم حافظه مجازی مولتیکس می‌پردازد:

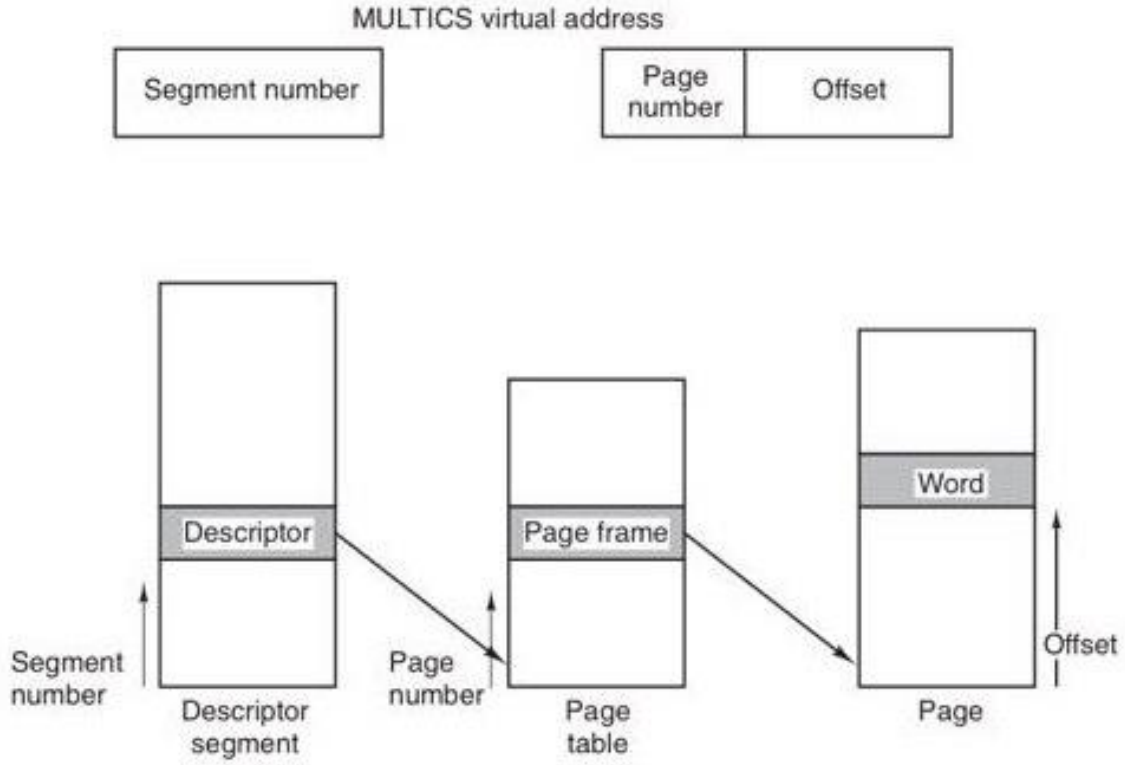


Figure 3-36. Conversion of a two-part MULTICS address into a main memory address.

Comparison field					Is this entry used?
Segment number	Virtual page	Page frame	Protection	Age	
4	1	7	Read/write	13	1
6	0	2	Read only	10	1
12	3	1	Read/write	2	1
					0
2	1	0	Execute only	7	1
2	2	12	Execute only	9	1

Figure 3-37. A simplified version of the MULTICS TLB. The existence of two page sizes made the actual TLB more complicated.

این بخش‌بندی از نسخه‌ی x86-64به بعد دیگر منسوخ شد. البته برخی از بقایای مکانیسم بخش‌بندی قدیمی هنوز در حالت بومی x86-64 فقط به منظور سازگاری با سایر نسخه‌ها وجود دارد .

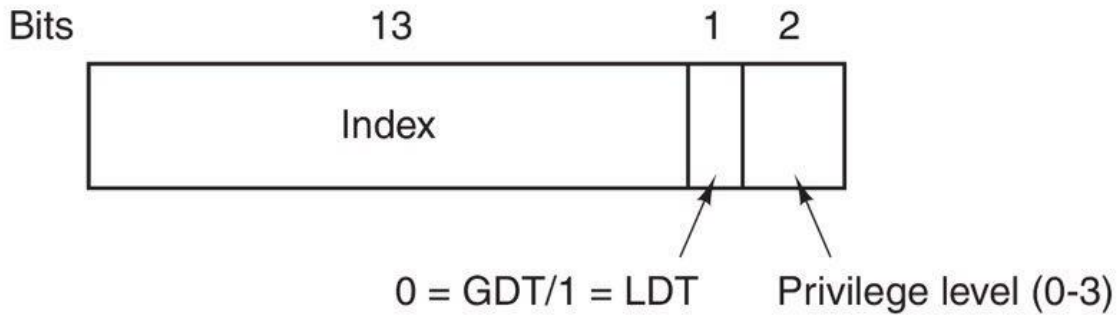
قلب حافظه مجازی x86 ا متشکل از دو جدول، به نام LDT (جدول توصیف محلی) و GDT (جهانی توصیف جدول) است.

هر برنامه LDT خود را دارد، اما یک GDT ،که توسط تمام برنامه به اشتراک گذاشته شده بر روی کامپیوتر وجود دارد.

LDT توصیف بخش محلی برای هر یک از برنامه ها، شامل کد آن، داده، پشته، و غیره، در حالی که GDT توصیف بخش‌بندی سیستم، از جمله خود سیستم عامل است.

برای دسترسی به یک بخش، یک برنامه های x86 در مرحله‌ی اول یک انتخابگر برای آن بخش در یکی از شش ثبات بخش دستگاه بارگذاری می‌کند.در طول اجرا، ثبات CS حاوی انتخابگر برای بخش کد و ثبات DS حاوی انتخابگر برای بخش داده است.سایر ثبات‌ها اهمیت کمتری نسبت به موارد ذکر شده دارند.

شکل زیر یک انتخابگر x86را نمایش می‌دهد.



یکی از بیت های انتخابگر می گوید که آیا بخش محلی و یا جهانی است .سیزده بیت ه‌ای دیگر مشخص کننده عدد ورودی LDTیا GDT است. دو بیت بعد مربوط به اطلات مربوط به حفاظت است.

دادن مقدار 0 به توصیف کننده ممنوع است. زیرا ممکن است با خیال راحت به یک ثبات سگمنت نشان دهد که ثبات سگمنت در حال حاضر در دسترس نیست. این باعث می شود که بتوان از این تکنینک به صورت یک تله استفاده کرد.

وقتی که یک سلکتور در ثبات سگمنت بارگذاری می شود، توصیف گر مربوطه از LDT یا GDTخوانده می شود و در ثبات های میکروپروگرمم ذخیره می شود.پس سریع تر می توان به آن دسترسی پیدا کرد. همانطور که در شکل ۳-۳۹ نشان داده شده است.یک توصیف گر ۸ بایتی، شامل آدرس بخش پایه، اندازه، و سایر اطلاعات می باشد.

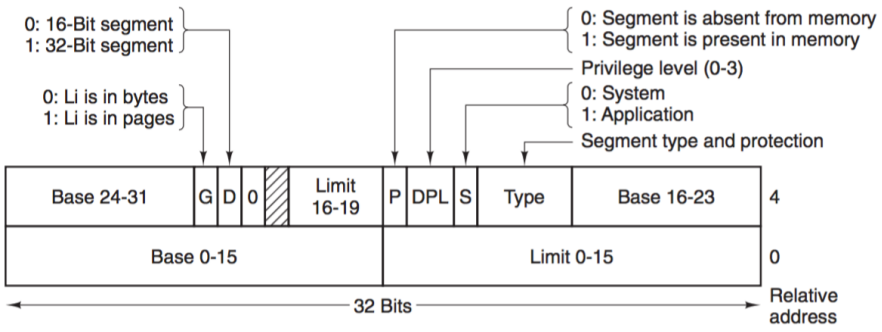


Figure 3-39. x86 code segment descriptor. Data segments differ slightly.

قالب سلکتور به شکل هوشمندانه ای انتخاب شده تامحل یابی توصیف گر به شکل آسان باشد. در ابتدا براساس بیت بیت دوم LDTیا GDTانتخاب میشود. سپس انتخاب مرحله قبل در یک ثبات داخلی ذخیره می‌شود و ۳ بیت کم ارزش مجموعه صفر می شود. در نهایت، آدرس جدول LDT یا GDT به آن اضافه میشود، تا یک اشاره گر مستقیم به توصیف گر داده شود. به عنوان مثال، سلکتور۷۲ اشاره به ورودی ۹ در GDT اشاره میکند، که در آدرس GDT + 72واقع شده است.

اکنون اجازه دهید مراحلی که توسط آن یک)سلکتور، افست (به یک آدرس فیزیکی تبدیل میشود را دنبال کنیم. به محض این که میکروپروگرمم می داند کدام ثبات سگمنت در حال استفاده است، می تواند توصیف گر مربوط به سلکتوری که در ثبات های داخلی آن است را بیابد. اگر سگمنت وجود نداشته باشد (سلکتور صفر) یک تله(trap) اتفاق می افتد.

سخت افزار از فیلد محدودیت استفاده میکند تا چک کند آیا آفست خارج سگمنت است یا داخل آن.منطقاتوصیفگری که سایز سگمنت را میدهد، باید ۳۲ بیت باشد، درحالیکه ۲۰ بیتی است. پس یک زیرکی در آن به کاربرده شده. اگر Gbit صفر باشد، فیلد محدودیت دقیقا به اندازه سایز سگمنت است تا M۱ . اگر آن یک

باشد،فیلد محدودیت سایز سگمنت را با صفحه ها میدهد به جای بایت. برای صفحاتی با سایز KB۴ ، ۲۰ بیت برای سگمنت هایی تا ۳۲^۲ بایت کافیست.

فرض کنید که سگمنت در مموری است، و آفست در محدوده مجاز می باشد، آنگاه x86 ب ۳۲ بیت پایه در توصیف گر را به آفست اضافه میکند تا آدرس خطی را تشکیل دهد. فیلد پایه به سه قسمت تقسیم می شود، و سراسر توصیف گر برای سازگاری با ۲۸۶ پخش میشود. که در آن پایه ۲۴ بیت است.

در واقع، بیت پایه به هر سگمنت اجازه می دهد تا از یک مکان دلخواه در فضای آدرس خطی ۳۲ بیتی شروع شود

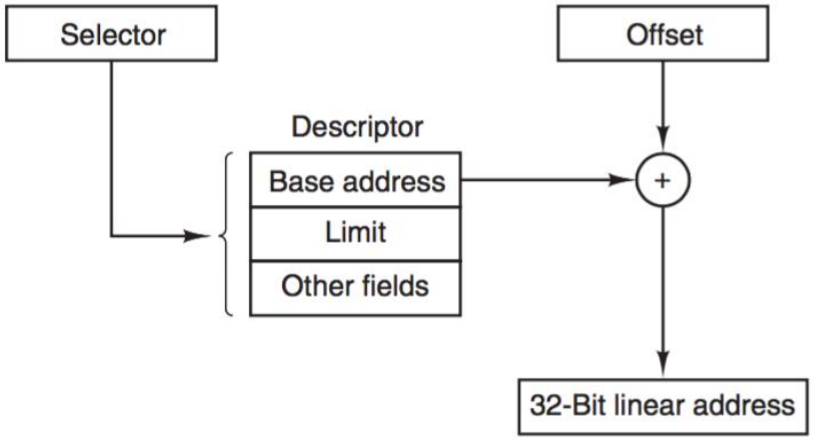


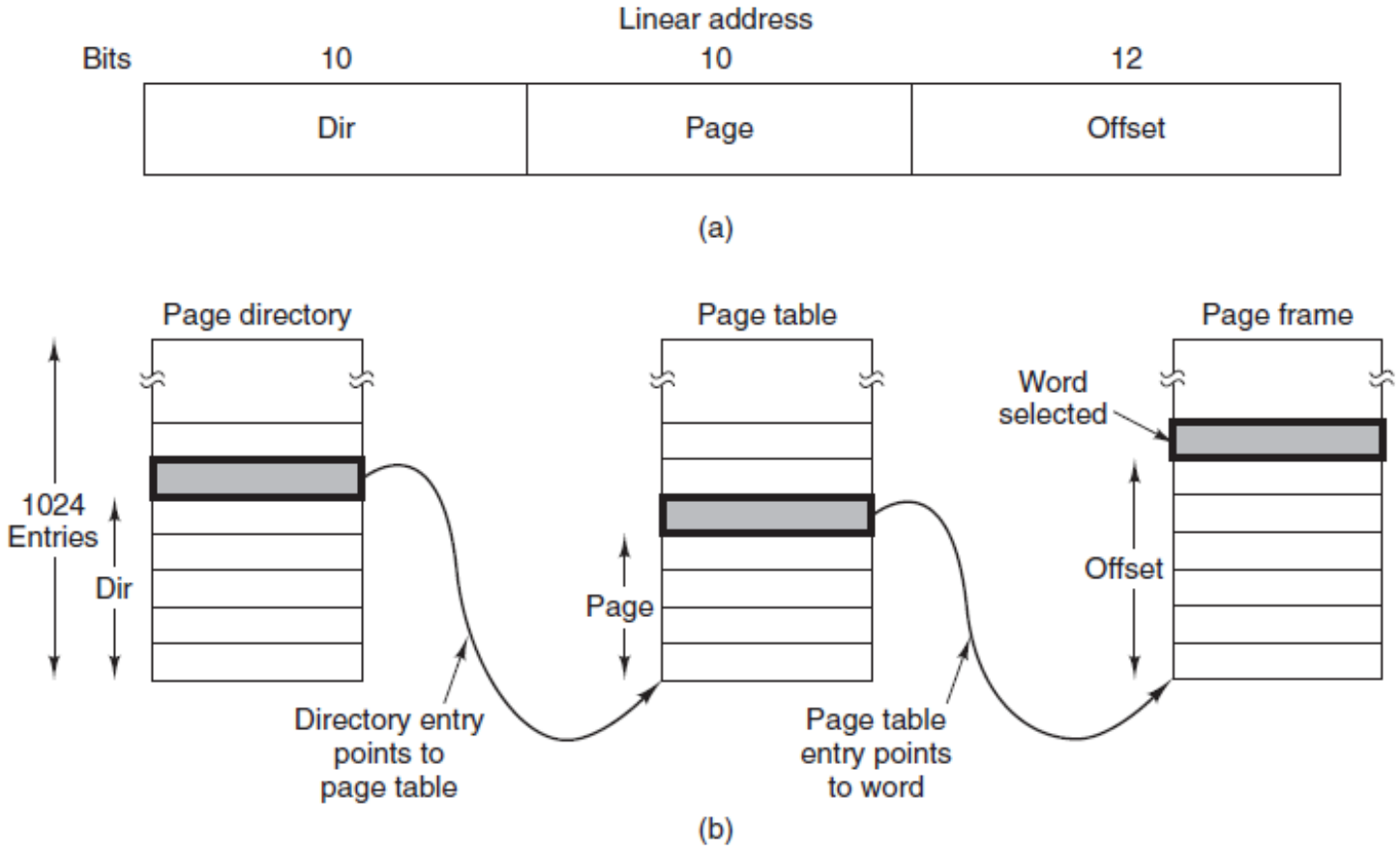
Figure 3-40. Conversion of a (selector, offset) pair to a linear address.

یکی از ویژگی های بارز سیستم صفحه بندی درخواستی این اشت که وقتی فرایندی شروع به کارمیکند تعداد زیادی از خطاهای صفحه به وحود می اید.این وضعیت ناشی از تلاش برای انتقال محل اولیه به حافظه است.همین وضعیت در زمان های دیگر ممکن است پیش بیاید.به عنوان مثال وقتی فرایندی از حافظه خارج شد دوباره شروع به کارمیکند و تمام صفحات ان برروی دیسک قرار دارد.و درهرخطای صفحه باید به حافظه اورده شوند.و پیش صفحه بندی سعی میکند ازاین صفحه بندی زیاد جلوگیری به عمل اورد.تمام صفحات مورد نیاز فرایند بطور یکجا درهمان اول به حافظه آورده میشود.پیش بندی صفحه دربعضی موارد امتیازاتی دارد.این پرسش مطرح است ایا هزینه پیش صفحه بندی کمتراز ارایه خدمات به خطای صفحات است یاخیر.ممکن است از تعداد صفحاتی که در راهبرد پیش صفحه بندی به حافظه اورده میشوند.مورد استفاده قرار نگیرند.فرض کنید s صفحه در راهبرد پیش صفحه بندی به حافظه اورده شوند. و a کسری ازاین sصفحه باشد که واقعن مورد استفاده قرار میگیرد.این سوال مطرح است که ایا هزینه صرفه جویی در a * s خطای صفحه بیشتر یا کمتر از هزینه پیش صفحه بندی s * (1-a)صفحه غیرضروری است.اگر a نزدیک به صفر باشد پیش صفحه بندی فایده ای ندارد.ولی اگر نزدیک به یک باشد مفید است. اگر صفحه بندی غیر فعال باشد (توط یک بیت در ثبات کنترل جهانی)، آدرس فیزیکی به عنوان آدرس خطی برای خواندن یا نوشتن به حافظه ارسال میشود .

بنابراین با صفحه بندی غیر فعال ، ما یک طرح قطعه بندی خالص داریم که آدرس پایه هر قطعه در توصیف کننده ی آن آمده است . قطعه ها احتمالا به دلیل اینکه زمان دردسر زیادی دارد و زمان زیادی باید صرف شود تا مشخص شود کجا از هم جدا بودند، از هم پوشانی جلو گیری نمیشوند .

از سوی دیگر ، اگر صفحه بندی فعال باشد ، آدرس خطی به عنوان آدرس مجازی است و اشاره به آدرس فیزیکی در جدول صفحه دارد. تنها عارضه واقعی این است که با یک آدرس مجازی ۳۲ بیتی و یک صفحه ۴ کیلو بایتی ، یک قطعه ممکنه شامل ۱ میلیون صفحه باشد بنابراین برای کم کردن اندازه جدول صفحه برای قطعه های کوچک از mapping سطح ۲ استفاده میشود .

هر برنامه در حال اجرا یک دایکرتوری صفحه با ۱۰۲۴ مدخل ۳۲ بیتی است . که مکانش توسط یک آدرس شاره گر به ثبات جهانی مشخص میشود . هر مدخل در این دایرکتوری به جدول صفحه که دارای ۱۰۲۴ تا مدخل ۳۲ بیتی است ، اشاره میکند . مدخل های جدول صفحه به قاب ها ی صفحه اشاره میکنند. در شکل زیر نشان داده شده اند .



Mapping of a linear address onto a physical address.

در این شکل میبینیم که یک آدرس خطی به ۳ قسمت Dirو page و offset تقسیم شده است . که Dir به عنوان یک ایندکس در دایرکتوری صفحه است که اشاره گر به جدول صفحه را مشخص میکند .

قسمت page به عنوان یک ایندکس در جدول صفحه است که آدرس فیزیکی قاب صفحه را پیدا میکند . و در نهایت offset به آدرس قاب صفحه اضافه میشود که آدرس فیزیکی بایت یا کلمه مورد نیاز پیدا شود .

هر مدخل جدول صفحه ۳۲ بیتی است که ۲۰ تا از آن ها شامل شماره قاب صفحه است . بقیه بیت های باقی مانده شامل بیت های دسترسی و کثیف که توسط سخت افزار برای نفع سیستم عامل مقدار دهی شده اند، بیت های محافظتی و بیت های دیگر اند . هر جدول صفحه حاوی "ورودی" برای ۱۰۲۴ عدد از قاب های صفحه 4_KB است، بنابراین هر جدول صفحه ۴ مگابایت از حافظه را در اختیار دارد. قطعه ی کوچکتر از ۴ مگا، یک دایرکتوری صفحه با یک ورودی منفرد اشاره گر به جدول صفحه خواهد داشت. در این صورت، سربار قطعات کوچک به جای میلیون ها صفحه مورد نیاز در جدول صفحه سطح اول، تنها دو صفحه خواهد بود.

برای جلوگیری از تولید ارجاعات تکراری به حافظه، x86، مانند MULTICS، یک TLB کوچک دارد که مستقیما Dir_Page ای که اخیرا بیشتر مورد استفاده قرار گرفته است را به آدرس فیزیکی قاب صفحه، مرتبط میکند. تنها هنگامی که داده مورد نیاز در TLB نباشد مکانیزم شکل بالا باعث بروز رسانی جدول میگردد. هرچه این فقدان ها در TLB کاهش یابد، کارایی افزایش خواهد یافت. همچنین برخی از برنامه ها نیازی به قطعه بندی ندارند، آنها بصورت ساده دارای یک فضای منفرد صفحه بندی شده ۳۲ بیتی هستند.

تمام قطعات ثبت شده میتوانند با یک انتخاب کننده یکسان که Base آن برابر صفر و Limit آن حداکثر باشد، راه اندازی شوند. در این صورت آفست، یک آدرس خطی با یک فضای آدرس دهی منفرد خواهد بود. در واقع تمام سیستم عامل های حال حاضر برای x86 به این روش عمل میکنند. OS/2 تنها مدلی است که از تمام قدرت معماری MMU بهره میگیرد.

این سوال مطرح است که چرا اینتل، چیزی که باعث تنوع فوق العاده مدل حافظه MULTICS میشد و نزدیک به ۳ دهه پشتیبانی شد را نابود کرد؟ شاید اصلی ترین دلیل این باشد که هیچ یک از سیستم عامل های UNIX و Windows هیچ گاه از آن استفاده نکردند، هرچند که آن فراخوانی سیستمی را با کمک قطعه بندی حذف کرده بود. هیچ کدام از طراحان سیستم عامل های گفته شده حاضر نبودند تا ساختار معماری خود را به مدل ارایه شده x86 تغییر دهند چرا که قابلیت انتقال به پلتفرم های دیگر از محصول آنان حذف میشد. بنابراین اینتل از صرف کردن مساحت قطعات خود برای دستیابی به چنین مدلی دست برداشت و آن را از CPU های ۶۴ بیتی حذف کرد. با این همه هنوز هم باید از طراحان x86 به خاطرجمع کردن ویژگی های متنقاضی مانند صفحه بندی، قطعه بندی، کارایی و سادگی در کنار هم، قدردانی کرد.

تمرین ۴:

فرض کنید سربار حافظه به ازای هر فرایند به دو بخش تقسیم میشود یکی میزان حافظه تلف شده به ازای اخرین صفحه فراند و دیگیر اندازه جدول صفحه فرایند اگر اندازه صفحات را کوچک بگیریم قسمت اول کاهش و قسمت دوم را افزایش مییاد .ایا میتوان یک مقدار بهینه برای اندازه صفحه تعین کرد؟

برای رسیدن به مقدار بهینه باید بین چندین عامل سبک و سنگین کنیم. برای شروع دو فاکتور وجود دارد که اندازه ی صفحه کوچک را استدلال می کند.یک محتوای داده نمیتوانند مقار صحیحی از حافظه را پر کنند. به طور میانگین نصف صفحه ی آخر خالی می ماند و به هدر می رود. که به آن internal fragmentation می گویند. اگر n قطعه در حافظه داشته باشیم و اندازه ی هر صفحه p باشد، np/2 فضای هدر رفته داریم. پس هر چه اندازه صفحه کوچکتر باشد برای این مورد بهتر است. یک مزیت دیگر برای بهتر بودن اندازه ی صفحه ی کوچک، با یک مثال آشکار می شود. یک برنامه را در نظر بگیرید که از ۴ بخش ۴ کیلو بایت تشکیل شده است. با اندازه صفحه ۱۶ کیلوبایتی، این برنامه باید ۱۶ کیلوبایت را به خود اختصاص دهد در حالی که فقط ۴ کیلو بایت نیاز دارد. حال اگر اندازه صفحه ۸ کیلو بایت بود، فضا ی کمتری به هدر می رود.

از طرف دیگر اندازه کوچک صفحه ها باعث می شود که تعداد صفحات بسیار زیاد شود و در نتیجه موجب بزرگ شدن جدول صفحه میگردد. در بعضی از ماشین ها، وقتی پردازنده از یک فرآیند به فرآیند دیگر سوییچ می کند، باید جدول صفحه در ثبات های سخت افزاری بارگذاری شوند. در این ماشین ها داشتن اندازه ی صفحه ی کوچک به این معنی است که، اینکار بیشتر طول می کشد.

اگر میانگین اندازه ی فرآیندها را s و اندازه صفحه را p و اندازه ی ورودی هر صفحه را e در نظر بگیریم، s/p صفحه احتیاج داریم، se/p هم از فضای جدول صفحه اشغال می شود. فضای به هدر رفته ی آخرین صفحه هم p/2 به طور میانگین است. پس مقدار سربار برابر است با : se/p + p/2

با افزایش اندازه ی صفحه عبارت اول کاهش و دومی افزایش می یابد. با کاهش آن هم برعکس. پس مقدار بهینه جایی در این وسط است. اگر از عبارت نسبت به p مشتق بگیریم و آن را مساوی صفر قرار دهیم، به یک معادله می رسیم که میتوان مقدار بهینه را از آن بدست آورد : $p = \sqrt{2se}$

تمرین ۵:

درباره پیاده سای حافظه بندی مجازی با تکینک قطعه بندی توضیح دهید.

برای این کار از تکنیک قطعه بندی صفحه بندی شده استفاده میشود.که در آن به ازای هر قطعه یک صفحه ساخته میشود.به عبارت دیگر برنامه نویس برنامه را قطعه بندی کرده و سیستم عامل هر قطعه را صفحه بندی میکند.در این روش هر قطعه از چند صفحهتشکیل شده در نتیجه آدرس منطقی شامل بخش های شماره قطعه ، شماره صفحه و انحراف خواهد بود.تبدیل آدرس منطقی به فیزیکی نیز بدین صورت انجام میگیرد : پس از مراجعه به جدول قطعه ، قطعه مورد نظر برنامه با توجه به ثبات STBR به کمک بخش شماره قطعه در آدرس منطقی آدرس پایه متناظر به عنوان آدرس شروع جدول صفحه از جدول قطعه استخراج میشود . سپس این آدرس پایه با مقدار شماره صفحه جمع شده تا شماره صفحه واقعی بدست آید حال به جدول صفحات مراجعه میشود تا شماره قاب صفحه مورد نظر یافت شود.

تمرین ۶:

درباره اشتراک گذاری حافظه بین فرایند ها تحقیق کنید.

یک مزیت در صفحه بندی امکان به اشتراک گذاری کد های مشترک است. این موضوع به خصوص در یک محیط به اشتراک گذاری زمانی مهم است.

یک سیستم که ۴۰ کاربر را ساپورت میکند در نظر بگیرید. که هرکدام یک ادیتور متن را اجرا می کنند. اگر ادیتور متن از۱۵۰کیلو بایت کد و ۵۰ کیلو بایت فضای داده تشکیل شده باشد ما به ۸۰۰۰ کیلو بایت برای پشتیبانی از ۴۰ کاربر نیاز داریم. حال اگر کد reentrant یا خالص باشد،می تواند به اشتراک گذاشته شود همانطور ک در شکل زیر نشان داده شده است.

اینجا سه فرایند را میبینیم که یک ادیتور سه صفحه ای را به اشتراک گذاشته اند. سایز هر صفحه ۵۰ کیلوبایت است.هر فرایند صفحه داده ی مخصوص به خود را دارد.

یک کد reentrant یک کد خود اصلاح کننده نیست. این کد هیچگاه در طول اجرا تغییر نمی کند.بنابراین یک یا چند فرایند می توانند یک کد را در همزمان اجرا کنند.

هر فرایند کپی مخصوص به خود را از ثبات ها و انباره های داده دارد. تا بتواند اطلاعات لازم برای اجرای فرایند را نگهداری کند. داده ها برای دو فرایند متفاوت حتما متفاوت خواهد بود.

تنها یک کپی از ادیتور نیاز است در حافظه ی فیزیکی نگه داری شود. هر جدول صفحه ی کاربر به کپی های فیزیکی یکسانی از ادیتور نگاشت میشود. اما صفحه های داده به قاب های متفاوتی نگاشت میشوند. در نتیجه برای پشتیبانی از ۴۰ کاربر به یک کپی از ادیتور (۱۵۰ کیلو بایت) به علاوه ی ۴۰ کپی از ۵۰ کیلو بایت فضای داده ای برای هر کاربر نیاز داریم.

مجموع فضای مورد نیاز در حال حاضر ۲۱۵۰ کیلو بایت به جای ۸۰۰۰ کیلوبایت است (یک صرفه جویی قابل توجه رخ داده است).

باقی برنامه های سنگین مانند کامپایلر ها، سیستم عامل ویندوز، پایگاه های داده و ... هم میتوانند تقسیم شوند.

برای قابل تقسیم بودن، کد باید بازگشتی reentrant باشد.

کد باید تنها قابل خواندن باشد و سیستم عامل باید این ویژگی را اجبارا ایجاد کند. سیستم عامل باید این ویژگی را داشته باشد.

به اشتراک گذاری حافظه بین فرآیند ها در یک سیستم عامل، همانند به اشترک گذاری فضای آدرسی یک برنامه بین نخ ها است. بعضی از سیستم های عامل حافظه ی به اشتراک گذاشته شده رو با استفاده از صفحات به اشتراک گذاشته شده پیاده سازی میکنند.

سازماندهی حافظه با توجه به صفحات مزایای متعددی را علاوه بر اجازه دادن به فرآیند های متعدد برای به اشتراک گذاری صفحه های فیزیکی یکسان، فراهم میکند.

تمرین ۷:

درباره تفاوت های سیستم عامل های ۳۲ بیتی و ۶۴ بیتی در مبحث مدیریت حافظه به طور دقیق تحقیق کنید.

یکی از بزرگ‌ترین مزیت‌های استفاده از سیستم‌های عامل ۶۴ بیتی، امکان دسترسی به حافظه‌های موقت (RAM)بیش از ۴ گیگابایت است و مهم‌ترین تفاوت سیستم‌های عامل ۳۲ بیتی و ۶۴ بیتی دسترسی به حافظه، مدیریت حافظه و ویژگی‌های امنیتی است.ر معماری رایانه ۳۲ بیتی، آدرس‌های حافظه یا دیگر واحدهای داده حداکثر می‌توانند ۳۲ بیت در خود داشته باشند؛ هم‌چنین ساختار پردازنده (CPU) و واحدهای محاسبه (ALU) با ۳۲ بیت نیز حداکثر ظرفیت ۳۲ بیت را دارند.یک واحد ۳۲ بیتی می‌تواند ارزشی بین ۰ تا ۴۲۹۴۹۶۷۲۹۶ را در خود ذخیره کند به همین دلیل یک پردازنده ۳۲ بیتی می‌تواند تنها به ۴ گیگابایت آدرس در حافظه (رم) به طور مستقیم دسترسی داشته باشد :در معماری رایانه ۶۴ بیتی آدرس‌های حافظه و یا دیگر واحدهای داده حداکثر می‌تواند ۶۴ بیت در خود داشته باشند و یک واحد ۶۴ بیتی می‌تواند ارزشی بین ۰ تا ۸۴۴۶۷۴۴۰۷۳۷۰۹۵۵۱۶۱۶ را در خود ذخیره کند و این یعنی یک پردازنده ۶۴ بیتی می‌تواند به هزار گیگابایت (یک ترابایت) آدرس در حافظه به طور مستقیم دسترسی داشته باشد. مزایای سیستم عامل ۶۴ بیتی اشاره کرد و خاطر نشان کرد: افزایش حجم دسترسی مستقیم به حافظه رم تا یک ترابایت (بسته به نسخه سیستم عامل)، افزایش عملکرد نرم‌افزارهایی که به توان پردازش بیش‌تری نیاز دارند (مثل نرم‌افزارهای ویرایش و ساخت تصور و فیلم، بازی‌های سنگین گرافیکی و ...) و افزایش قابلیت‌های امنیتی را می‌توان از بارزترین مزایای این سیستم عامل دانست.: با وجود مزایای قابل توجه سیستم عامل ۶۴ بیتی این سیستم دارای معایبی نیز هست که می‌توان به فقدان سخت‌افزار های موجود برای درایور ۶۴ بیتی (گاهی اوقات، امکان استفاده از این سخت‌افزارها وجود نخواهد داشت و این امکان برای قطعات قدیمی‌تر بسیار محتمل است) و تعداد کم نرم‌افزارهایی که برای سیستم‌های عامل ۶۴ بیتی اشاره کرد

تمرین ۸:

درباره پیش بینی صفحه تحقیق کنید.

یکی از ویژگی های بارز سیستم صفحه بندی درخواستی این است که وقتی فرایندی شروع به کارمیکند تعداد زیادی از خطاهای صفحه به وجود می اید.این وضعیت ناشی از تلاش برای انتقال محل اولیه به حافظه است.همین وضعیت در زمان های دیگر ممکن است پیش بیاید.به عنوان مثال وقتی فرایندی از حافظه خارج شد دوباره شروع به کارمیکند و تمام صفحات ان برروی دیسک قرار دارد.و درهرخطای صفحه باید به حافظه اورده شوند.و پیش صفحه بندی سعی میکند ازاین صفحه بندی زیاد جلوگیری به عمل اورد.تمام صفحات مورد نیاز فرایند بطور یکجا درهمان اول به حافظه آورده میشود.پیش بندی صفحه دربعضی موارد امتیازاتی دارد.این پرسش مطرح است ایا هزینه پیش صفحه بندی کمتراز ارایه خدمات به خطای صفحات است یاخیر.ممکن است از تعداد صفحاتی که در راهبرد پیش صفحه بندی به حافظه اورده میشوند.مورد استفاده قرار نگیرند.فرض کنید s صفحه در راهبرد پیش صفحه بندی به حافظه اورده شوند. و a کسری ازاین sصفحه باشد که واقعن مورد استفاده قرار میگیرد.این سوال مطرح است که ایا هزینه صرفه جویی در a * s خطای صفحه بیشتر یا کمتر از هزینه پیش صفحه بندی s * (1-a)صفحه غیرضروری است.اگر a نزدیک به صفر باشد پیش صفحه بندی فایده ای ندارد.ولی اگر نزدیک به یک باشد مفید است.