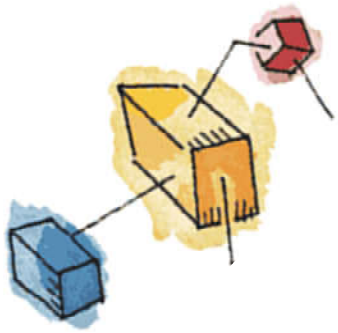
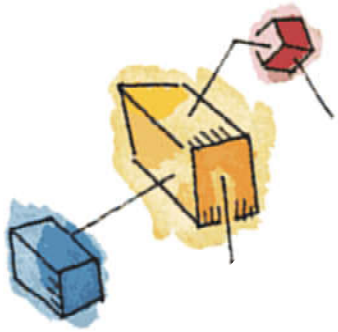
به نام خدا

فصل هشتم:
حافظه مجازی (بخش دوم)

Virtual Memory



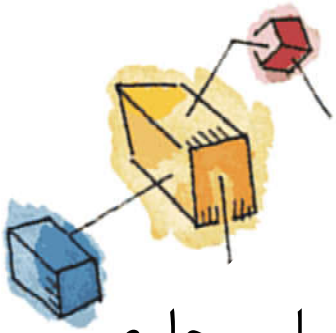
- اندازه صفحه



اندازه صفحه

- اندازه صفحه کوچک تر ← مقدار تکه تکه شدن داخلی کمتر
- اندازه صفحه کوچک تر ← تعداد صفحات مورد نیاز برای هر فرآیند بیشتر
- تعداد صفحات بیشتر برای هر فرآیند، به معنی جداول صفحه بزرگتر خواهد بود.
- هرچه جداول صفحه بزرگتر باشند، بخشی از جداول صفحه که در حافظه مجازی قرار دارد نیز بزرگتر خواهد بود.

- از طرف دیگر، حافظه جانبی به شکلی طراحی شده است که بتواند بلاک های بزرگ داده را به شکل موثری انتقال دهد، بنابراین از این نظر، اندازه صفحه بزرگتر بهتر خواهد بود.

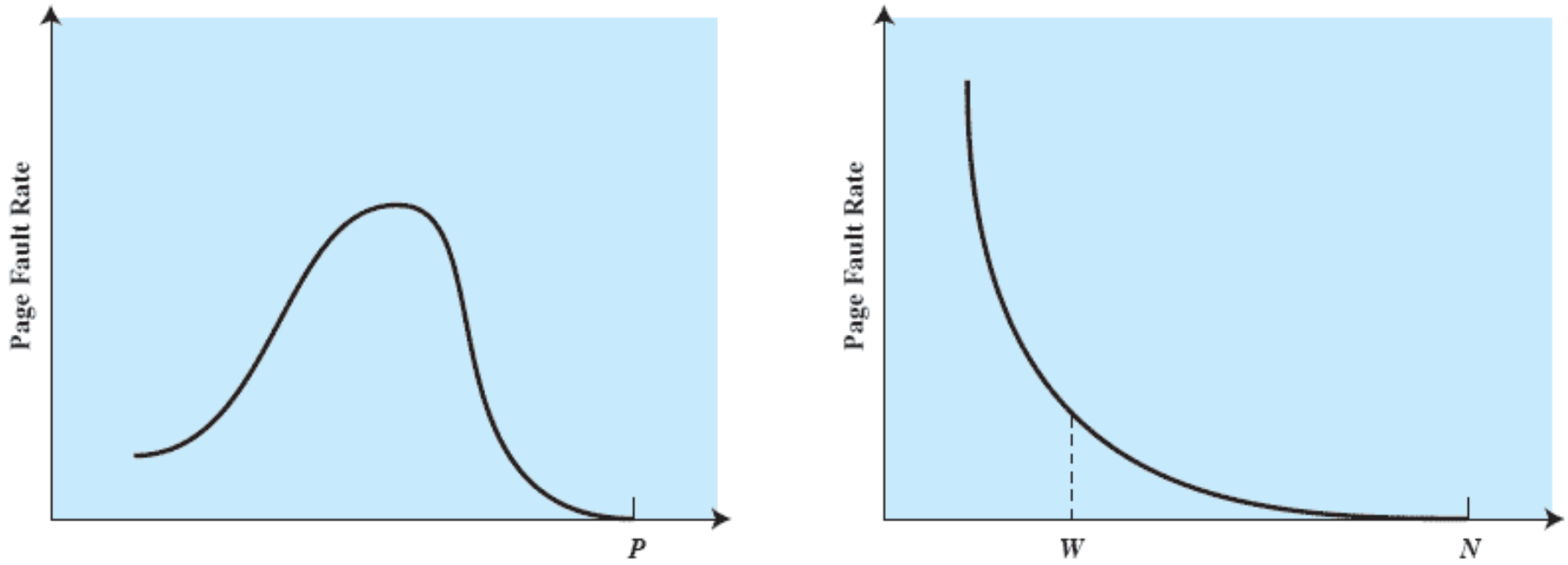


اندازه صفحه

- با گذشت زمان در هنگام اجرا، صفحات موجود در حافظه اصلی حاوی بخش های (آدرس های) نزدیک به ارجاعات اخیر فرآیند خواهند بود.
  - خطای صفحه کم می شود.

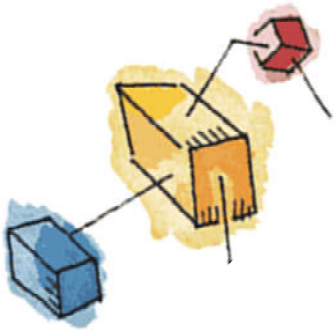
- افزایش اندازه صفحه موجب می شود که صفحات حاوی داده هایی شوند که فاصله زیادی با ارجاعات اخیر دارند.
  - خطای صفحه زیاد می شود.

رفتار معمول در صفحه بندی یک برنامه



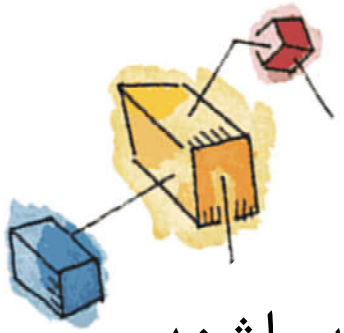
P = size of entire process
W = working set size
N = total number of pages in process

Figure 8.11 Typical Paging Behavior of a Program



سرفصل مطالب

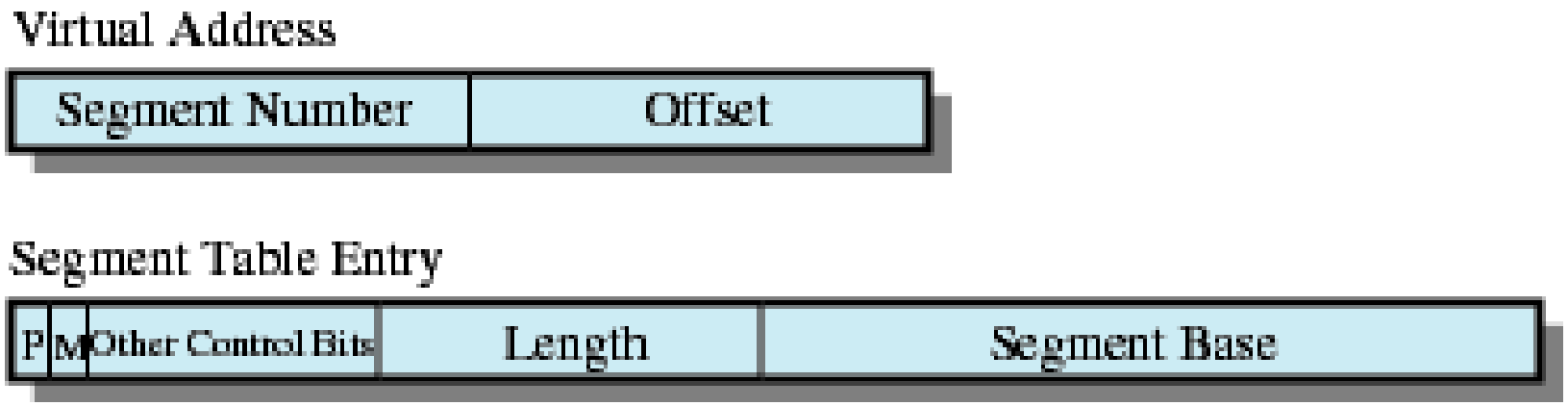
1. معرفی حافظه مجازی
2. صفحه بندی و صفحه بندی با حافظه مجازی
3. قطعه بندی و قطعه بندی با حافظه مجازی
4. سیاست های نرم افزاری برای حافظه مجازی



قطعه بندی (Segmentation)

- قطعه ها می توانند اندازه های نامساوی و پویایی داشته باشند.
- در این صورت می توان رشد ساختمان داده ها را ساده تر مدیریت کرد.
- اجازه می دهد که برنامه ها به صورت مستقل، تغییر داده شوند و مجددا کمپایل شوند.
- به اشتراک گذاری داده ها بین فرآیندها را راحت تر می توان انجام داد.
- حفاظت از داده ها را بهتر می توان انجام داد.

یک مدخل جدول قطعه



(b) Segmentation only

ترجمه آدرس در سیستم قطعه بندی

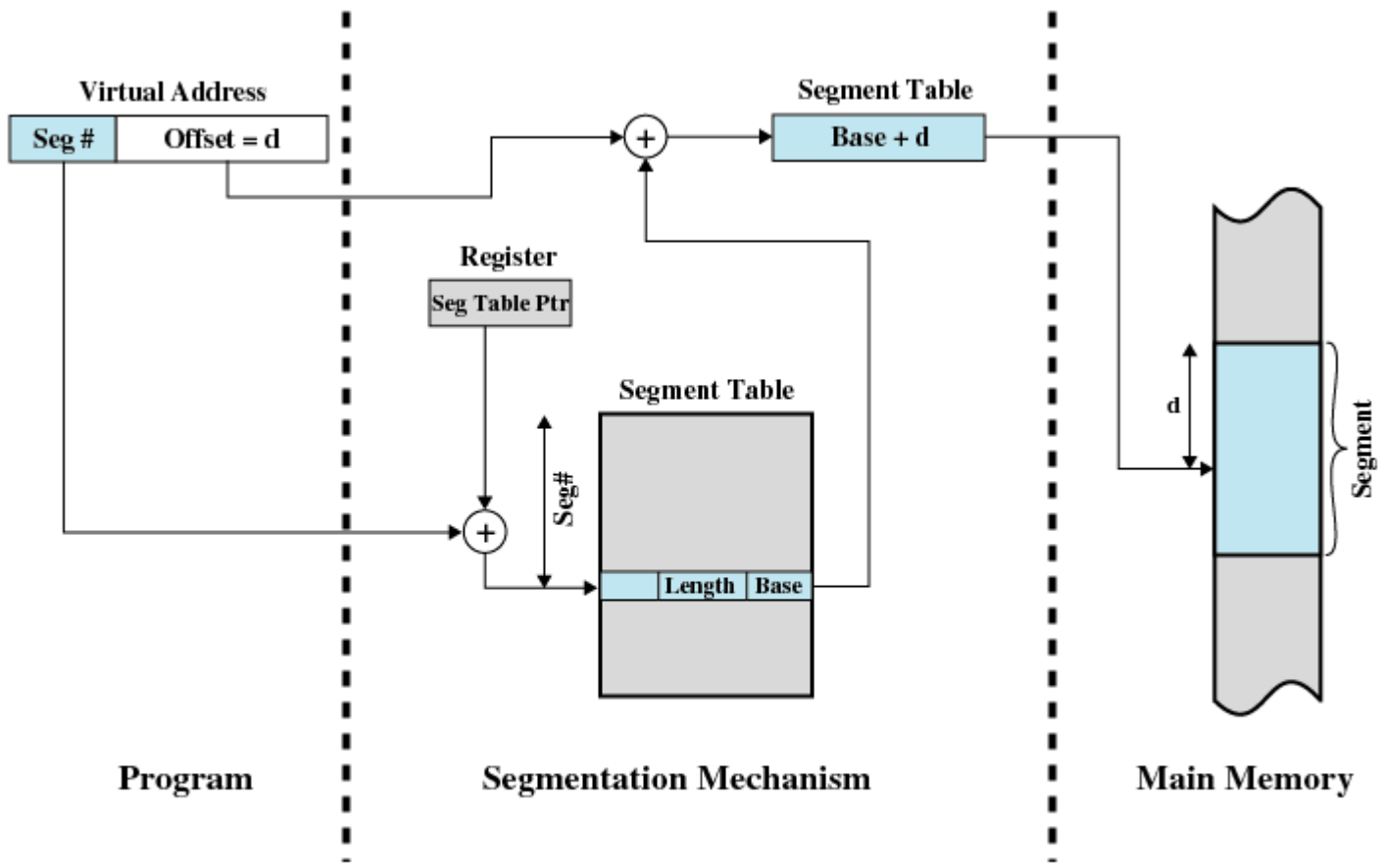
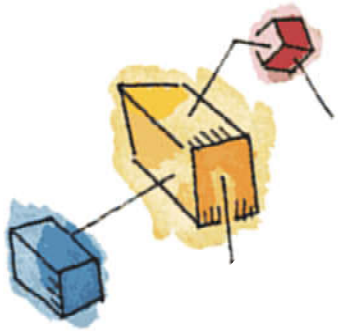


Figure 8.12 Address Translation in a Segmentation System

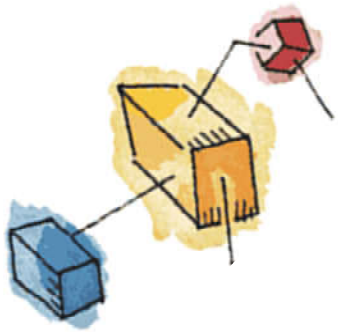


جدول قطعه (Segment Tables)

- جدول قطعه محل قطعه ها در حافظه اصلی را نشان می دهد.
- هر مدخل جدول قطعه شامل آدرس شروع قطعه در حافظه اصلی و همچنین طول قطعه است.

- در هر مدخل، به بیتی نیاز است که مشخص کند آیا آن قطعه هم اکنون در حافظه اصلی حضور دارد یا خیر.

- بیت دیگری که نیاز است، بیت تغییر است که نشان می دهد آیا این قطعه از وقتی که به حافظه اصلی لود شده است، تغییر یافته است یا خیر.

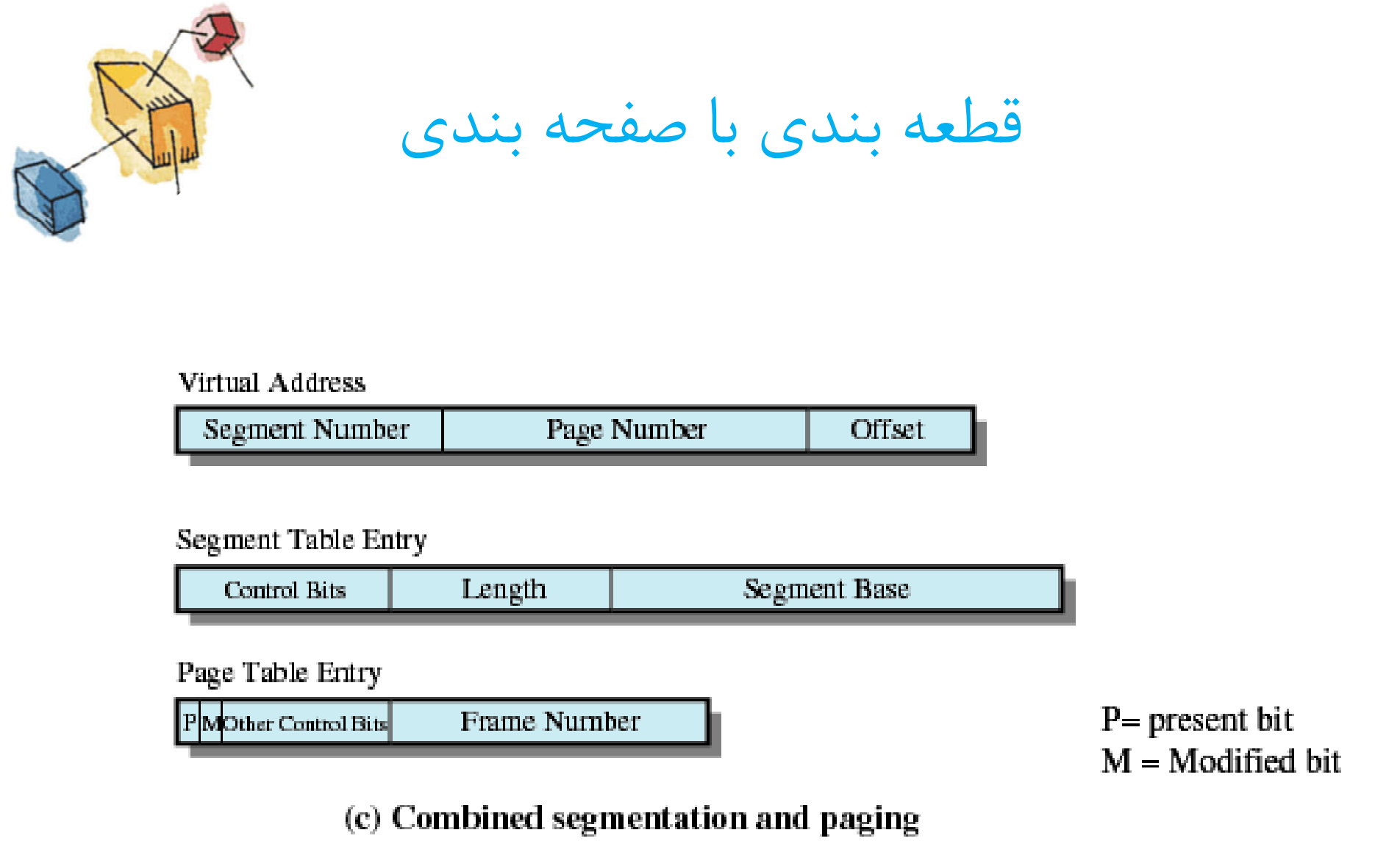


قطعه بندی با صفحه بندی

در این روش فرآیند ابتدا از نظر منطقی به قطعه ها تقسیم می شود.

سپس هر قطعه، صفحه بندی می شود (صفحاتی با اندازه مساوی، مانند آنچه که در صفحه بندی گفته شد).

قطعه بندی با صفحه بندی



(c) Combined segmentation and paging

قطعه بندی با صفحه بندی

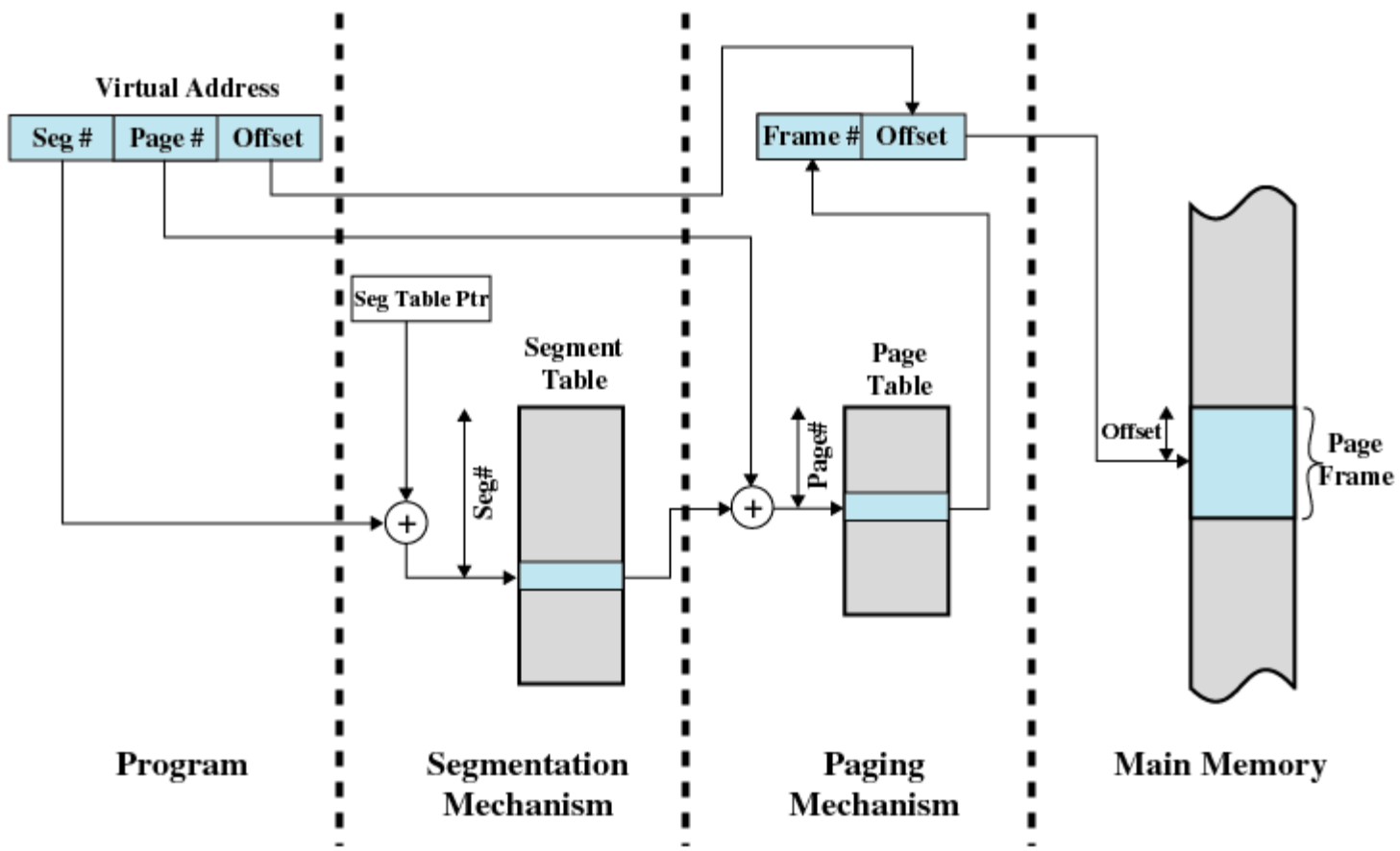
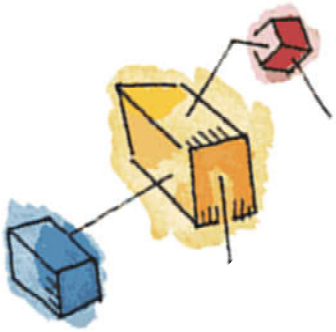


Figure 8.13 Address Translation in a Segmentation/Paging System

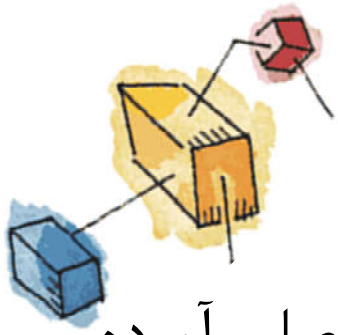


سرفصل مطالب

1. معرفی حافظه مجازی
2. صفحه بندی و صفحه بندی با حافظه مجازی
3. قطعه بندی و قطعه بندی با حافظه مجازی
4. سیاست های نرم افزاری برای حافظه مجازی

سیاست های نرم افزاری برای حافظه مجازی

- سه سیاست نرم افزاری در سیستم عامل برای مدیریت حافظه مجازی:
  - سیاست واکشی (Fetch Policy)
  - سیاست جاگذاری
  - سیاست جایگزینی



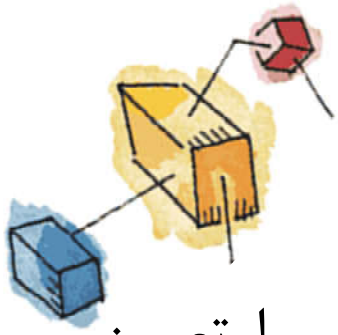
سیاست واکشی (Fetch Policy)

- مشخص می کند که یک صفحه چه هنگام بایستی به حافظه اصلی آورده شود. دو رویکرد معمول:

1. **صفحه بندی درخواستی (demand paging)**: فقط زمانی یک صفحه به حافظه اصلی آورده می شود که مراجعه ای به مکانی از آن صفحه انجام گیرد.
   - هنگام شروع برنامه، خطاهای صفحه زیادی رخ می دهد.
2. **پیش صفحه بندی (prepaging)**: در پیش صفحه بندی، صفحه هایی بیش از آنچه به دلیل خطای صفحه درخواست شده است، به داخل حافظه آورده می شوند.
   - صفحاتی که بطور پیوسته روی دیسک قرار دارند، اگر به حافظه آورده شوند، موثرتر است.

سیاست های نرم افزاری برای حافظه مجازی

- سه سیاست نرم افزاری در سیستم عامل برای مدیریت حافظه مجازی:
  - سیاست واکشی (Fetch Policy)
  - سیاست جایگذاری
  - سیاست جایگزینی



سیاست جایگذاری

- سیاست جایگذاری، محل قرار گرفتن فرآیند در حافظه اصلی را تعیین می کند.
  - در یک سیستم قطعه بندی، سیاست جایگذاری بسیار مهم است.
    - سیاست هایی مثل اولین برازش، بهترین برازش، ...
  - در سیستم صفحه بندی (یا ترکیب صفحه بندی و قطعه بندی) سیاست جایگذاری مهم نیست.

سیاست های نرم افزاری برای حافظه مجازی

- سه سیاست نرم افزاری در سیستم عامل برای مدیریت حافظه مجازی:
  - سیاست واکشی (Fetch Policy)
  - سیاست جایگذاری
  - سیاست جایگزینی

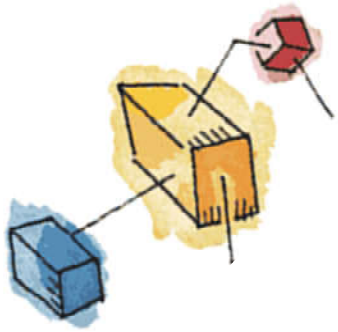
سیاست جایگزینی

- هنگامی که تمام قاب های حافظه اصلی اشغال باشند و لازم باشد صفحه جدیدی به حافظه آورده شود، کدام صفحه جایگزین شود؟

- صفحه حذف شده باید صفحه ای باشد که در آینده نزدیک احتمال کمتری برای مراجعه به آن وجود دارد.

- اکثر سیاست ها رفتار آینده را بر اساس رفتار گذشته پیش بینی می کنند.

- سیاست جایگزینیِ پیچیده تر، سربار سخت افزاری و نرم افزاری بیشتری در پی دارد.



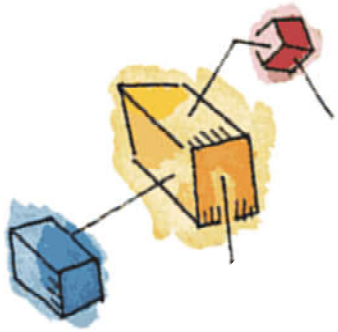
سیاست جایگزینی

- قفل کردن قاب

– صفحه ای که در یک قابِ قفل شده باشد، نمی تواند جایگزین شود.

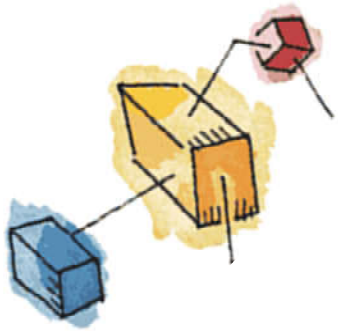
- هسته و ساختارهای کنترلی اصلی سیستم عامل، در قاب های قفل شده هستند.
- میانگیرهای ورودی/خروجی و دیگر نواحی بحرانی می توانند در قاب های قفل شده قرار گیرند.

– یک بیت قفل به هر قاب اختصاص می یابد.



الگوریتم های جایگزینی پایه

- سیاست بهینه (Optimal)
- سیاست حداقل استفاده در گذشته نزدیک (LRU)
- سیاست خروج به ترتیب ورود (FIFO)
- سیاست ساعت



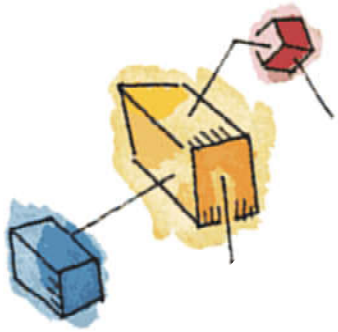
الگوریتم های جایگزینی پایه

- **سیاست بهینه** (Optimal)

– صفحه ای را برای جایگزینی انتخاب می کند که فاصله زمانی تا مراجعه بعدی به آن طولانی ترین باشد.

– کمترین تعداد خطای صفحه را در پی خواهد داشت.

– ولی امکان اجرای این الگوریتم وجود ندارد، زیرا نیازمند دانش کامل سیستم عامل از وقایع آینده است.



الگوریتم های جایگزینی پایه

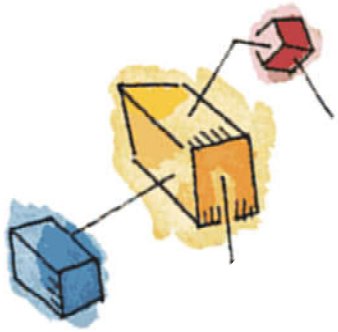
- **حداقل استفاده در گذشتهٔ نزدیک(LRU)**

– صفحه ای را حذف و جایگزین می کند که مدت طولانی تری مورد مراجعه قرار نگرفته است.

– بر اساس اصل محلیت، این صفحه باید صفحه ای باشد که کمترین احتمال مراجعه در آینده نزدیک را داشته باشد.

– روش های مختلفی برای پیاده سازی این روش ارائه شده است. مثلا هر صفحه را می توان با زمان آخرین مراجعه به آن، برچسب گذاری کرد.

– ولی این کار سربار (overhead) زیادی ایجاد خواهد نمود.



الگوریتم های جایگزینی پایه

- **خروج به ترتیب ورود (FIFO)**

– با قاب های تخصیص یافته به فرآیند مانند یک بافر حلقوی برخورد می کند.

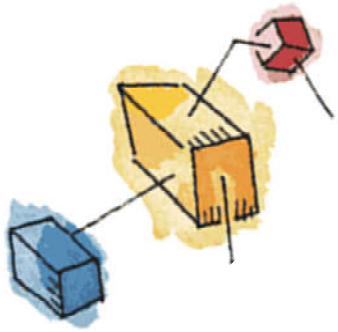
– صفحات به سبک نوبت-گردشی از حافظه خارج می شوند.

– تنها به یک اشاره گر نیاز دارد که بطور چرخشی به قاب های فرآیند اشاره کند.

– ساده ترین پیاده سازی را دارد.

– صفحه ای که بیشترین مدت در حافظه بوده، جایگزین می شود.

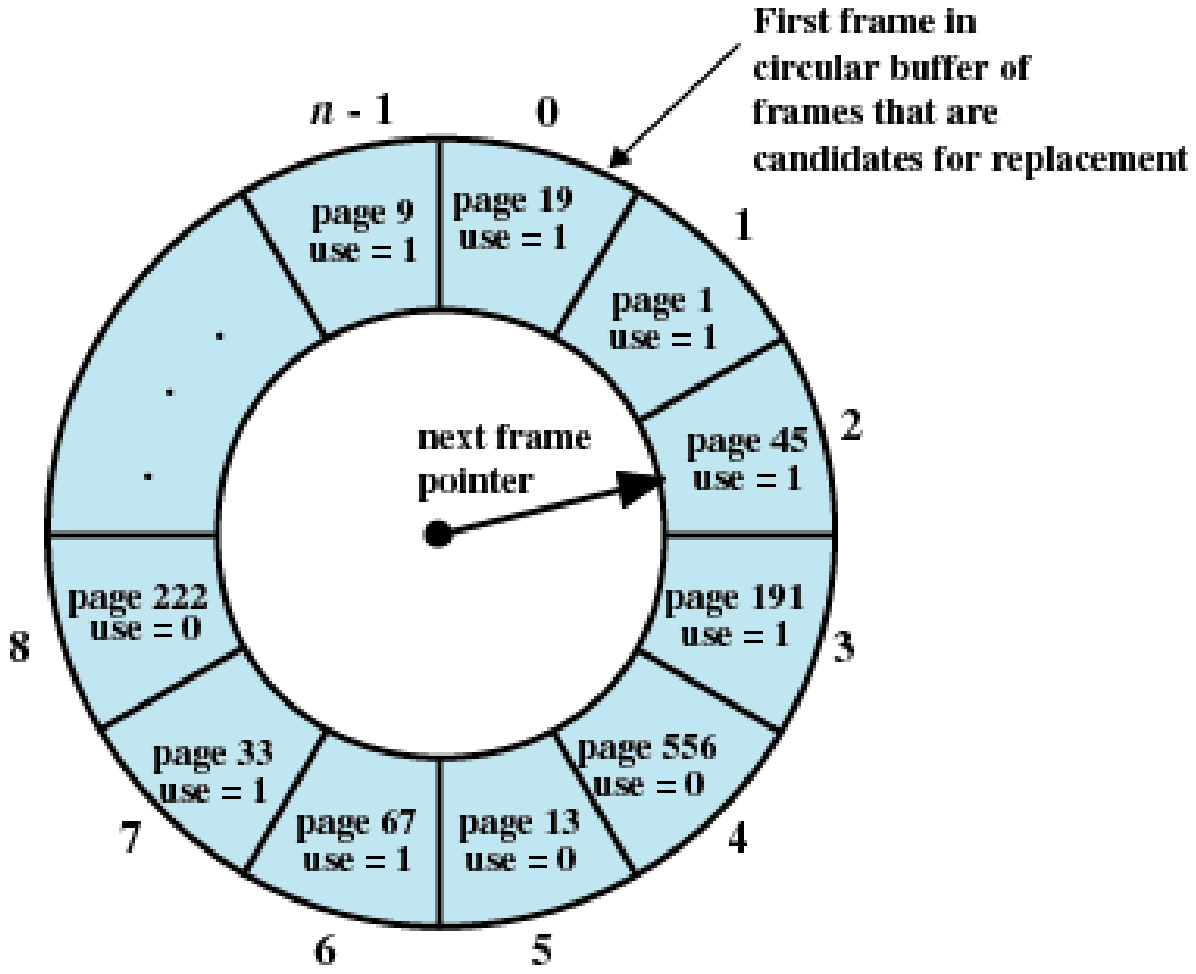
– ولی صفحات خارج شده ممکن است مجددا و به زودی مورد نیاز باشند.



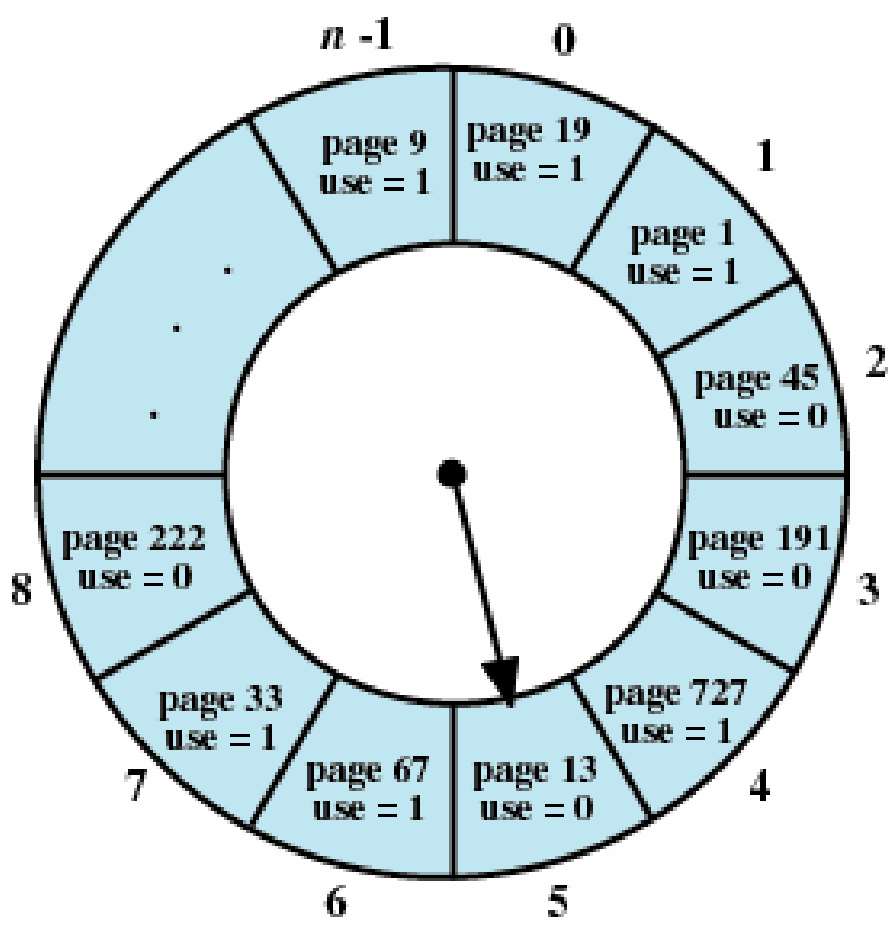
الگوریتم های جایگزینی پایه

- **سیاست ساعت** (Clock Policy)
  - شبیه سیاست FIFO، با این تفاوت که از یک بیت اضافی که به آن **بیت استفاده** می گویند نیز استفاده می شود.
  - اولین بار که یک صفحه به داخل قابی در حافظه اصلی بار شود، این بیت ۱ است.
  - در حین جستجو برای جایگزینی، وقتی الگوریتم با قابی با بیت استفاده ۱ مواجه می شود، بیت استفاده آن قاب از ۱ به صفر تغییر می کند.
  - زمانی که به صفحه ای در حافظه اصلی مراجعه شود، این بیت ۱می شود (یعنی مراجعه ای جدید به صفحه، پس از صفر کردن آن).
  - هنگامیکه زمان جایگزینی صفحه فرا می رسد، سیستم عامل بافر حلقوی را مرور کرده و اولین قابی که بیت استفاده آن صفر باشد را برای جایگزینی انتخاب می کند.

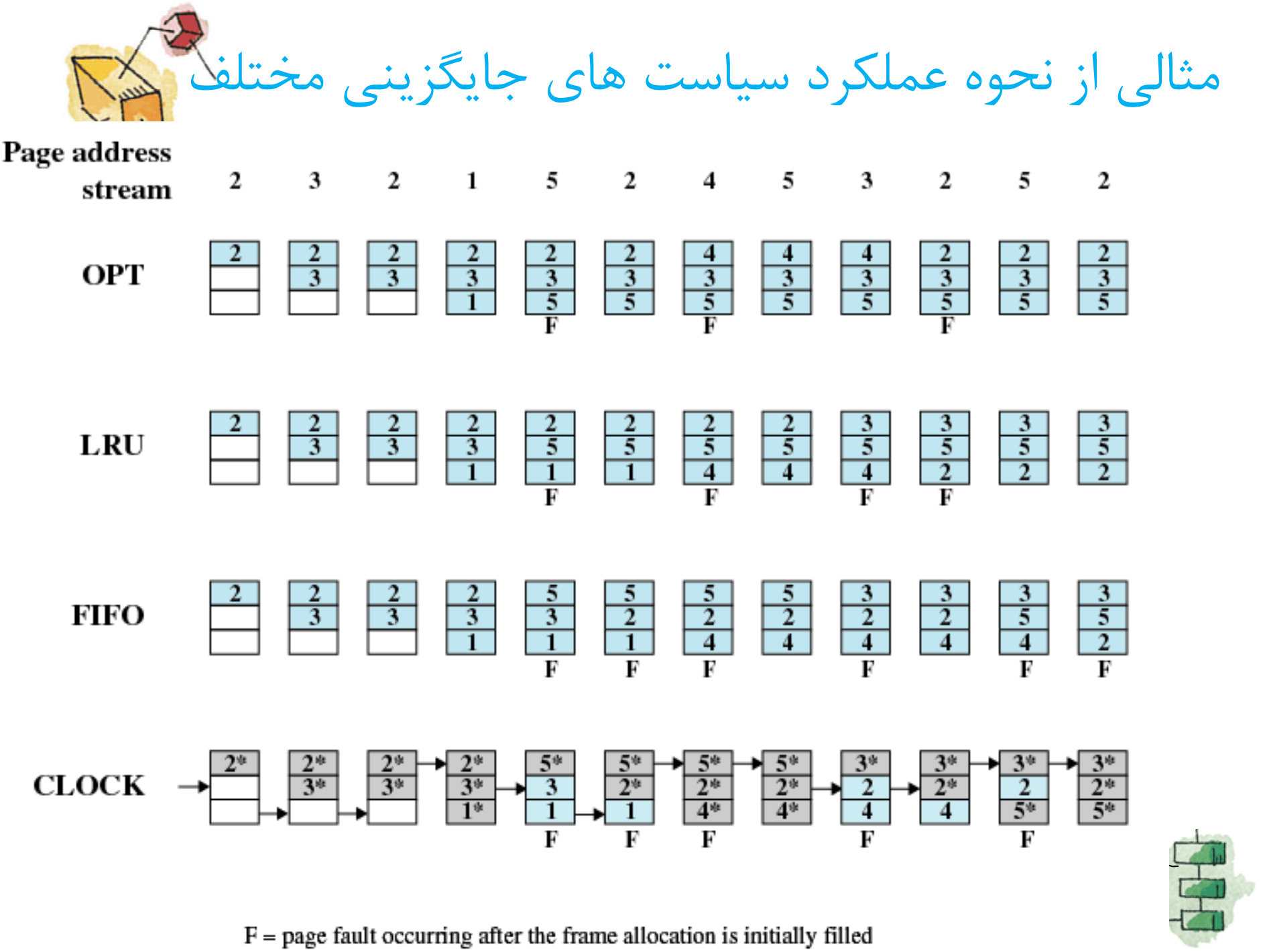
وضعیت بافر درست قبل از یک جایگزینی صفحه

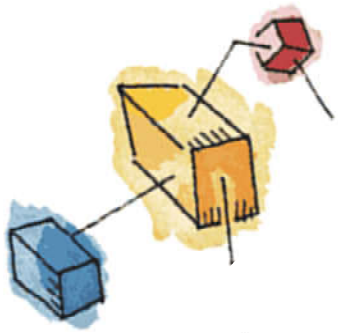


وضعیت بافر درست بعد از یک جایگزینی صفحه



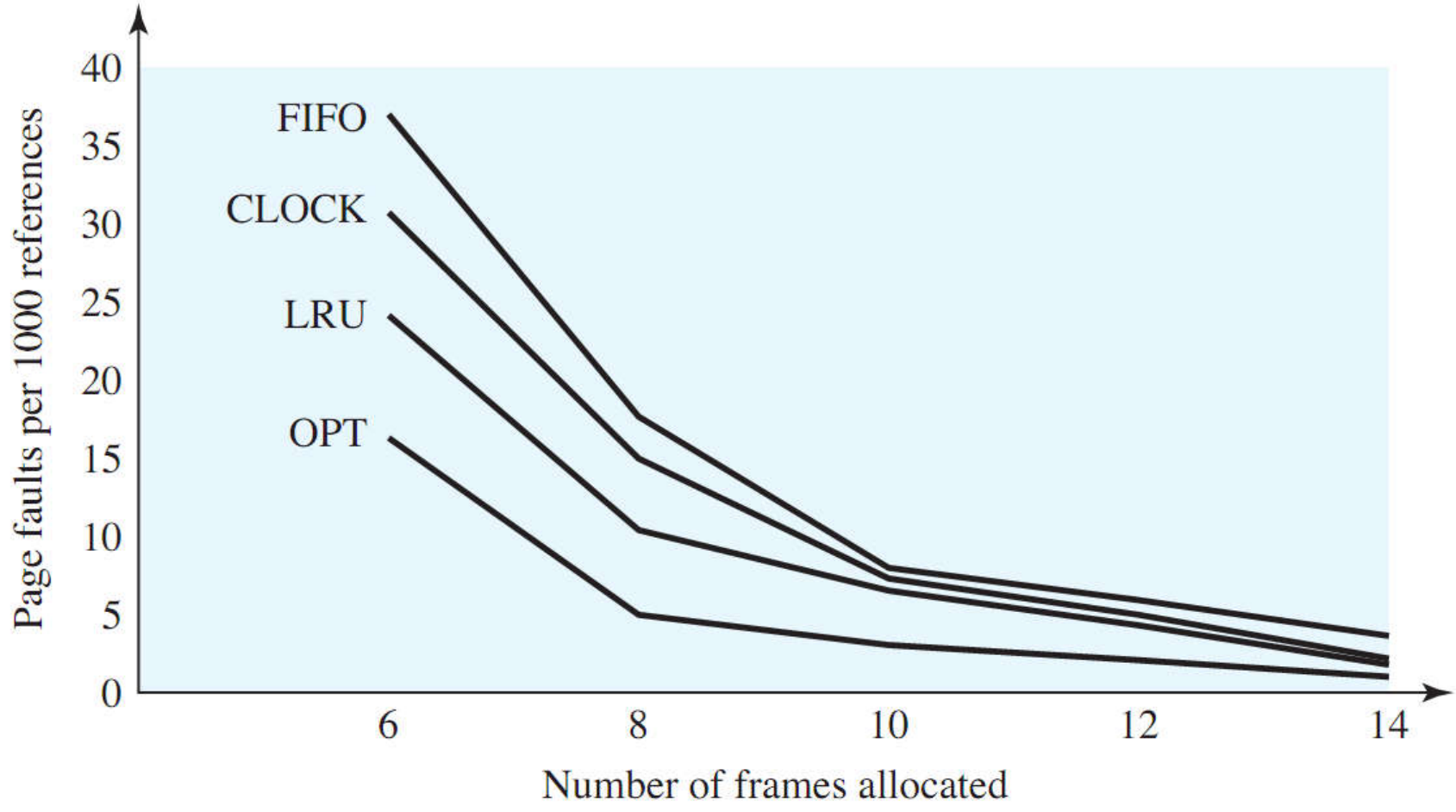
مثالی از نحوه عملکرد سیاست های جایگزینی مختلف

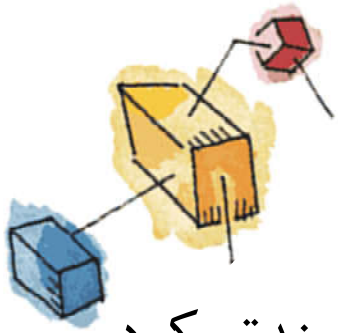




مقایسه الگوریتم های جایگزینی صفحه

در حالت تخصیص ثابت و دیدگاه محلی

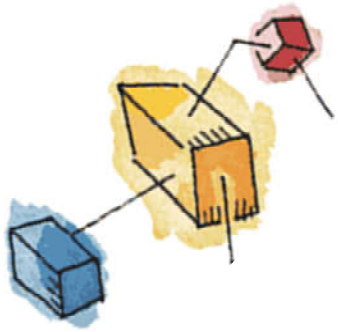




قدرتمندتر کردن الگوریتم ساعت

- می توان الگوریتم ساعت را با استفاده از بیت های بیشتر قدرتمندتر کرد.
- برای این منظور، بیت های استفاده (u) و تغییر (m) را همراه هم در نظر می گیریم.

- هر قاب در یکی از دسته بندی های زیر قرار می گیرد:
  - اخیرا دستیابی نشده، تغییر نیافته (u=0 , m=0)
  - اخیرا دستیابی شده، تغییر نیافته (u=1 , m=0)
  - اخیرا دستیابی نشده، تغییر یافته (u=0 , m=1)
  - اخیرا دستیابی شده، تغییر یافته (u=1 , m=1)



قدرتمندتر کردن الگوریتم ساعت

- **مراحل الگوریتم:**

1. با شروع از موقعیت فعلی اشاره گر، قاب ها را مرور کن. در حین مرور در بیت استفاده تغییری ایجاد نکن. اولین قابی که برای آن u=0 , m=0 است، انتخاب می شود.
2. در صورت شکست مرحله اول، قاب ها را برای یافتن قابی با u=0 , m=1 مرور کن. این بار، بیت استفاده هر قابی که مرور می شود، صفر می شود.
3. در صورت شکست مرحله دوم، اشاره گر باید به موقعیت اولیه بازگردانده شود و تمام قاب های مجموعه دارای بیت استفاده صفر خواهند بود. مرحله اول را تکرار کن. این بار قابی برای جایگزینی پیدا خواهد شد.

مثال

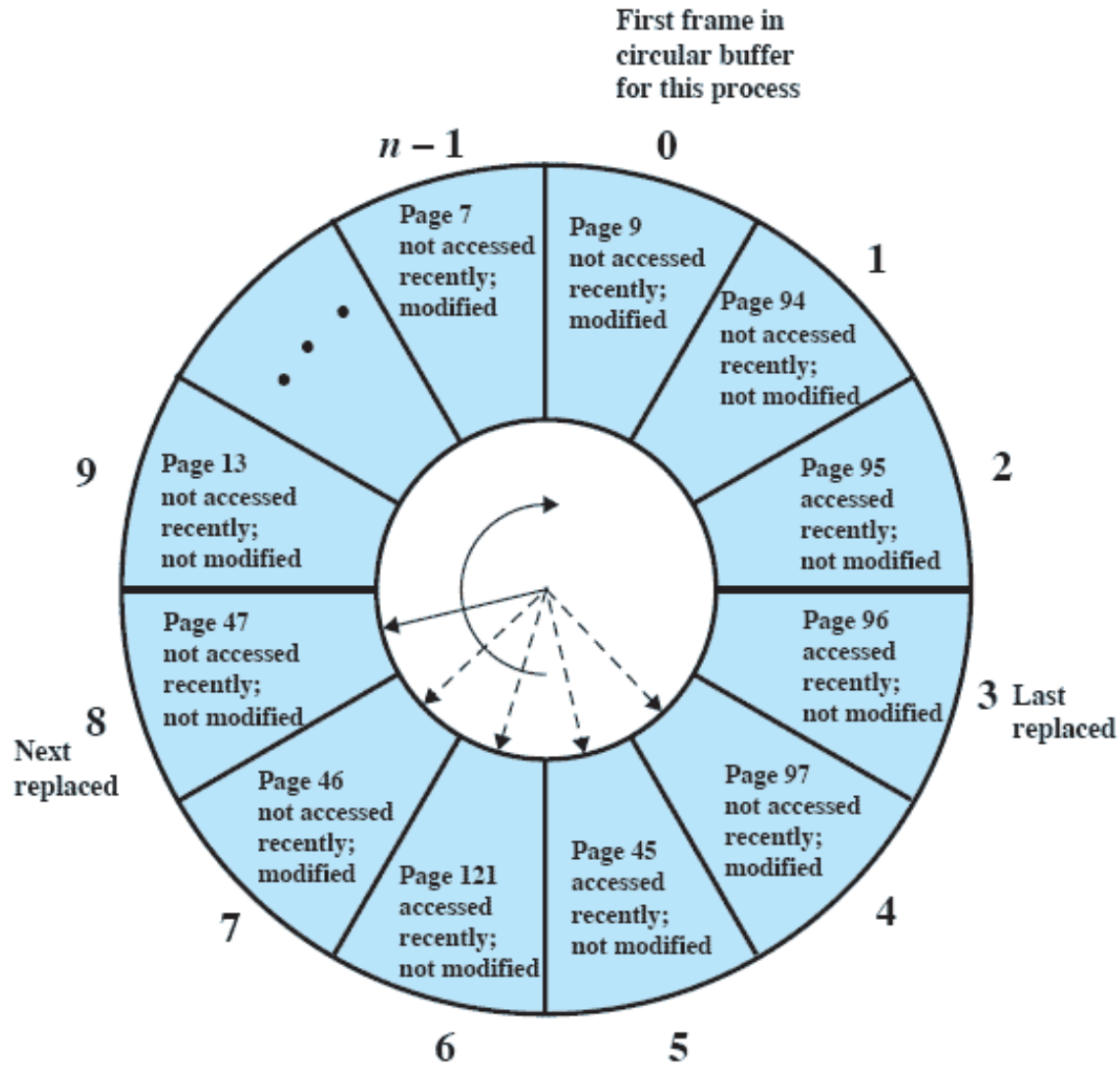
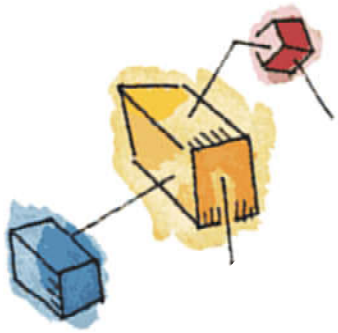
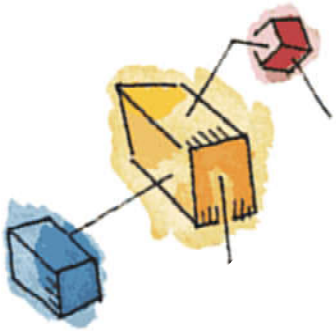


Figure 8.18 The Clock Page-Replacement Algorithm [GOLD89]



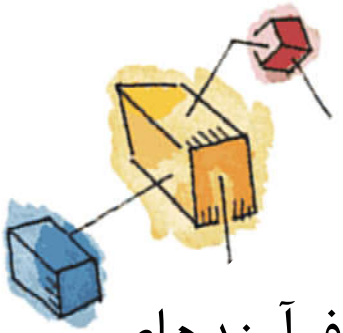
مدیریت مجموعه مقیم (Resident Set)



مدیریت مجموعه مقیم

- اندازه مجموعه مقیم
  - چه مقدار از حافظه اصلی به یک فرآیند به خصوص تخصیص داده شود.

- قلمروی جایگزینی
  - سیاست محلی یا سراسری

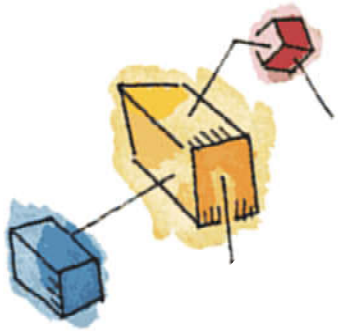


اندازه مجموعه مقیم

- هرچه حافظه تخصیص یافته به یک فرآیند کمتر باشد، تعداد فرآیندهای بیشتری می توانند در هر لحظه در حافظه باشند.

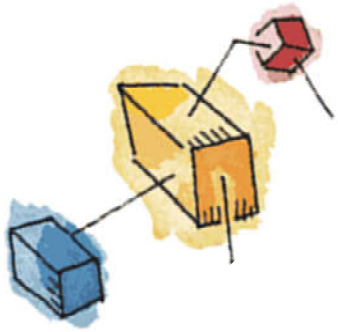
- ولی اگر تعداد کمی از صفحه های یک فرآیند در حافظه اصلی باشند نرخ خطای صفحه زیاد می شود.

- از طرف دیگر، تخصیص حافظه بیشتر از یک حد مشخص، به دلیل اصل محلی بودن ارجاعات، تاثیر قابل توجه ای بر روی نرخ خطای صفحه نخواهد داشت.



اندازه مجموعه مقیم

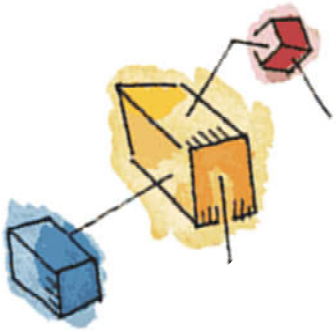
- تخصیص ثابت
  - تعداد ثابتی قاب در حافظه اصلی به یک فرآیند می دهد (در زمان بار کردن اولیه و بر اساس نوع فرآیند)
- تخصیص متغیر
- تعداد قاب تخصیص یافته به هر فرآیند، می تواند در دوره زندگی آن فرآیند متغیر باشد.
  - فرآیندی که اصل محلی بودن ارجاعات در آن زیاد صدق نمی کند و دائم دچار خطای صفحه می شود، قاب های اضافی دریافت می کند.
  - برای فرآیندی که خیلی کم دچار خطا صفحه می شود (اصل محلی بودن در آن زیاد صدق می کند)، تعداد قاب های تخصیص یافته کاهش می یابد.
  - نیازمند ارزیابی دائم یک فرآیند (سربار زیاد)



قلمروی جایگزینی

- سیاست جایگزینی محلی
  - در انتخاب صفحه برای جایگزینی، فقط از بین صفحه های مقیم مربوط به همان فرآیندی که ایجاد خطای صفحه کرده، صفحه ای انتخاب می شود.

- سیاست جایگزینی سراسری
  - کلیه صفحات داخل حافظه اصلی را به عنوان کاندید برای جایگزینی در نظر می گیرد و توجه ندارد که کدام صفحه به کدام فرآیند اختصاص دارد.



پایان فصل هشتم