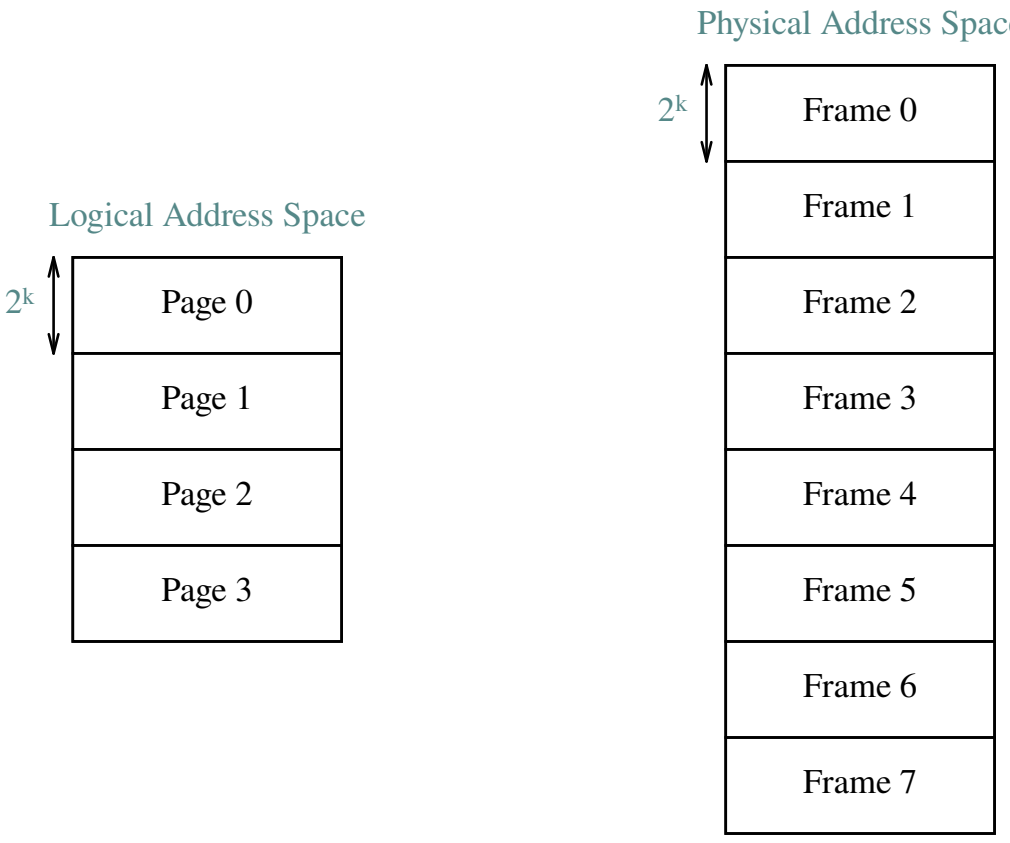
# یادداشت‌های درس سیستم‌های عامل - بخش یازدهم

از این بخش از درس به مدیریت حافظه در سیستم عامل با کمک صفحه‌بندی می‌پردازیم.

ایده‌ی کلی صفحه‌بندی:

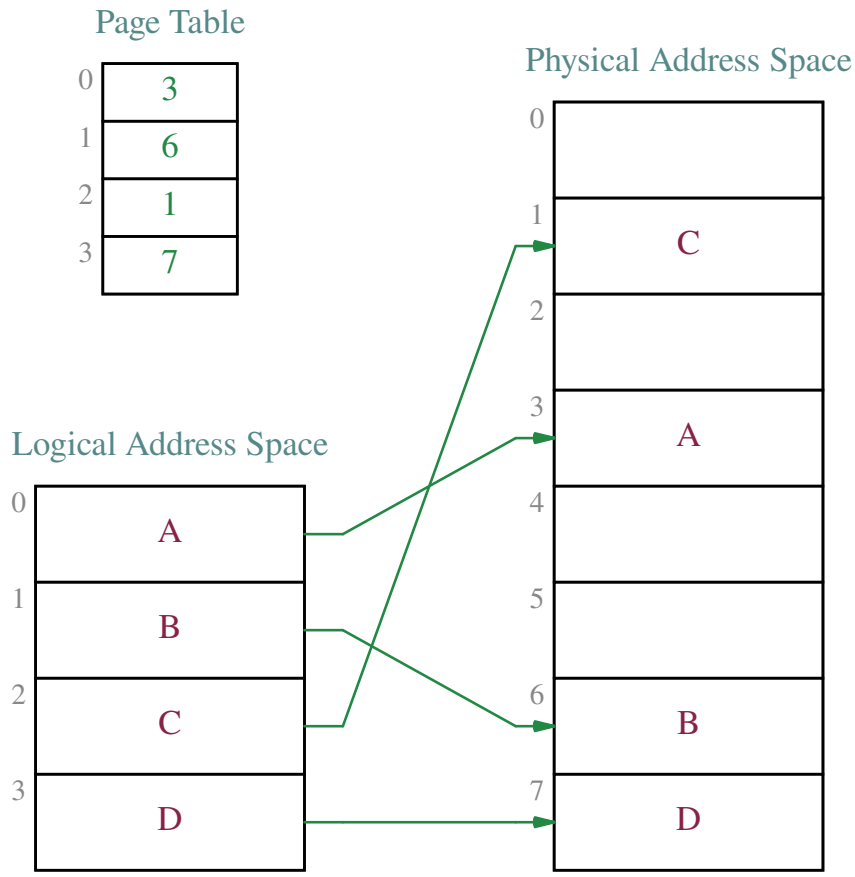
- فضای آدرس منطقی (مجموعه‌ی همه‌ی آدرس‌هایی که می‌تواند توسط پردازنده تولید شود) را به تعدادی تکه‌ی برابر به نام صفحه (Page) تقسیم می‌کنیم.
- فضای آدرس فیزیکی (مجموعه‌ی همه‌ی آدرس‌هایی که می‌تواند به حافظه‌ی اصلی فرستاده شود) را به تعدادی تکه‌ی برابر به نام قاب (Frame) تقسیم می‌کنیم.
- اندازه‌ی قاب‌ها با اندازه‌ی صفحه‌ها برابر است.




- اندازه‌ی صفحه توانی از دو است و با توجه به معماری و تنظیمات سیستم عامل معمولا از یک کیلوبایت تا چهار مگابایت است.
- دقت کنید که تعداد قاب‌ها ممکن است کمتر، مساوی یا بیشتر از تعداد صفحه‌ها باشد.

# جدول صفحه‌ها

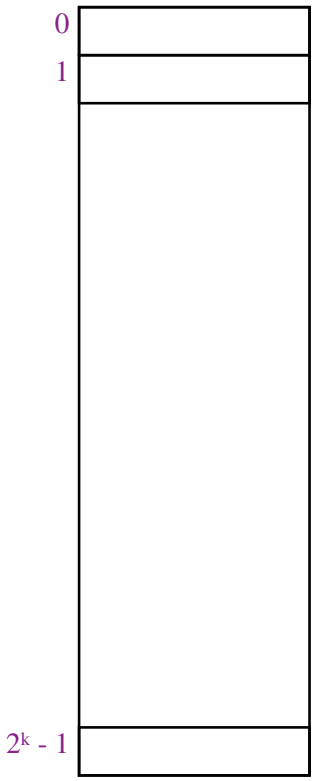
- به ازای هر پردازه، صفحه‌ها را در فضای آدرس منطقی به قاب‌ها در فضای آدرس فیزیکی نگاشت می‌کنیم.
- نگاشت صفحه‌ها به قاب‌ها را در جدولی به نام جدول صفحه نگه می‌داریم: سطر $i$-ام جدول صفحه نشان می‌دهد صفحه‌ی $i$-ام به چه قابی نگاشت شده است.
- دقت کنید که هر پردازه جدول صفحه‌ی خودش را دارد.



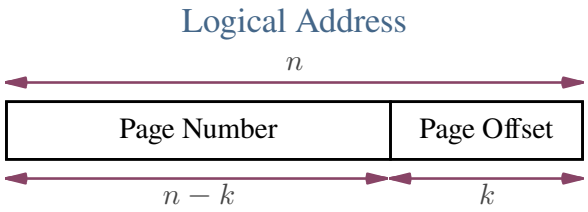

- فرض کنید در جدول صفحه، صفحه‌ی $p$-ام به قابل $f$-ام نگاشت شده باشد.
- در صفحه‌بندی اگر پردازه‌ای بخواهد به بایت $x$-ام از صفحه‌ی $p$-ام دسترسی داشته باشد، باید به بایت $x$-ام در قاب $f$-ام در حافظه‌ی اصلی دسترسی داشته باشد.

فضای آدرسی با اندازه‌ی $۲^k$ بایت در نظر بگیرید.

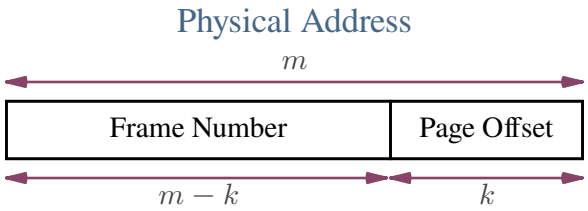
- برای اینکه به هر بایت از این فضا دسترسی داشته باشیم، بایت‌ها را از صفر تا $۲^k - ۱$ شماره‌گذاری می‌کنیم.
- برای اینکه بتوان به هر بایت از این فضای آدرس دسترسی داشت، به یک عدد $k$ بیتی احتیاج داریم.
- بنابراین برای آدرس دهی یک فضای آدرس با اندازه‌ی $۲^k$ به آدرس‌هایی با $k$ بیت احتیاج داریم.



- فرض کنید فضای آدرس منطقی $۲^n$ بایت باشد؛ در این صورت طول آدرس منطقی $n$ بیت است.
- همچنین اگر فضای آدرس فیزیکی $۲^m$ بایت باشد؛ در این صورت طول آدرس فیزیکی $m$ بیت است.
- فرض کنید اندازه صفحه $۲^k$ بایت باشد؛ در این صورت $k$ بیت لازم است تا مشخص شود به چه بایتی از یک صفحه دسترسی انجام می‌شود.
- در صفحه‌بندی، $n - k$ بیت با ارزش آدرس منطقی شماره‌ی صفحه (Page Number) و $k$ بیت کم‌ارزش آن شماره‌ی بایت صفحه (Page Offset) را نمایش می‌دهد.

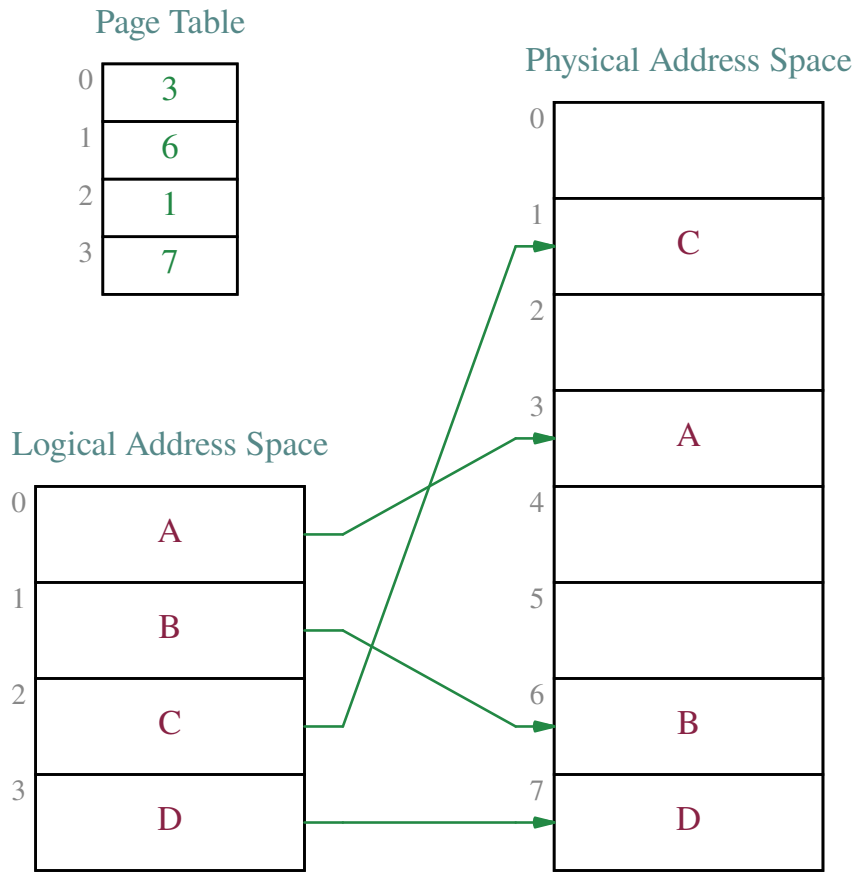



- در صفحه‌بندی با مراجعه به جدول صفحه مشخص می‌شود صفحه‌ی مورد نظر به چه قابی نگاشت شده است. در تولید آدرس فیزیکی، قسمت شماره‌ی صفحه را با شماره‌ی قاب جایگزین می‌کنیم و شماره‌ی بایت صفحه تغییر نمی‌کند.

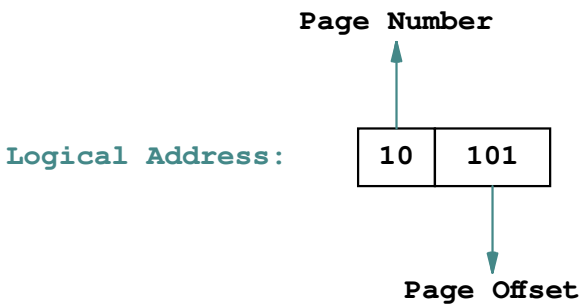


## مثال صفحه‌بندی

برای مثال فرض کنید اندازه‌ی فضای آدرس منطقی ۳۲ بایت و فضای آدرس فیزیکی ۶۴ بایت باشد؛ شکل زیر را در نظر بگیرید.

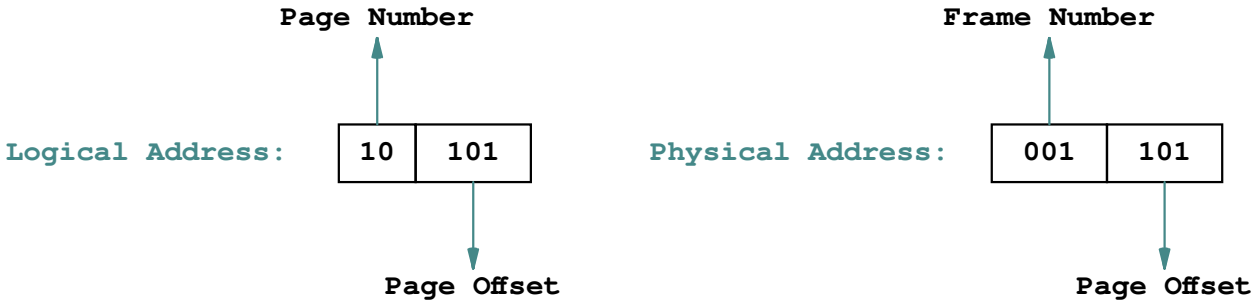



- چون چهار صفحه موجود هستند، اندازه‌ی هر صفحه هشت بایت است.
- آدرس 15 در مبنای شانزده را در نظر بگیرید؛ نمایش مبنای دوی این آدرس 10101 هست. چون اندازه‌ی صفحه هشت بایت است، سه بیت کم ارزش این آدرس مقدار Page Offset را نشان می‌دهد و دو بیت پر ارزش آن شماره‌ی صفحه را نشان می‌دهد. بنابراین، این آدرس به بایت پنجم از صفحه‌ی دوم اشاره می‌کند.



# مثال صفحه‌بندی

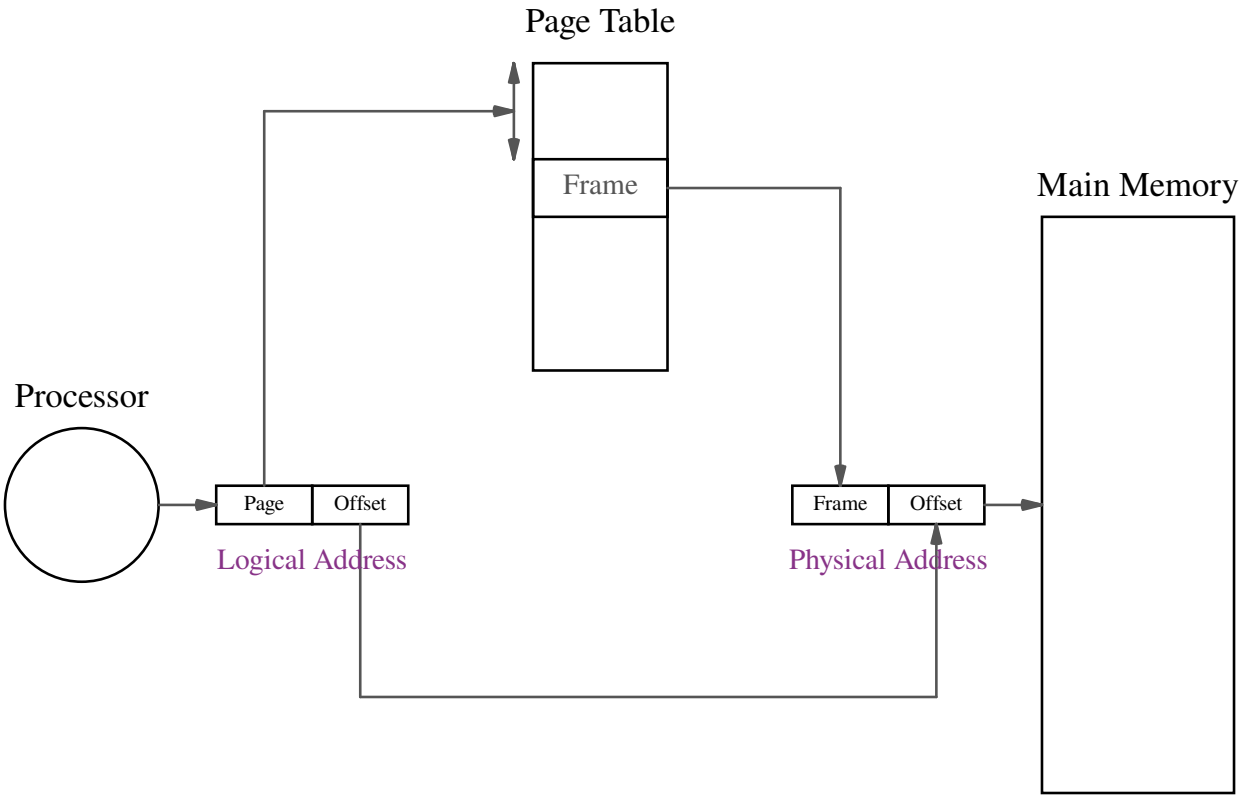
- برای بدست آوردن آدرس فیزیکی این آدرس، ابتدا با مراجعه به جدول صفحه می‌بینیم که صفحه‌ی دوم به قاب یکم نگاشت شده است. آدرس فیزیکی را با قرار دادن شماره‌ی قاب و همان Page Offset آدرس منطقی بدست می‌آوریم.




- بنابراین آدرس فیزیکی 001101 در نمایش دودویی و 0D در نمایش مبنای شانزده هست و این آدرس به بایت پنجم از قاب یکم اشاره می‌کند.
- جدول زیر چند مثال دیگر را نشان می‌دهد. آدرس‌ها در مبنای شانزده هستند و عدد داخل پرانتز آدرس را در مبنای دو نشان می‌دهد.

| Logical Address | Page Number | Page Offset | Frame Number | Physical Address |
|---|---|---|---|---|
| 10 (10000) | 2 | 0 | 1 | 08 (001000) |
| 09 (01001) | 1 | 1 | 6 | 31 (110001) |
| 1B (11011) | 3 | 3 | 7 | 3B (111011) |

شکل زیر عملکرد MMU را با صفحه‌بندی نشان می‌دهد.



وقتی از Paging استفاده می‌شود برای دسترسی به هر آدرس منطقی دو خانه از حافظه‌ی فیزیکی خوانده می‌شود.

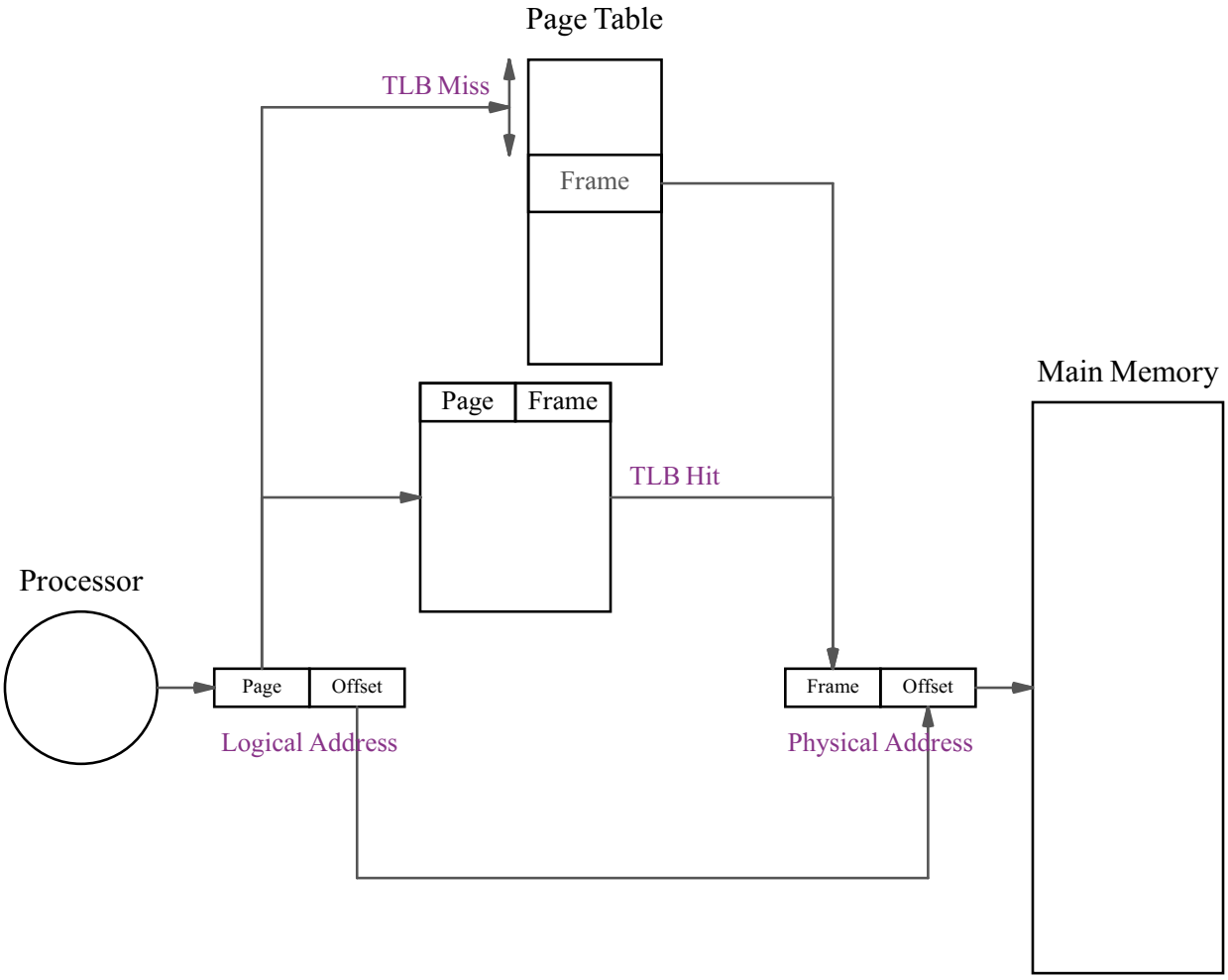
الف) یک بار برای خواندن جدول صفحه برای بدست آوردن شماره‌ی قاب

ب) یک بار برای دسترسی به آدرس فیزیکی

- بنابراین، سرعت دسترسی به حافظه حدود نصف می‌شود (چون به ازای هر دسترسی به یک آدرس منطقی، دو دسترسی به حافظه انجام می‌شود).
- برای بهبود سرعت صفحه‌بندی، قطعه‌ی سخت‌افزاری به نام Translation Lookaside Buffer یا به صورت مخفف TLB به واحد مدیریت حافظه اضافه شده است.
- قطعه‌ی TLB چند نگاشت از صفحه‌ها به قاب‌ها که اخیرا به آنها دسترسی انجام شده است را نگه می‌دارد؛ اگر شماره‌ی صفحه‌ای در TLB موجود باشد برای یافتن قابی که صفحه به آن نگاشت شده است لازم نیست به جدول صفحه دسترسی انجام شود و شماره‌ی قاب از TLB خوانده می‌شود.
- اندازه‌ی TLB کوچک است (سطرهای محدودی از جدول صفحه در آن جا می‌گیرد) و ساختار و جستجو در آن شباهت زیادی به حافظه‌ی نهان پردازنده دارد.

## شمای صفحه‌بندی با وجود TLB

- در صورتی که TLB موجود باشد، برای نگاشت آدرس منطقی به آدرس فیزیکی ابتدا TLB برای وجود شماره‌ی صفحه جستجو می‌شود. اگر شماره‌ی صفحه در TLB موجود باشد TLB Hit (برخورد) رخ می‌دهد و اگر شماره‌ی صفحه در TLB موجود نباشد TLB Miss رخ می‌دهد.
- سرعت جستجو در TLB بسیار زیاد است.



# زمان مؤثر دسترسی به حافظه

- زمان مؤثر دسترسی به حافظه (Effective Memory Access Time) متوسط زمانی است که برای دسترسی به یک آدرس منطقی لازم است.
- فرض کنید نرخ برخورد (احتمال رخداد برخورد TLB) برابر $p$ باشد و زمان هر بار دسترسی به حافظه‌ی اصلی $t$ نانوثانیه و زمان جستجوی یک شماره‌ی صفحه در TLB ناچیز باشد.
- در این صورت زمان مؤثر دسترسی به حافظه از رابطه‌ی زیر بدست می‌آید (امید ریاضی زمان دسترسی به یک آدرس منطقی):

$$۲t(۱-p)+tp$$

- توضیح: با احتمال $p$ برخورد رخ می‌دهد و در صورت برخورد یک بار دسترسی به حافظه انجام می‌شود و با احترام $(۱-p)$ Miss رخ می‌دهد و در این حالت دو بار دسترسی به حافظه انجام می‌شود.
- برای مثال، فرض کنید نرخ برخورد برابر نود درصد باشد و زمان هر بار دسترسی به حافظه‌ی اصلی ۱۰۰ نانوثانیه و زمان جستجوی یک شماره‌ی صفحه در TLB ناچیز باشد. در این صورت زمان مؤثر دسترسی به حافظه برابر است با:

$$۲۰۰ns\frac{۱}{۱۰}+۱۰۰ns\frac{۹}{۱۰}=۱۱۰ns$$

- در واقعیت معمولا نرخ برخورد بالاتر از نود و نه درصد است.

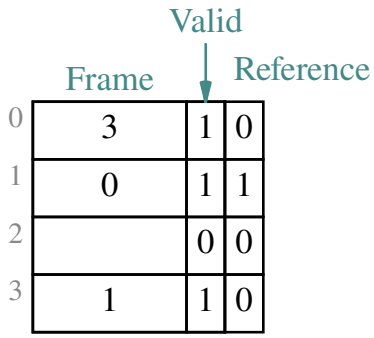
در کنار هر قاب در جدول صفحه، تعدادی پرچم (Flag، یعنی یک مقدار Boolean که معمولا توسط یک بیت ذخیره می‌شود) قرار دارد. برخی از این پرچم‌ها در ادامه معرفی می‌شوند.

- در یک جدول صفحه ممکن است برخی از صفحه‌ها به قابی نگاشت نشده باشند.
- دسترسی به این صفحه‌ها موجب خطای صفحه (Page Fault) می‌شود که یک Trap هست.
- هر سطر از جدول صفحه یک پرچم اعتبار یا عدم اعتبـار (Valid/Invalid) دارد که مشـخص می‌کند آن سطر از جدول معتبر هست یا خیر.
- اگر صفحه‌ای به قابی نگاشت نشده باشد، بیت اعتبار آن صفر است.
- برای نمونه در شکل زیر، بیت اعتبار صفحه‌ی شماره‌ی دو صفر است. یعنی صفحه‌ی شماره‌ی دو به قابی نگاشت نشده است و در نتیجه اگر پردازه به این صفـحه دسترسی داشتـه باشد، ایـن دسترسی غیـر مجاز اسـت و خطای صفحه رخ می‌دهد.

| | Frame | Valid |
|---|---|---|
| 0 | 3 | 1 |
| 1 | 0 | 1 |
| 2 | | 0 |
| 3 | 1 | 1 |

- پس از رخداد خطای صفحه، سیستم عامل تصمیم می‌گیرد با پردازه چه کـاری انجام دهـد (برای نمونـه پردازه را خاتمه و خطا را به کاربر گزارش دهد).

- در برخی از معماری‌ها، به ازای هر سطر از جدول صفحه، بیتی به نام Reference وجود دارد.
- اگر به صفحه‌ای دسترسی انجام شود، مقدار بیت Reference مربوط به آن صفحه در جدول صفحه به یک تغییر می‌کند.
- برای نمونه در شکل زیر، از بین سه صفحه‌ای که در جدول صفحه معتبر هستند، بیت Reference صفحه‌ی شماره‌ی یک مقدار یک دارد. یعنی، یکی از دسترسی‌های اخیر به کلمه‌ای در این صفحه دسترسی داشته است.




- دقت کنید که مقدار بیت Reference توسط پردازنده یا واحد مدیریت حافظه در هنگام دسترسی به یک آدرس منطقی از صفر به یک تغییر می‌کند.

- در برخی از معماری‌ها، به ازای هر سطر از جدول صفحه، بیتی به نام Dirty وجود دارد.
- اگر به بایتی از صفحه‌ای نوشته شود، مقدار بیت Dirty مربوط به آن صفحه در جدول صفحه به یک تغییر می‌کند.
- برای نمونه در شکل زیر، از بین سه صفحه‌ای که در جدول صفحه معتبر هستند، بیت Dirty صفحه‌ی شماره‌ی سه مقدار یک دارد. یعنی، یکی از دسترسی‌های اخیر به کلمه‌ای در این صفحه نوشته شده است.

| | Frame | Valid | Dirty |
|---|---|---|---|
| 0 | 3 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | | 0 | 0 |
| 3 | 1 | 1 | 1 |

- بیت Dirty یک صفحه توسط پردازنده یا واحد مدیریت حافظه در هنگام نوشتن به یک آدرس منطقی در آن صفحه از صفر به یک تغییر می‌کند.

## پرچم فقط خواندنی

- در برخی از معماری‌ها، به ازای هر سطر از جدول صفحه، بیتی به نام Read-only وجود دارد.
- اگر برای صفحه‌ای بیت Read-only مقدار یک را داشته باشد یعنی خواندن بایت‌های این صفحه مجاز است ولی نوشتن به آنها مجاز نیست.
- اگر پردازه به آدرسی بنویسد که مربوط به صفحه با بیت Read-only باشد، یک خطای صفحه رخ می‌دهد (یک Trap) و سیستم عامل تصمیم می‌گیرد با پردازه چه کند.

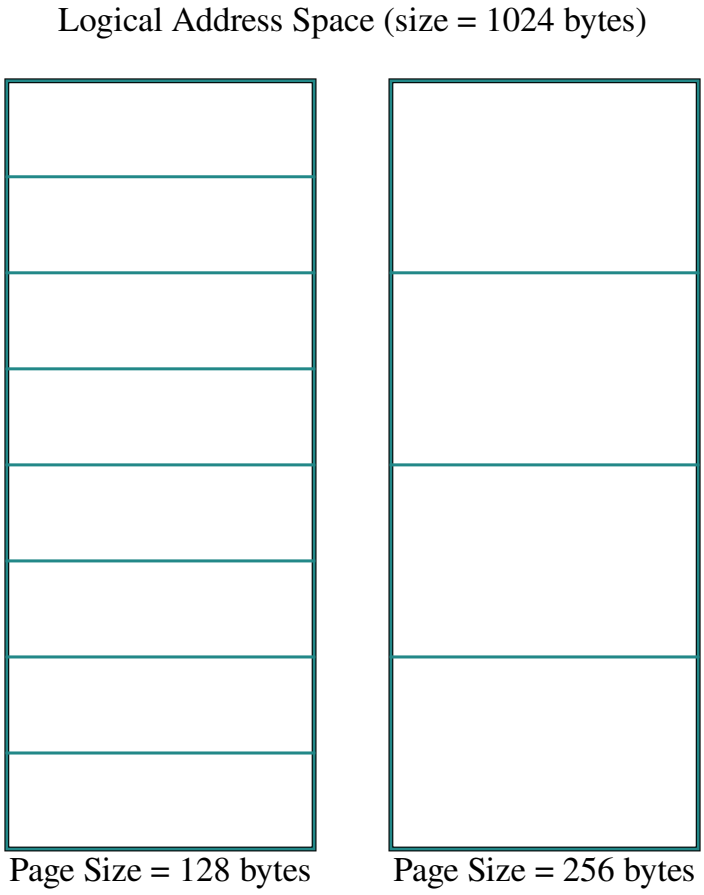
یک پردازنده‌ی سی و دو بیتی را در نظر بگیرید (آدرس‌های منطقی سی و دو بیتی هستند) و فرض کنید اندازه‌ی صفحه ۴۰۹۶ بایت باشد.

- در این صورت تعداد صفحه‌ها (و در نتیجه تعداد سطرهای جدول صفحه) $۲^{۲۰}$ عدد است.
- اگر هر سطر از جدول صفحه چهار بایت باشد، حجم جدول صفحه چهار مگابایت خواهد بود.
- اگر در سیستم عامل صد پردازه در حال اجرا باشند، در مجموع چهارصد مگابایت از حافظه فقط برای نگهداری جدول‌های صفحه‌ی پردازه‌ها استفاده می‌شود (هر پردازه جدول مخصوص خودش را دارد).
- در معماری‌های شصت و چهار بیتی اندازه‌ی جدول صفحه بسیار بزرگ‌تر است (با فرض اینکه طول آدرس منطقی شصت و چهار بیت و اندازه‌ی صفحه چهار مگابایت باشد محاسبه کنید برای نگهداری جدول صفحه‌ی هر پردازه چقدر حافظه لازم است).
- برای کاهش حافظه‌ی مورد نیاز برای نگهداری جدول صفحه‌ی پردازه‌ها از روش‌های مختلفی استفاده می‌شود.

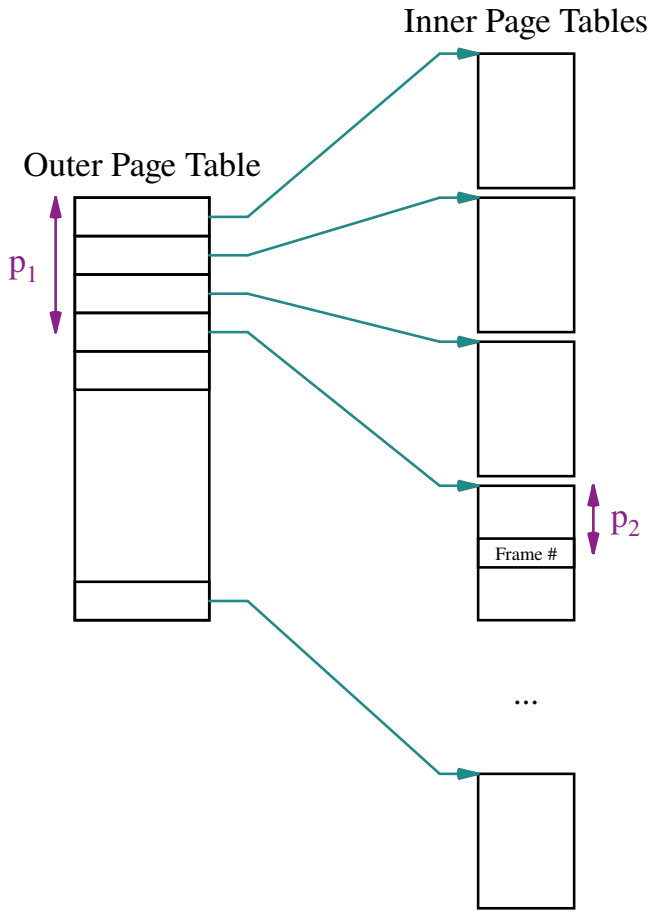
## تأثیر اندازه‌ی صفحه

هر چه اندازه‌ی صفحه بزرگ‌تر شود، تعداد صفحه‌ها کمتر می‌شود و در نتیجه حجم جدول صفحه کاهش می‌یابد. اما چند پارگی داخلی افزایش می‌یابد (چرا؟).

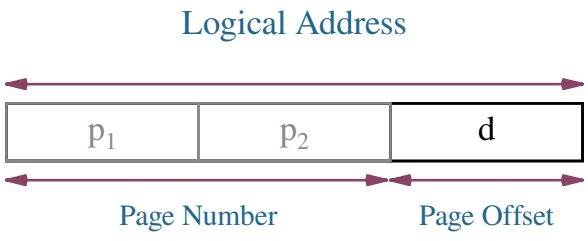
- اگر اندازه‌ی صفحه کوچک‌تر باشد چند پارگی داخلی کاهش می‌یابد اما حجم جدول صفحه زیاد می‌شود.



- در جدول صفحه‌ی دو رده‌ای، جدول صفحه به تعدادی تکه‌ی کوچک‌تر تقسیم می‌شود.
- سپس آدرس این تکه‌های جدول صفحه در جدول دیگری به نام جدول صفحه‌ی خارجی (Outer Page Table) یا Page Directory نگهداری می‌شود.




- تکه‌هایی از جدول صفحه که هیچ شماره‌ی قابی را نگه نمی‌دارند، در حافظه قرار نمی‌گیرند و سطر مربوط به آنها در جدول صفحه‌ی خارجی غیر معتبر علامت‌گذاری می‌شود.
- قسمت شماره‌ی صفحه در آدرس منطقی به دو قسمت تقسیم می‌شود (شکل زیر). قسمت اول ($p_1$) شماره‌ی تکه‌ی جدول صفحه در جدول صفحه‌ی خارجی و قسمت دوم ($p_1$) شماره‌ی سطر در تکه‌ی جدول صفحه را نشان می‌دهد.



## جدول صفحه‌ی چند رده‌ای

- بسیاری از معماری‌های امروزی از جمله X86 و ARM از جدول صفحه‌ی چند رده‌ای برای نگهداری جدول‌های صفحه‌ی بزرگ استفاده می‌کنند.
- به صورت مشابه می‌توان جدول‌های صفحه‌ای با رده‌های بیشتر نیز تعریف کرد؛ برای مثال، در معماری X86-64 جدول صفحه چهار رده‌ای است.

## جدول صفحه‌ی درهم‌سازی

در جدول صفحه‌ی درهم‌سازی (Hashed Page Table)، از یک جدول درهم‌سازی (Hash Table) برای جدول صفحه استفاده می‌شود.

- در درس ساختمان داده با جدول‌های درهم‌سازی آشنا شده‌اید.
- در این جدول، شماره‌ی صفحه برای محاسبه‌ی مقدار تابع درهم‌سازی استفاده می‌شود.
- برای نگهداری عناصری از جدول صفحه که برای آنها تصادم (Conflict) رخ می‌دهد، معمولا روشی مثل لیست پیوندی برای نگهداری عناصر استفاده می‌شد.

## جدول صفحه‌ی معکوس

اگر چه تعداد سطرهای جدول صفحه (در نتیجه تعداد صفحه‌های فضای آدرس منطقی) بسیار بزرگ است، تعداد قاب‌های حافظه‌ی فیزیکی محدود است.

- در جدول صفحه به ازای هر پردازه یک جدول صفحه نگهداری می‌شود که مشخص می‌کند هر صفحه فضای آدرس منطقی پردازه به چه قابی در حافظه‌ی اصلی نگاشت شده است.
- در جدول صفحه‌ی معکوس (Inverted Page Tables) به ازای هر پردازه جدول صفحه‌ای نگهداری نمی‌شود. در عوض، برای هر قاب حافظه‌ی اصلی نگهداری می‌شود چه صفحه‌ای از چه پردازه‌ای به آن نگاشت شده است.