سوال۱

علت اصلی ایجاد معماری ۶۴ بیت این بوده است که تعداد دستورهای ممکن یا به اصطلاح instruction set را به نسبت معماری ۳۲ بیت افزایش بدهند. یعنی ریزپردازنده ما بتواند با معماری ۶۴ بیت بازه‌ی بیشتری از دستورات را پشتیبانی کند و دستورات پیچیده‌تری را شامل شود.

امکانات بیشتری که معماری ۶۴ بیت برای ما فراهم می‌آورد:

- فضای قابل آدرس‌دهی در معماری ۶۴ بیت واضحا بیشتر است. با ۶۴ بیت میتوان ۱۶ گیگ حافظه را آدرس‌دهی کرد اما با ۳۲ بیت تنها میتوان ۴ گیگ را آدرس دهی کرد.
- در معماری ۶۴ بیت برای الگوریتم paging حجم های 1GB، 2MB و 4KB وجود دارد. اما در معماری ۳۲ بیت مقادیر 4MB و 4KB موجود است.
- در معماری ۶۴ بیت از ۴ لایه سلسله مراتبی برای paging میتوان استفاده کرد.

سوال۲

در سیستم های موبایلی به جای استفاده از hard disk از حافظه flash استفاده میشود، و به همین علت است که امکان استفاده از swap موجود نیست. و به دو علت این اتفاق رخ میدهد. علت اول این که اندازه حافظه flash به اندازه حافظه Disk نیست و علت دوم این است که محدودیت نوشتن در حافظه‌های flash باعث میشود که استفاده از swap سبب کاهش شدید عمر این حافظه ها بشود. بنابراین در سیستم های موبایلی از روش های دیگری استفاده میشود که به شرح زیر است:

در IOS سیستم عامل هنگامی که حافظه پر شد و نیاز به جا داشتیم از فرآیند های موجود در حافظه اصلی تقاضا میشود که حجم خود را کم کنند به این صورت که اطلاعات خواندنی خود را (مثل سورس کد) از حافظه اصلی خارج کنند که در صورت نیاز از همان حافظه flash بخواند. و قسمت هایی از حافظه که تغییر میکنند، مثل استک ها، آرایه ها و... در حافظه باقی بمانند. برنامه هایی که قادر به کاهش حافظه نیستند. میبایست حافظه اصلی را ترک کنند.

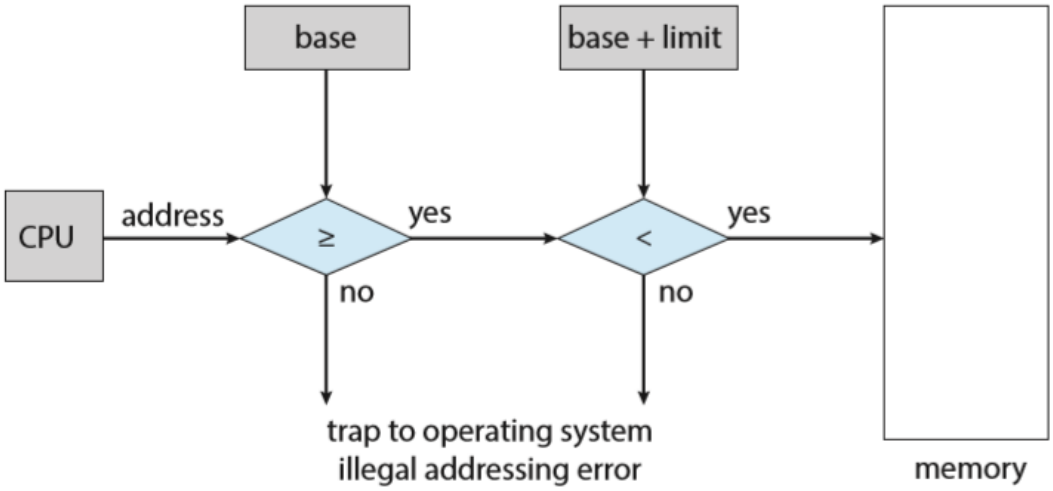
در Android نیز سیستم نسبتا مشابه است. برنامه‌هایی که فضای کافی برای اجرا ندارند را میبندد و حالت فعلی آن ها را ذخیره میکند تا مجددا بتواند آن فرآیند را ادامه دهد.(بنابراین بهتر است برنامه نویس برنامه خود را به گونه بنویسد که حجم کمی را از حافظه بخواهد.)

سوال۳

در سوال قبل بررسی شد که در موبایل قابلیت swap نداریم و زمانی که main memory پر میشود، اپلیکیشن ها داوطلابانه مموری را ترک میکنند. این ویژگی به این صورت است که سیستم عامل به فرآیند پیامی مبنی بر پایان رها کردن حافظه میدهد و در صورتی که فرآیند به این موضوع توجه نکند، فرآیند تمام(terminate) میشود.

سوال۴

ما میباست اجازه دسترسی و تغییر یک پروسس به بخش های غیر مرتبط با خودش را بگیریم. با استفاده از دو رجیستر میتوانیم این مسئله را هندل کنیم. یک رجیستر relocation register است که مکان خانه اول حافظه که این فرآیند در آن قرار گرفته است. و limit register که در آن میزان محدوده حافظه قابل استفاده برای فرآیند نگهداری میشود. وقتی در یک فرآیند که در حال اجرا است یک خانه حافظه درخواست میشود، با خواندن دو رجیستر گفته شده معتبر بودن خانه حافظه خواسته شده را بررسی میکنیم. به این صورت که ابتدا آن را با limit register چک میکند، اگر بزرگتر بود خطا رخ میدهد. در غیر این صورت به سراغ relocation register میرود و میزان آدرس داده شده را با مقدار رجیستر جمع میکند. در این لحظه آدرس واقعی خانه حافظه خواسته شده را به دست آورده‌ایم. و میتوانیم از مموری بخوانیم.



سوال۵

Dynamic loading روشی است که به واسطه آن دیگر نیازی به load کردن کل برنامه در حافظه نداریم. در این روش روتین‌ها تنها در حالتی وارد حافظه میشوند که توسط روتینی دیگر فراخوانی شوند. در ابتدا main routine را به داخل حافظه می‌آوریم و سپس شروع به اجرا میکند و چنانچه routine دیگری نیاز داشت؛ اگر قبلا در حافظه نبود آن را وارد حافظه میکنیم. این روش برای اجرای برنامه هایی که بخش های بزرگی دارند که کم استفاده میشوند مناسب است. ضمنا این امکان را برای برنامه نویس ایجاد

میکند که خودش برنامه خود را به گونه ای توسعه دهد که فضای کمتری در حافظه اصلی بگیرد(هر چند خود سیستم عامل ها هم کتابخانه هایی برای کمک به برنامه‌نویسان طراحی کرده اند.)

_____سوال۶_____

روش‌های متفاوتی برای پیاده سازی page table وجود دارد که به این صورت است که برخی برای هر فرآیند یک جدول در نظر میگیرند و برخی دیگر یک یا چند جدول برای **همه** فرآیندها در نظر میگیرند. یکی از ساده‌ترین راه‌های پیاده‌سازی سخت افزاری page table این است که از رجیستر استفاده شود برای نگهداری هر سطر از جدول. این راه سبب میشود که context switch سربار زیادی داشته باشد. این روش برای سیستم هایی مفید است که جدول زمانی تا حداکثر ۲۵۶ سطر دارند و برای حالاتی که تعداد سطرها بیشتر شود هزینه ساخت را بالا میبرد. برای page table های بزرگتر از روش PTBR استفاده میکنند. به این شکل که خود جدول ها در حافظه نگهداری میشوند. و رجیسترهایی برای برای اشاره به آن استفاده میکنیم. در این حالت نیز چون برای تغییر در یک جدول کافیست یک رجیستر را فقط تغییر دهیم، مقدار سربار برای context switch کاهش میابد.

_____سوال۷_____

الف) وقتی جدول ما در حافظه قرار میدهیم برای دسترسی به عنصر جدول میبایست مجددا به حافظه مراجعه کنیم. در این حالت دو مراجعه به حافظه داشته‌ایم. یک بار آدرس منطقی را دریافت میکنیم و بار دیگر مقدار آن خانه از حافظه را میبایستی پیدا کنیم. این موضوع سبب میشود که کارایی سیستم پایین بیاید. برای حل این مشکل از یک حافظه میانی استفاده میکنیم که سرعت آن سریع تر از حافظه اصلی است. در این حافظه میانی آدرس صفحات پرکاربرد را نگهداری میکنیم و به آن TLB میگویند. زمانی که بخواهیم به یک عنصر دسترسی پیدا کنیم ابتدا با همه خانه‌های TLB چک میکنیم و چنانچه در آن ها نبود به سراغ حافظه اصلی میرویم.

ب) زمان موثر دسترسی به حافظه :

$$1 \times 35 + 0.15 \times 40 = 41$$

_____سوال۸_____

مشکلی که در جداول صفحه ها وجود دارد این است که در این روش مقدار حافظه لازم به خاطر صفحات مجازی زیاد میشود. یک راه خوب برای رفع این مشکل استفاده از جداول معکوس صفحات است. در این روش به این ترتیب عمل میشود که از یک تابع hash استفاده میشود و برای ورودی های جدول صفحه ها به خانه های جدول معکوس صفحه ها نگاشت میشود. چنانچه دو ورودی به یک خانه نگاشت شوند به کمک ساختار لینک لیست هر دو ورودی در یک خانه قراره میگیرند (به صورت آماری

تعداد ورودی هایی که به یک خانه نگاشت میشود با توزیعی عادی کم است). ضمنا تابع hash میبایستی دسترسی نسبتا سریعی به ما بدهد.

_____سوال۹_____

یک راه برای ذخیره سازی اطلاعاتی است که با ۳۲ بیت نمیتوان آدرس داد. به این راه hashed page table میگویند. در این راه جداول hash صفحات وجود دارند که شماره صفحات مجازی را مشخص میکنند. هر عنصر در جدول دارای سه ورودی است:

- آدرس مجازی صفحه
- مقدار page frame نگاشت شده به آن
- Pointer به عنصر بعدی

برای یافتن یک آدرس از این روند استفاده میکنیم که شماره صفحه مجازی به کمک توابع hash بدست می‌آید و آن قدر جست‌وجو میکند که با فیلد اول تطابق داشته باشد و آن را برمیگرداند وگرنه به جست‌وجو ادامه میدهد.

_____سوال۱۰_____

صفحه بندی و قطعه بندی هر دو مزایای خود را دارند در یک روش برنامه نویس را نادیده میگیرد اما بیشترین استفاده را از فضای حافظه میبرد و در روش دیگر این امکان فراهم میشود که مدیریت ساختمان داده های بزرگ شدنی هندل شود ضمنا در این روش اشتراک امن هم ممکن است. در صفحه بندی هر برنامه را به صفحه های هم اندازه تقسیم میکنند که برنامه نویس در انجام آن نقشی ندارد اما در قطعه بندی میتوانیم برنامه را ماژولار کنیم. ترکیب این دو روش نیاز به حمایت سخت افزار و سیستم عامل دارد؛ به این صورت که هر برنامه به پیشنهاد برنامه نویس به قطعه هایی تبدیل میشود و سپس هر قطعه توسط سیستم عامل به یک صفحه اختصاص داده میشود و هر صفحه به یک قاب نگاشت میشود. هر آدرس مجازی شامل شماره قطعه، شماره صفحه و offset است. و به کمک این سه پارامتر قطعات مختلف حافظه قابل پیدا کردن هستند. این روش پیاده سازی شده است مثلا پردازنده سری پنتیوم اینتل از این قابلیت پشتیبانی میکرد.

_____سوال۱۱_____

الف) با توجه بزرگ شدن حافظه اصلی در سیستم های امروزی، جداول صفحات خودشان ممکن است بسیار بزرگ بشوند. ما نیازی به ذخیره‌ی جدول به صورت یکجا نداریم. به همین علت میتوان جدول صفحات را قسمت بندی کرد. برای این کار از الگوریتم چند بندی استفاده میکنیم که در جدول برای تبدیل آدرس منطقی به فیزیکی استفاده میشود.

ب)

$$32 - 10 - 8 = 16 \rightarrow 64KB$$

$$8 + 10 = 18 \rightarrow 2^{18} \rightarrow 256KB$$

2^8 → سطح نخست

2^{10} → سطح دوم

سوال۱۲

$$first\ address : 330 + 110 = 440$$

$$Second\ address : 111 + 80 = 191$$

$$Third\ address : 876 + 231 = !$$

$$Fourth\ address : 498 + 210 = 708$$

$$Fifth\ address : 111 + 110 = !$$

مورد سوم و آخر چون مقدار offset از مقدار limit بیشتر است به همین دلیل خارج از segmentation را آدرس دهی میکند و به همین دلیل بهتر ایست اجازه دسترسی داده نشود.