بسمه تعالی

پروژه پنجم آزمایشگاه سیستم عامل

دانشکده برق و کامپیوتر دانشگاه تهران

پاییز ۹۶

Virtual Memory Manager

هدف:

در این پروژه قرار است شما برنامه ای بنویسید که برای یک فضای آدرس مجازی به اندازه 2^{16} = 65,536 ، آدرس منطقی را به آدرس فیزیکی ترجمه کند. برنامه شما از یک فایل شامل آدرس‌های منطقی میخواند و به آدرس‌های فیزیکی ترجمه میکند و محتوای آن آدرس را (که بصورت بایتی ذخیره شده) بعنوان خروجی میدهد. این برنامه باید از Page Table و TLB استفاده کند. هدف این پروژه شبیه‌سازی مراحلی است که در ترجمه کردن یک آدرس منطقی به مجازی دخیل هستند. همچنین شما قرار است تا در انتها با شبیه‌سازی عملی، یک مقایسه‌ی نتایج، برای مشاهده بهینگی چند روش، ارائه کنید.

شرح پروژه:

قسمت اول:

شما در برنامه یک فایل را میخوانید که شامل تعداد زیادی عدد ۳۲ بیتی است که نشان دهنده‌ی آدرس‌های منطقی هستند. اما شما باید تنها از ۱۶ بیت آن بعنوان آدرس منطقی استفاده کنید(از Masking استفاده کنید). این ۱۶ بیت از دو بخش تشکیل شده است:

۱) ۸ بیت Page Number

۲) ۸ بیت Page Offset

در شکل زیر ساختار آدرس دهی نمایش داده شده است.

Offset (7 – 0)	Page Number (15 – 8)	(31 – 16)

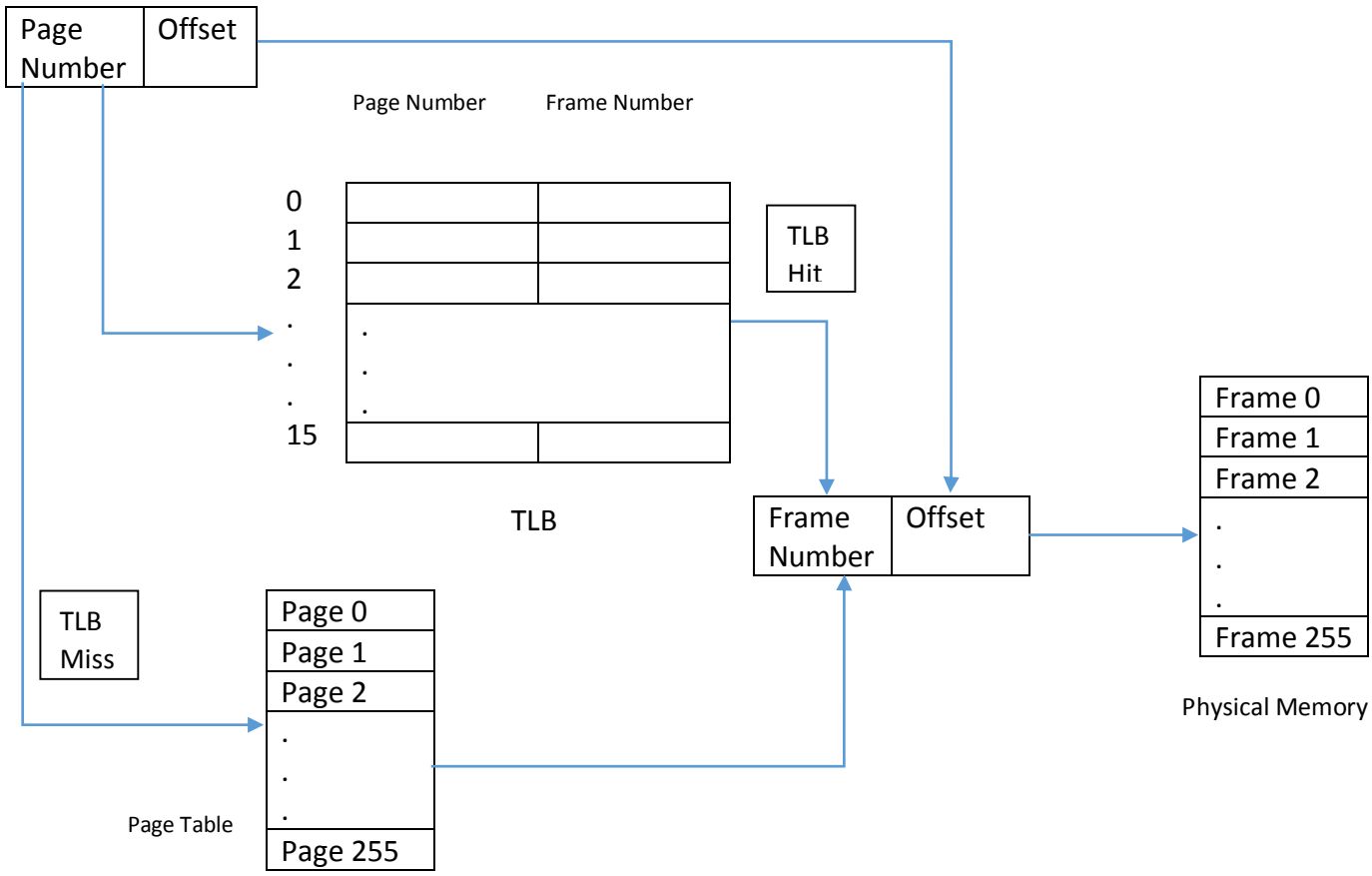
به ویژگی‌های زیر توجه کنید:

۱) Number of Entries In Page Table: 2^8
۲) Page Size: 2^8 Bytes
۳) Number of Entries in TLB: 16
۴) Frame Size: 2^8 Bytes
۵) Number of Frames: 256
۶) Physical Memory Size: 65,536 Bytes (256 Frames x 256-Byte Frame Size)

توجه کنید که شما تنها از آدرس فیزیکی میخوانید و قرار نیست در فضای آدرس منطقی چیزی بنویسید.

ترجمه آدرس:

شما باید آدرس منطقی را به آدرس فیزیکی ترجمه کنید و همچنین از TLB و Page Table استفاده کنید. ابتدا شماره صفحه از آدرس منطقی استخراج شده و از TLB استفاده میکنید. در صورتی که مراجعه به TLB موفقیت آمیز باشد که شماره صفحه از TLB استخراج میشود. در غیر این صورت از Page Table کمک میگیرید که در این صورت یا Frame Number استخراج میشود یا Page Fault رخ میدهد.



شما باید در برنامه خود همچنین Demand Paging را پیاده سازی کنید. شما باید یک فایل با نام BACKING_STORE.bin داشته باشید که اندازه آن 65,536 بایت است و هنگامی که یک Page Fault رخ میدهد شما باید از فایل BACKING_STORE.bin به اندازه ۲۵۶ بایت بخوانید و آن را در یک Page Frame در حافظه فیزیکی ذخیره کنید. برای مثال اگر یک آدرس منطقی با شماره صفحه ۱۵ دچار Page Fault شد، برنامه شما باید از BACKING_STORE.bin صفحه شماره ۱۵ را بخواند (به یاد داشته باشید که صفحات شما ۲۵۶ بایتی هستند و شماره آنها از صفر آغاز میشود.) و آن را در یک فریم حافظه فیزیکی ذخیره کند. هنگامی که این فریم ذخیره شد و Page Table و TLB بروزرسانی شدند، دسترسی‌های بعدی به این صفحه از طریق TLB یا Page Table امکانپذیر است.

توجه داشته باشید که شما به فایل BACKING_STORE.bin باید بصورت random access file دسترسی داشته باشید و بتوانید به یک مکان مشخص در فایل برای خواندن بروید. (با استفاده از کتابخانه C برای I/O)

تا اینجا سایز آدرس منطقی شما با سایز حافظه فیزیکی برابر است (۶۵۵۳۶) و نیازی نیست که نگران جایگزینی صفحات هنگام رخداد Page Fault باشید. در ادامه باید یک چاره‌ای برای شرایطی که تعیین میکنیم بیندیشید! ☺

قسمت دوم:

همانطور که میدانید ما تا کنون فرض کردیم که سایز فضای آدرس مجازی با حافظه فیزیکی برابر است، در حالی که در حقیقت سایز حافظه فیزیکی بسیار کوچکتر از فضای آدرس مجازی‌ای است که در اختیارتان قرار میگیرد. پس در این قسمت ما بدنبال حقیقت رفته و سعی میکنیم تا از فضای آدرس فیزیکی کوچکتری استفاده کنیم. در این قسمت شما باید بجای 256 Page Frame از 128 Page Frame استفاده کنید. شما باید طوری برنامه خود را تغییر دهید که اولا برنامه شما حساب Page Frame های خالی را داشته باشد و همچنین سیاست جایگزینی پیاده‌سازی کنید. شما موظف هستید که ۴ سیاست زیر را پیاده سازی کنید:

۱- FIFO
۲- LRU
۳- Second Chance
۴- Random Replacement

نکات تحویلی:

۱- هنگام تحویل پروژه یک فایل addresses.txt در اختیارتان قرار میگیرد که باید برنامه خود را بصورت زیر با فایل مذکور اجرا کنید:

./a.out addresses.txt

که این فایل شامل اعداد صحیحی بین ۰ تا ۶۵۵۳۵ است که برنامه شما فایل را باز کرده، میخواند و محتوای آن را بصورت Signed Bytes بعنوان خروجی میدهد.

۲- شما باید مشابه فایلی که در مورد ۱ بیان شد ۱۰ فایل با حداقل محتوای ۱۰۰۰۰ عدد تصادفی در هر یک از آنها که بین ۰ تا ۶۵۵۳۵ هستند راایجاد کرده، برنامه خود را با آن‌ها اجرا کنید و آمارهای زیر را در گزارش خود یادداشت کنید:

- تعداد Page Fault و درصد ارجاعاتی که منجر به Page Fault شده است.
- تعداد TLB Hit و درصد ارجاعاتی که در TLB یافت شدند و به مراحل بعد نرسیدند.
- با توجه به اعداد زیر، سربار زمانی خواندن آدرس‌های موجود در هر فایل را محاسبه کنید:
  - TLB R/W: 0.5 ns
  - Main Memory R/W: 100 ns
  - Disk R/W: 250,000 ns

توجه کنید که این آمارها باید به ازای هر فایل تست، یکبار برای قسمت اول پروژه و یکبار برای هریک از سیاست‌های بیان‌شده در قسمت دوم پروژه در گزارش ارائه شود. (قطعا برنامه شما حضورا نیز با فایل ذکر شده در مورد یک به همین ترتیب تست میشود)

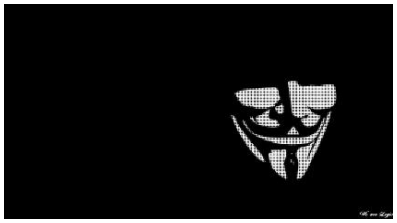
۳- پیام‌های مناسب و مفهوم جهت روشن بودن نحوه اجرای برنامه مهم است.

۴- تمامی اعضای گروه باید به تمام مفاهیم این فصل کتاب و به خصوص پروژه هنگام تحویل مسلط باشند. لزوما نمره تمام افراد در یک گروه شبیه هم نیست.

۵- هرگونه مشابهت در کد، و گزارش حرام است!

۶- شما باید در گزارش خود علاوه بر ارائه آمارهای بالا، یک مقایسه میان سیاست‌های مذکور داشته باشید.

۷- راهنمایی: با توجه به مورد ۶ نتیجه میگیریم که در ۱۰ فایل تستی که تولید میکنید تنوع را رعایت کنید. به طور مثال دو فایل از این ۱۰ فایل حتما در آدرس‌های خود ***Memory Access Locality*** داشته باشند تا بتوانیم تاثیر هریک از سیاست‌های بیان شده را درحالت‌های مختلف آدرس‌دهی مشاهده کنیم. شما باید حالتهای مختلف را تولید کنید، سیاست‌های مختلف را روی آن‌ها امتحان کرده و با اشاره به آنها در گزارش خود، سیاست‌ها را باهم مقایسه کنید.



Have Fun in Cyber Space and Physical Space...