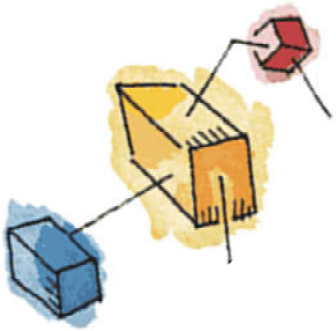
به نام خدا

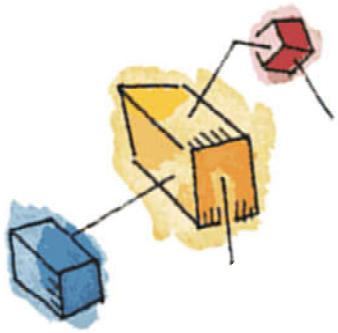
فصل هشتم:
حافظه مجازی (بخش اول)

Virtual Memory



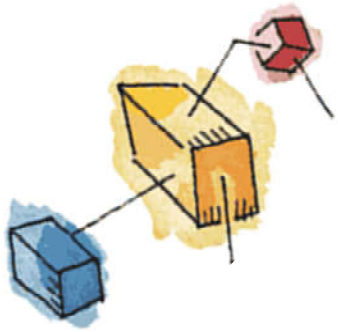
سرفصل مطالب

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



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

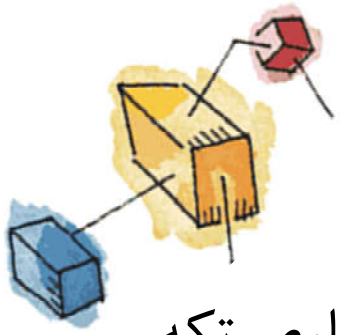


ساختارهای سخت افزاری و کنترلی

دو پیشرفت در مدیریت حافظه:

1. کلیه مراجعات یک فرآیند به حافظه، آدرس های منطقی هستند که به صورت پویا در زمان اجرا به آدرس فیزیکی ترجمه می شوند.
   - فرآیند ممکن است به داخل و خارج حافظه مبادله شود بنابراین ممکن است بخش های مختلفی از حافظه را اشغال کند.
2. یک فرآیند ممکن است به بخش هایی تقسیم شود که نیاز نیست به صورت پیوسته در حافظه قرار گیرند.

- اگر این دو ویژگی در سیستمی بر قرار باشند، برای اجرای فرآیند نیازی به بار شدن تمام بخش های فرآیند در حافظه اصلی نیست.

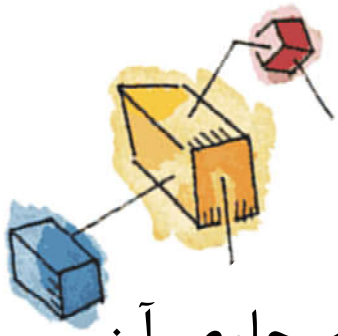


اجرای برنامه

- بنابراین سیستم عامل بخشی از فرآیند (یک یا چند تکه که حاوی تکه آغازین برنامه است) را به حافظه اصلی بار می کند.
- مجموعه مقیم: بخشی از فرآیند که در حافظه است.

- زمانی که پردازنده به یک آدرس منطقی نیاز دارد که در حافظه وجود ندارد، وقفه ای تولید می شود.

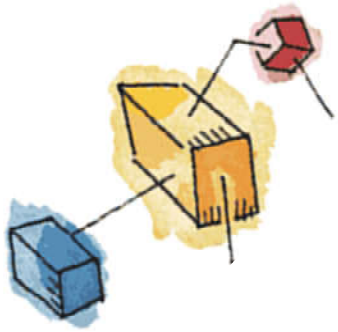
- سیستم عامل فرآیند وقفه خورده را مسدود نموده و کنترل را بدست می گیرد تا وقفه را مدیریت کند.



اجرای برنامه

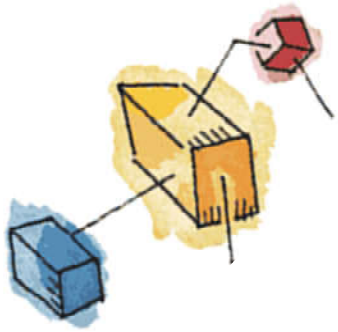
- برای مدیریت این وقفه، سیستم عامل آن قسمت از فرآیند که حاوی آن آدرس منطقی و تولید کننده خطای حافظه است را به حافظه اصلی می آورد.

- برای این منظور:
  - سیستم عامل یک درخواست I/O برای خواندن به دیسک صادر می کند.
  - در حین عمل I/O سیستم عامل فرآیند دیگری را اجرا می کند.
  - زمانی که عمل I/O کامل شد، وقفه ای صادر می شود و موجب می شود فرآیند مسدود به حالت آماده تغییر حالت دهد.



مزایای تکه تکه کردن و بخش به بخش آوردن فرآیند روی حافظه اصلی

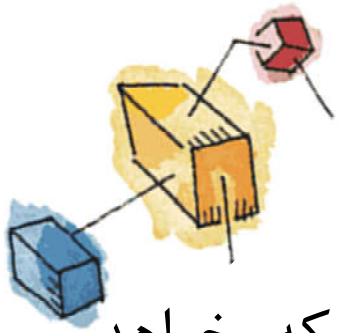
1. فرآیندهای بیشتری می توانند در حافظه نگه داشته شوند.
   - تنها بخشی از فرآیند به حافظه بار می شود.
   - با داشتن فرآیندهای بیشتر در حافظه احتمال وجود فرآیند آماده به اجرا بیشتر می شود و این موجب افزایش کارایی پردازنده می شود.
2. فرآیند می تواند از حافظه بزرگتر باشد.



انواع حافظه

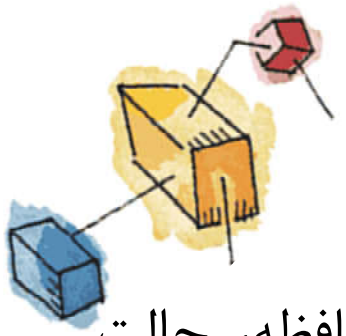
- حافظه حقیقی
  - حافظه اصلی (Main Memory)

- حافظه مجازی (Virtual Memory)
  - در واقع بخشی از حافظه روی دیسک است.
  - چندبرنامگی را بصورت موثرتری ممکن ساخته و کاربر را از محدودیت های سخت حافظه اصلی رها می کند.



کوبیدگی (Thrashing)

- چنانچه سیستم عامل تکه ای از یک فرآیند را دقیقا قبل از اینکه بخواهد مورد استفاده قرار گیرد از حافظه اصلی خارج کند، لازم است با فاصله کمی مجددا آن را به داخل حافظه بیاورد.
- درصورت وقوع مکرر این اتفاق، پردازنده بیشتر وقت خود را به جای اجرای دستورات و برنامه های کاربر، صرف مبادله تکه ها می کند.
- به چنین شرایطی **کوبیدگی** گفته می شود.
- برای جلوگیری از این مشکل، سیستم عامل سعی می کند حدس بزند که کدام قسمت های حافظه در آینده نزدیک مورد استفاده قرار خواهند گرفت و آن قسمت ها را از حافظه خارج نکند:
  - این حدس بر اساس تاریخچه اخیر انجام می شود.



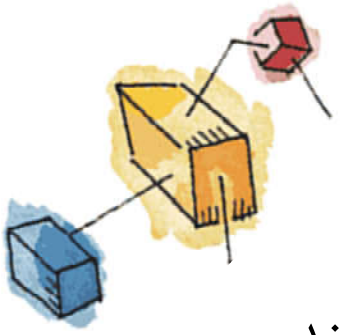
اصل محلیت (Locality)

- درون یک فرآیند مراجعات به برنامه و داده های موجود در حافظه، حالت خوشه ای و نزدیک به هم دارند.

- در یک بازه زمانی کوتاه فقط به تعداد کمی از تکه های یک فرآیند نیاز است.

- می توان حدس های هوشمندانه ای زد که کدام تکه های یک فرآیند در آینده نزدیک مورد نیاز خواهند بود.

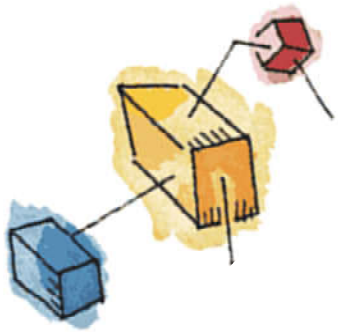
- بنابراین می توان از حافظه مجازی به شکل موثری بهره گرفت. در واقع، اصل محلی بودن به این اشاره دارد که حافظه مجازی می تواند مفید باشد و کار کند.



پشتیبانی های مورد نیاز برای حافظه مجازی

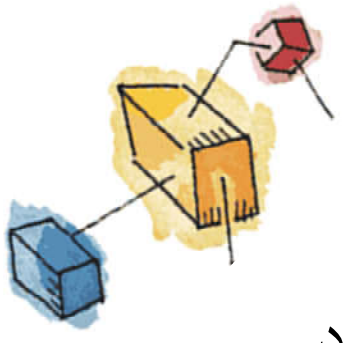
- سخت افزار بایستی از صفحه بندی یا قطعه بندی پشتیبانی کند.

- سیستم عامل باید قادر به مدیریت انتقال صفحه ها و یا قطعه ها بین حافظه ثانویه و حافظه اصلی باشد.



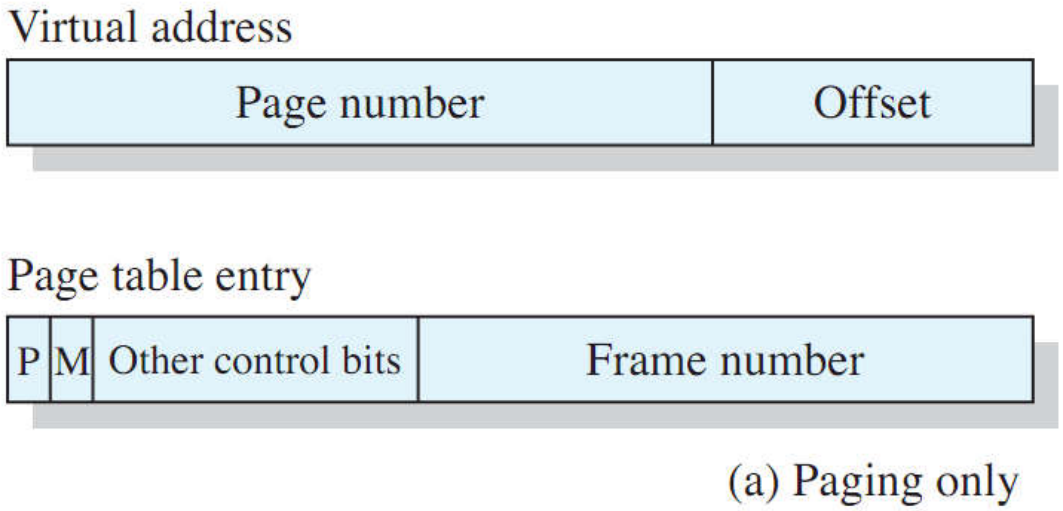
سرفصل مطالب

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

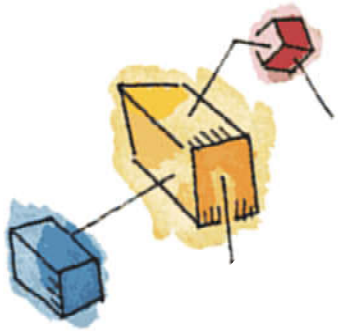


صفحه بندی

- هر فرآیند جدول صفحه (page table) مخصوص به خود دارد.
- هر ورودی (مدخل یا سطر) جدول صفحه شامل شماره قاب متناظر با آن صفحه در حافظه اصلی است.



(a) Paging only



بیت حضور و تغییر در جدول صفحه

- بیت حضور (**P**)
  - هر ورودی جدول صفحه دارای یک بیت حضور است.
  - بیت حضور مشخص می کند که آیا صفحه مربوطه در حافظه اصلی قرار دارد یا خیر.
- بیت تغییر (**M**)
  - مشخص می کند که آیا محتوای صفحه از زمان آخرین بار شدن در حافظه اصلی تغییر نموده است یا خیر.
  - اگر هیچ تغییری در محتوای صفحه صورت نگرفته باشد، هنگام تعویض صفحه نیازی به نوشتن آن بر روی دیسک نیست.

ترجمهٔ آدرس در یک سیستم صفحه بندی

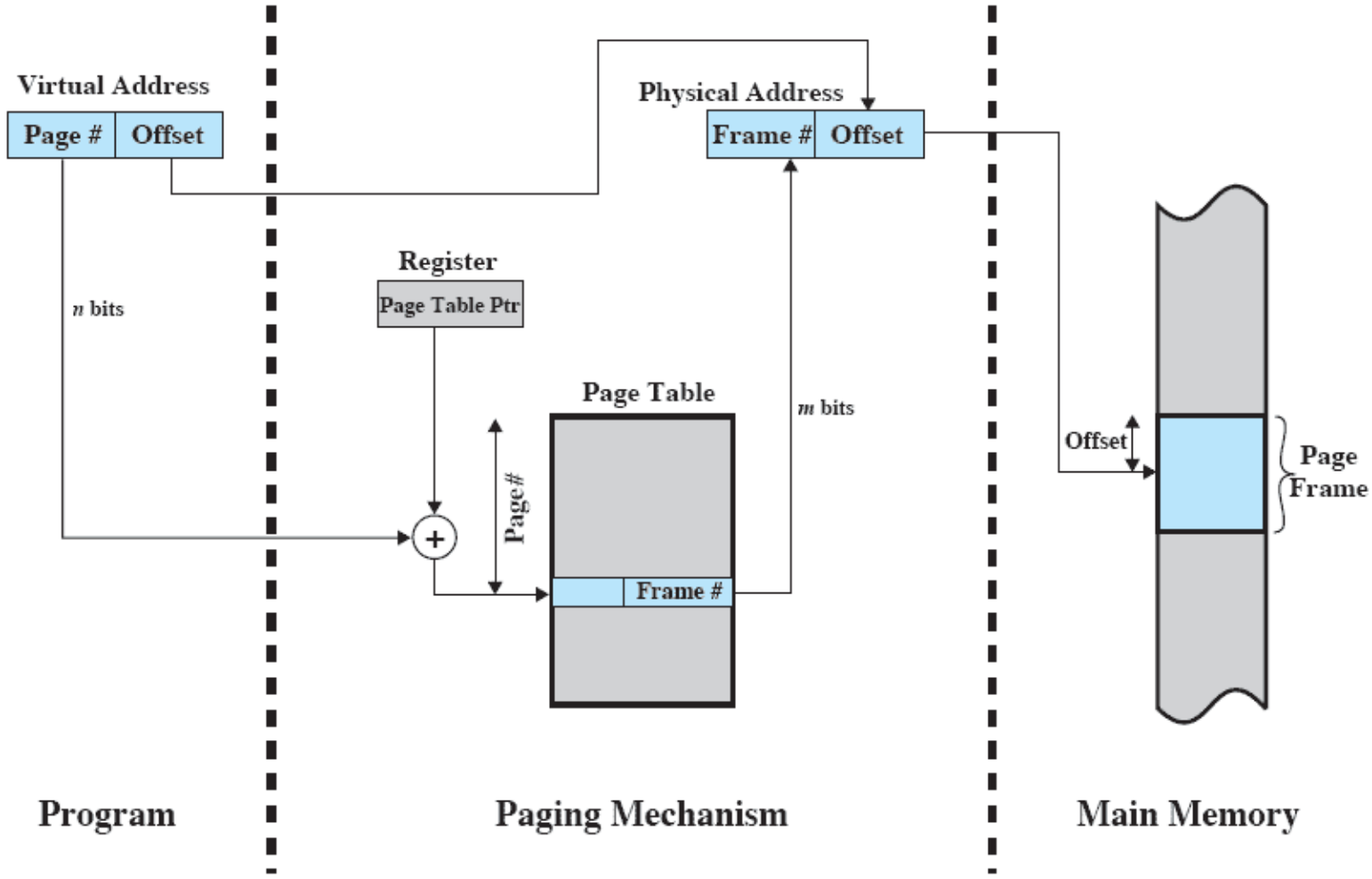
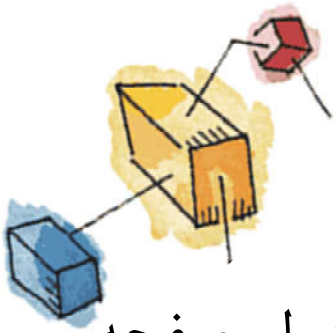


Figure 8.3 Address Translation in a Paging System

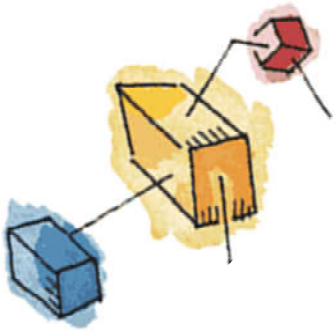


جدول صفحه

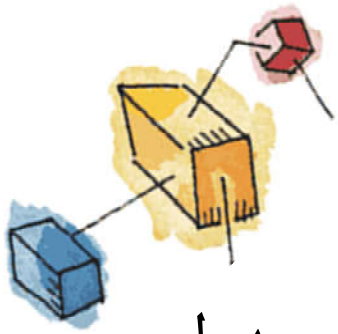
- اگر برنامه بزرگ باشد، صفحات زیادی دارد و بنابراین، کل جدول صفحه ممکن است مقدار زیادی از حافظه اصلی را اشغال کند.

- جداول صفحه نیز خودشان صفحه بندی می شوند و در حافظه مجازی ذخیره می شوند.

- هنگامی که یک فرآیند در حال اجرا است، بخشی از جدول صفحه آن که شامل مدخل صفحه در حال اجرا است، در حافظه اصلی قرار دارد.



- جدول صفحه دو سطحی

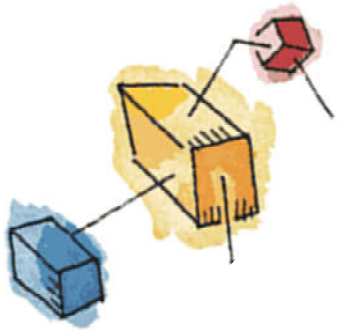


جدول صفحه دو سطحی

- بعضی پردازنده ها از طرح دو سطحی برای سازماندهی جدول های بزرگ صفحه استفاده می کنند.

- در این طرح یک فهرست راهنمای صفحه وجود دارد که هر مدخل آن به یک جدول صفحه اشاره می کند.

- اگر اندازه فهرست راهنما X باشد و حداکثر طول یک جدول صفحه Y باشد، یک فرآیند می تواند X*Y تا صفحه داشته باشد.



مثالی از جدول صفحه دوسطحی

- در سیستمی آدرس های منطقی ۳۲ بیتی هستند.
- فرض: آدرس دهی در سطح بایت انجام می شود (نه کلمه).
- و اندازه صفحه ۴ کیلوبایت (یعنی 2^{12}) است.
- بنابراین، فضای آدرس مجازی، ۴ گیگابایت است (یعنی 2^{32}) که شامل 2^{20} صفحه می شود.
- اگر هر کدام از این صفحه ها توسط یک مدخل صفحه ۴ بایتی نگاشته شوند، جدول صفحه ای با 2^{20} مدخل به وجود می آید که نیازمند ۴ مگابایت (یعنی 2^{22} بایت) است.
- این جدول صفحه عظیم، خودش 2^{10} صفحه است که در حافظه مجازی است و توسط یک جدول صفحه ریشه (یعنی جدول صفحه ی جدول صفحه ها) شامل 2^{10} مدخل نگاشته می شود.
- جدول صفحه ریشه، ۴ کیلوبایت (2^{12} بایت) از حافظه اصلی را اشغال می کند و همیشه در حافظه اصلی است.

جدول صفحه دو سطحی

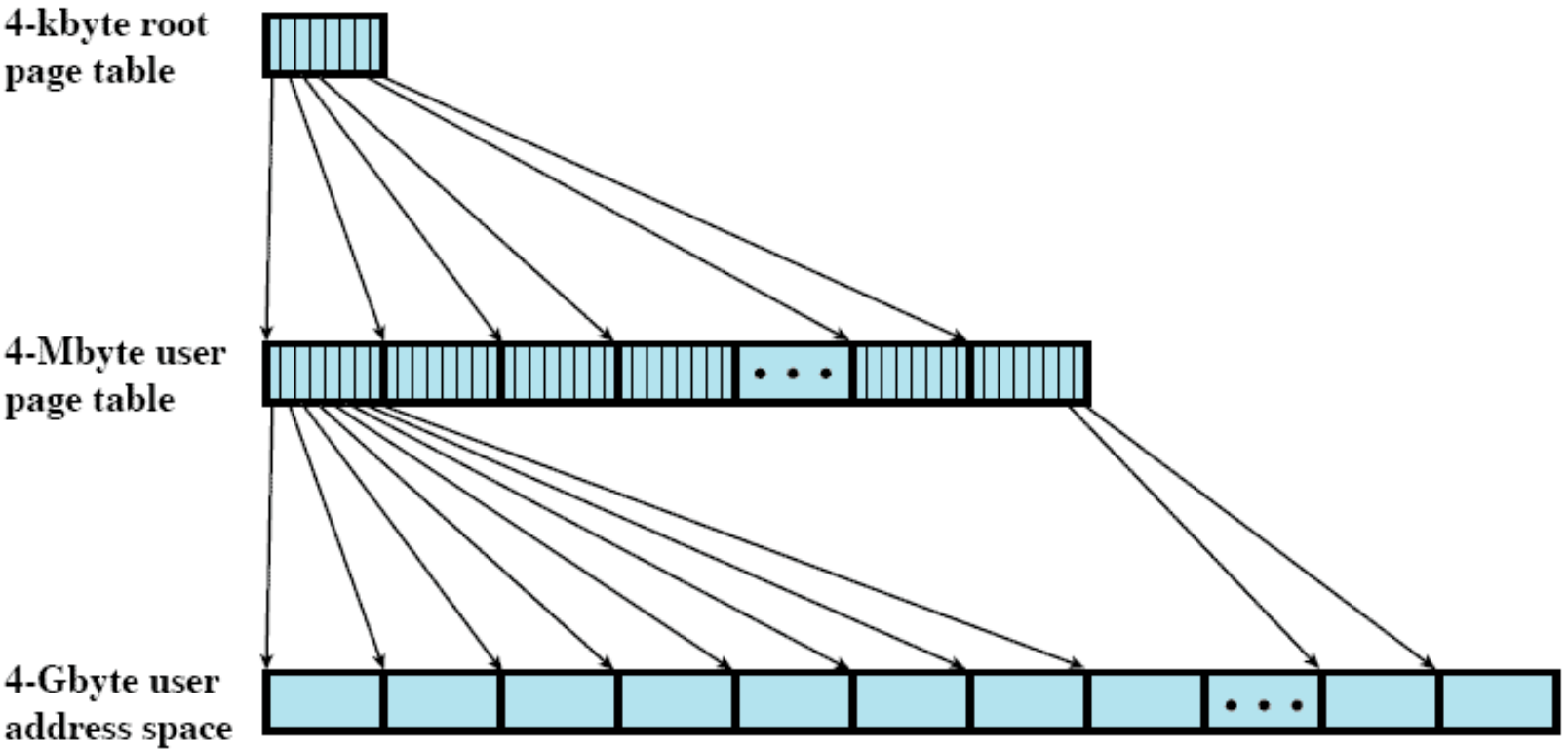


Figure 8.4 A Two-Level Hierarchical Page Table

ترجمه آدرس در جدول صفحه سلسله مراتبی (Hierarchical page table)

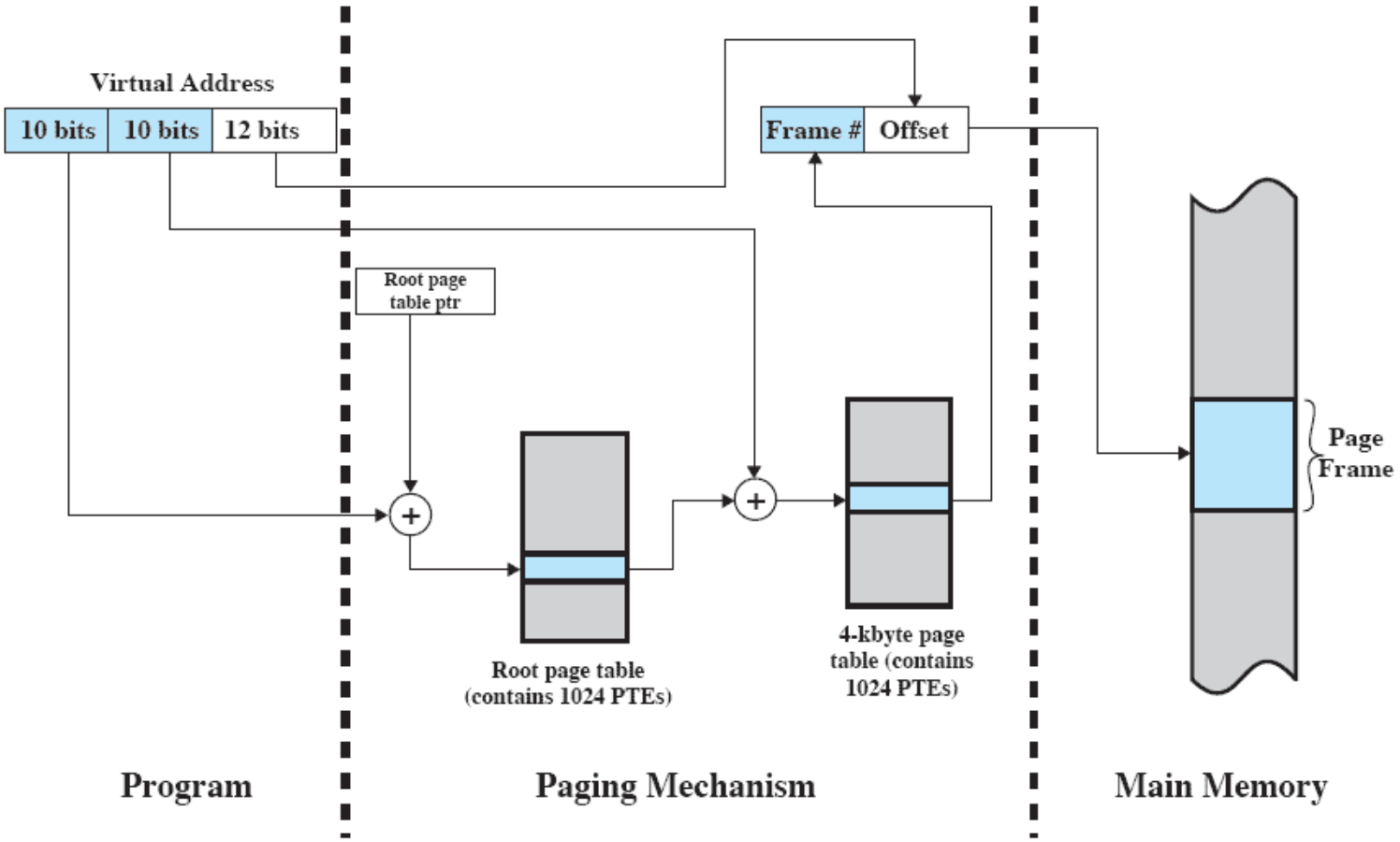
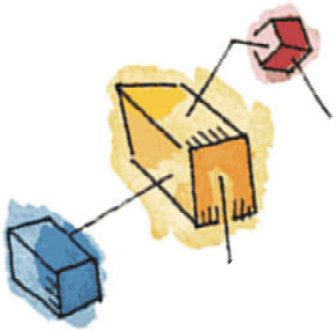


Figure 8.5 Address Translation in a Two-Level Paging System



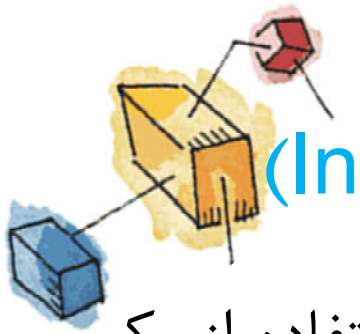
- جدول صفحه معکوس

جدول صفحهٔ معکوس (Inverted Page Table)

- یک اشکال جدول صفحه هایی که تا کنون موردبحث قرار گرفتند این است که اندازه آنها متناسب با فضای آدرس مجازی است.

- رویکرد دیگر برای جداول صفحه یک سطحی یا دوسطحی، استفاده از ساختار جدول صفحه معکوس است.

- عنوان معکوس به این علت است، که هر مدخل از این نوع جدول صفحه، مربوط به یک قاب است و در داخل آن مدخل، صفحه یک فرآیند که در آن قاب قرار دارد، ذکر شده است.

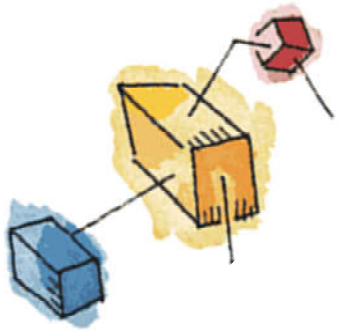


جدول صفحهٔ معکوس (Inverted Page Table)

- در این رویکرد، قسمت شماره صفحه از آدرس مجازی با استفاده از یک تابع درهم ساز ساده به یک جدول درهم (hash table) نگاشته می شود.

- جدول درهم به جدول صفحه معکوس اشاره می کند.

- مزیت: بدون توجه به تعداد فرآیندها و یا صفحه های مجازی، بخش ثابتی از حافظه حقیقی برای نگهداری جدول ها مورد نیاز است.

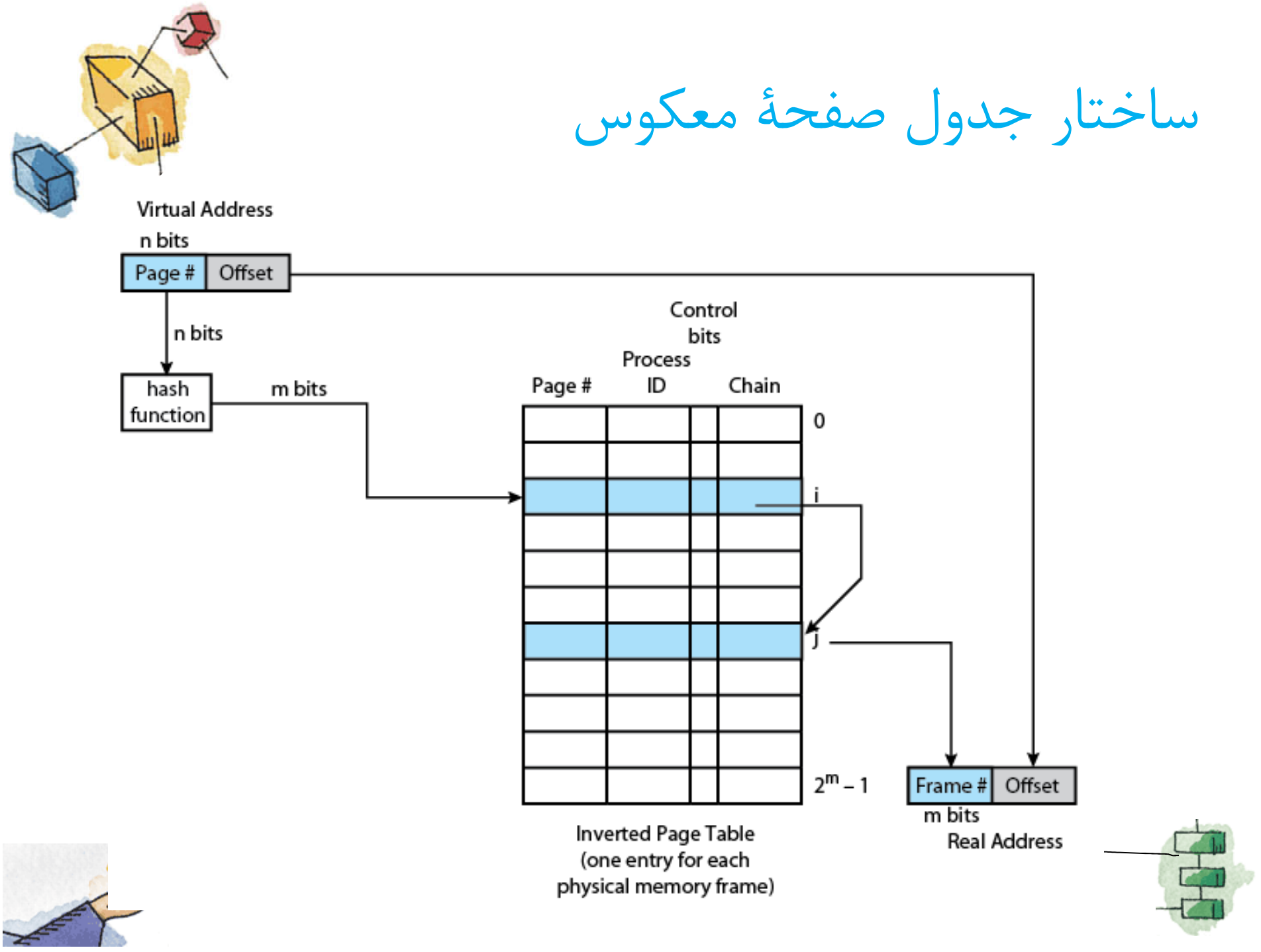


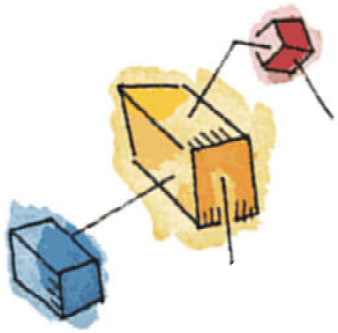
جدول صفحهٔ معکوس

- جدول صفحه معکوس شامل موارد زیر است:
  - شماره صفحه
  - شناسهٔ فرآیند (فرآیندی که مالک این صفحه است)
  - بیت های کنترلی (شامل پرچم هایی مثل اعتبار، ارجاع، و ...)
  - اشاره گر زنجیره (اندیس مدخل بعدی در زنجیره).

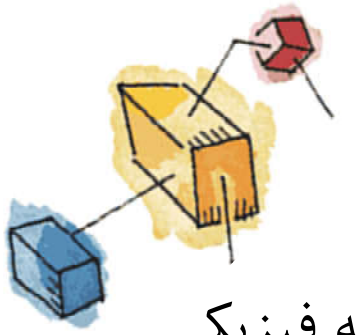
- زیرا ممکن است در تابع درهم ساز، بیش از یک آدرس مجازی به یک مدخل جدول، hash شود. در این شرایط از یک روش زنجیره ای برای اداره سرریز استفاده می شود.

ساختار جدول صفحهٔ معکوس





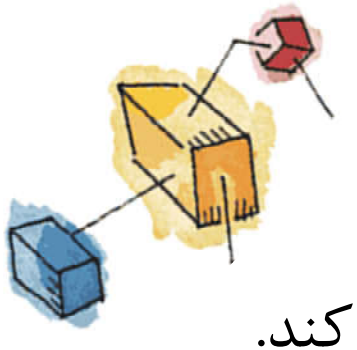
- میانگیر دم دستی ترجمه (یا TLB)



بافر (میانگیر) دم دستی ترجمه (Translation Lookaside Buffer)

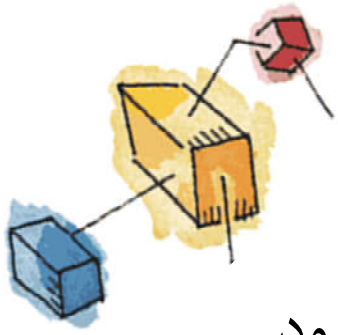
- هر ارجاع به حافظه مجازی می تواند موجب دو دسترسی به حافظه فیزیکی شود:
  - یکی برای واکشی جدول صفحه مربوطه
  - یکی برای واکشی داده مورد نظر

- پس یک نتیجه منفی استفاده از حافظه مجازی، دو برابر شدن زمان دسترسی به حافظه است.
- برای غلبه بر این مشکل از یک حافظه نهان بسیار سریع برای رکوردهای (مدخل های) جدول صفحه استفاده می شود.
- به این حافظه نهان، بافر دم دستی ترجمه (TLB) گفته می شود.
- TLB حاوی رکوردها یا مدخل هایی از جدول صفحه است که اخیرا مورد استفاده قرار گرفته اند.



بافر(میانگیر) دم دستی ترجمه

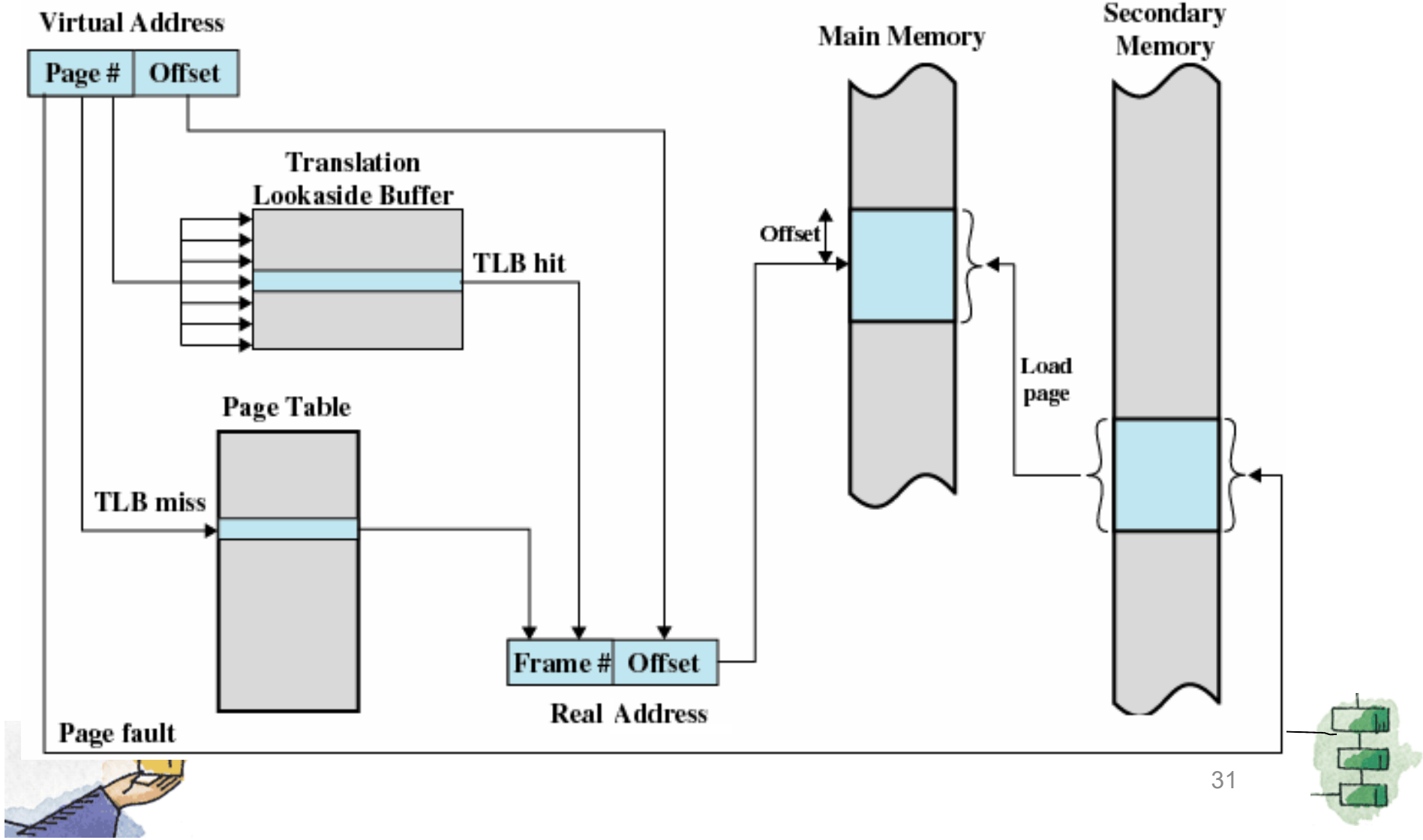
- برای ترجمه یک آدرس مجازی، پردازنده TLB را بررسی می کند.
  - اگر مدخل مورد نظر جدول صفحه پیدا شد (اصابت یا TLB hit)، شماره قاب بازیابی می شود و آدرس حقیقی شکل می گیرد.
  - اگر مدخل مورد نظر از جدول صفحه پیدا نشد (عدم اصابت TLB miss)، پردازنده از شماره صفحه به عنوان اندیس (شاخص) جدول صفحه فرآیند استفاده نموده و مدخل مورد نظر را از جدول صفحه بازیابی می کند.



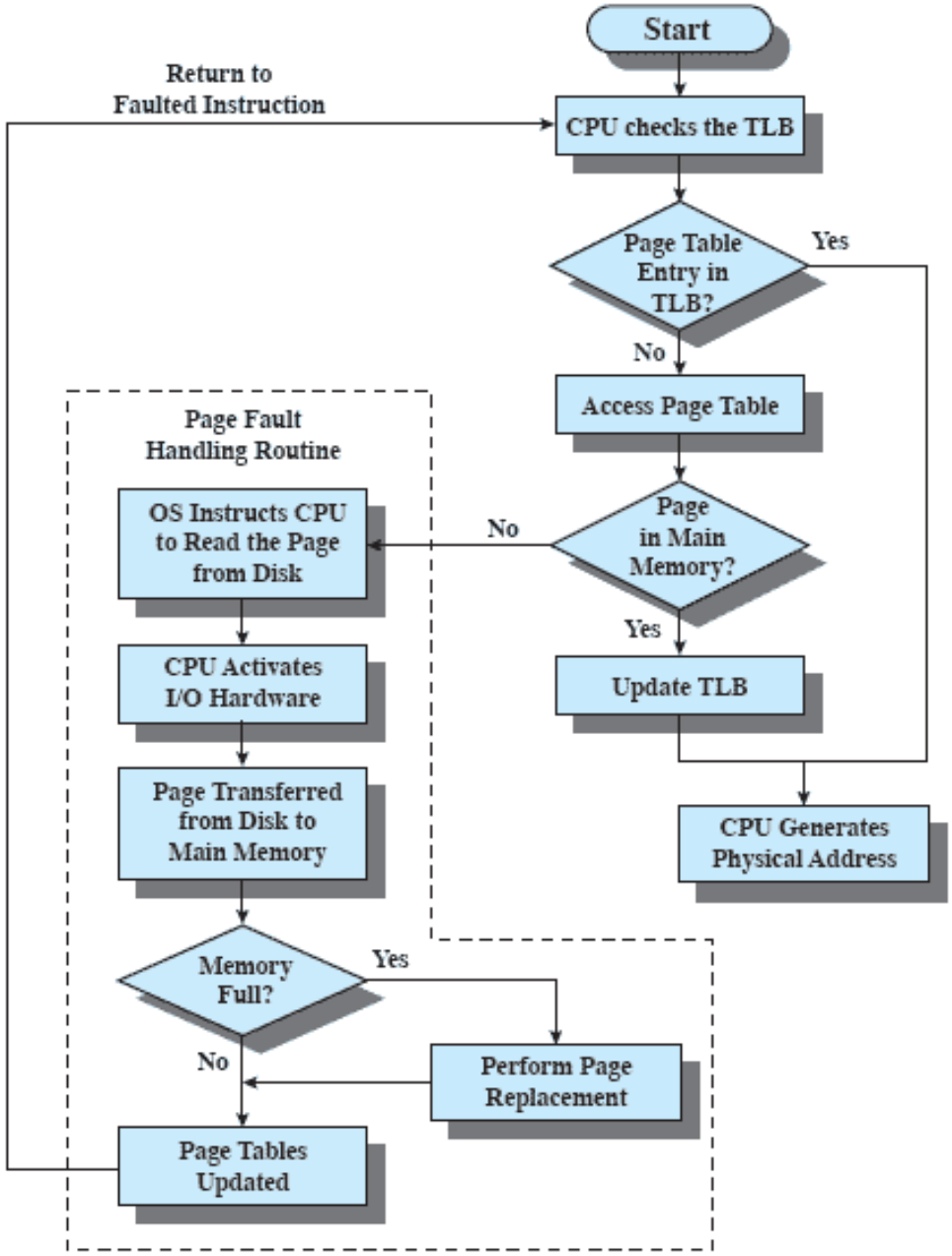
بافر(میانگیر) دم دستی ترجمه

- حال، ابتدا بیت حضور (حضور صفحه در حافظه) بررسی می شود.
  - اگر صفحه مورد نظر در حافظه اصلی نبود، خطای فقدان صفحه (Page Fault) صادر می شود.
  - اگر در حافظه اصلی بود، پردازنده شماره قاب را از مدخل جدول صفحه برای تشکیل آدرس حقیقی بدست می آورد و TLB را به روز می کند تا شامل این مدخل صفحه جدید شود.
- به طور کلی هدف از به کارگیری میانگیر دم دستی ترجمه (TLB)، کم کردن تعداد مراجعات به حافظه اصلی برای واکشی مدخل جدول صفحه است.

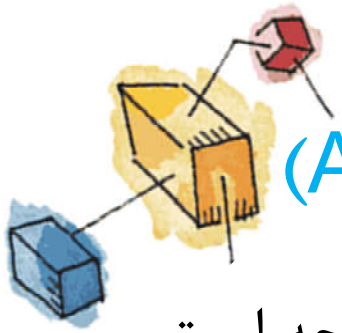
استفاده از میانگیر دم دستی ترجمه



عملیات استفاده از TLB



igure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB) [FURH87]



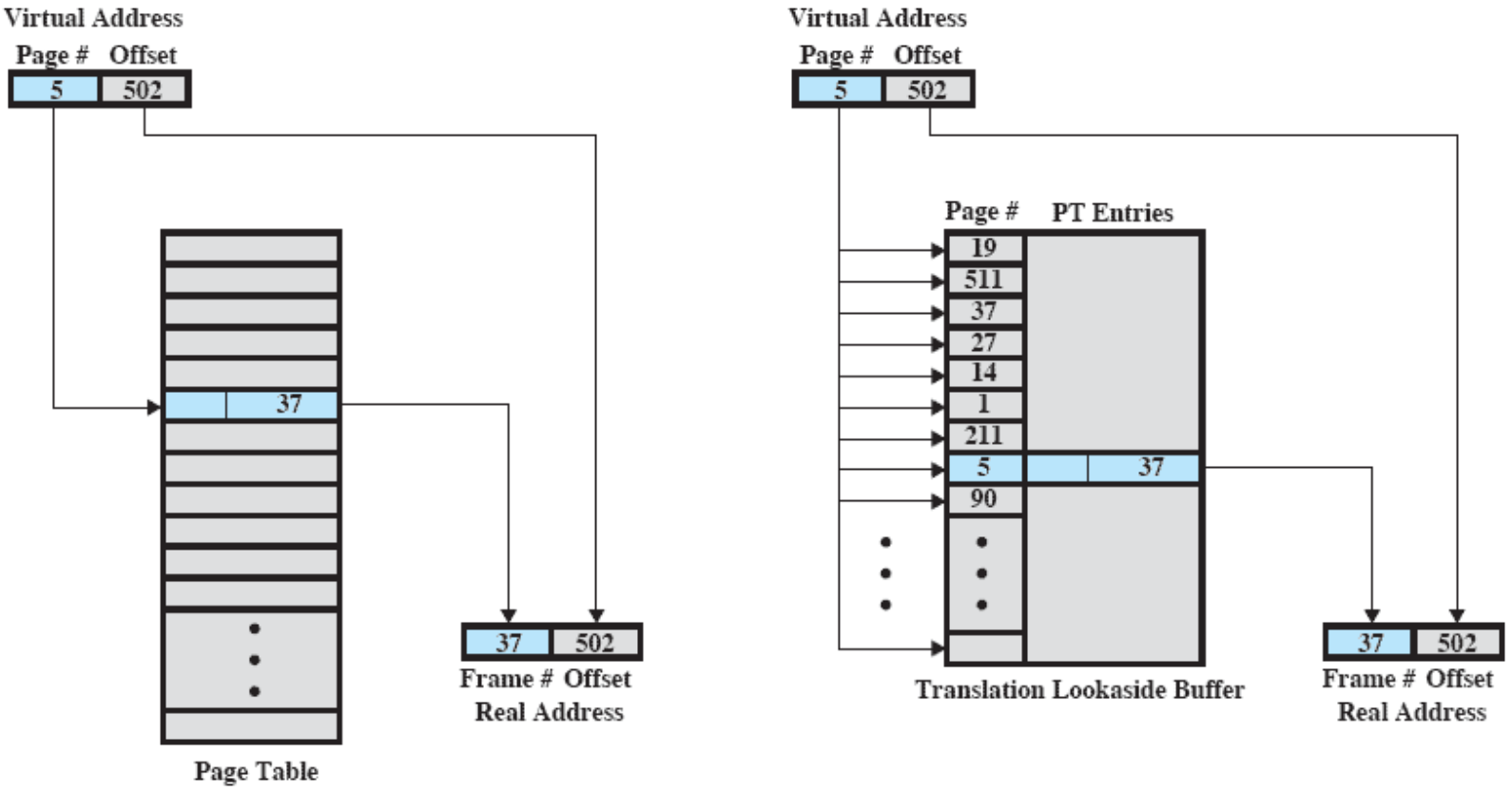
نگاشت انجمنی (Associative Mapping)

- میانگیر دم دستی فقط شامل برخی از مدخل های جدول صفحه است. بنابراین نمی توان صرفا بر اساس شماره صفحه به میانگیر دم دستی مراجعه کرد.

- در عوض، هر مدخل میانگیر دم دستی باید شامل شماره صفحه و مدخل کامل جدول صفحه باشد.

- پردازنده مجهز به سخت افزاری است که به او اجازه می دهد به طور همزمان تعدادی از مدخل های میانگیر را برای یافتن شماره صفحه وارسی کند.

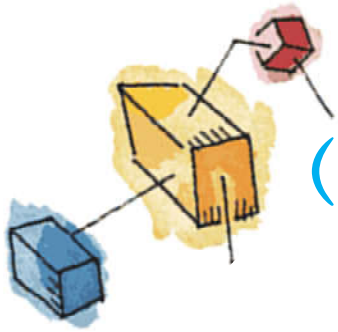
جستجوی مستقیم در مقابل جستجوی انجمنی برای مدخل های جدول صفحه



(a) Direct mapping

(b) Associative mapping

Figure 8.9 Direct Versus Associative Lookup for Page Table Entries



عملکرد TLB و حافظه نهان (cache)

- جدا از بحث هایی که تا کنون گفته شد:
- پس از تولید آدرس حقیقی (به هر نحوی)، حافظه نهان آزمایش می شود تا مشخص شود که آیا بلوکی حاوی کلمه موردنظر در آن است یا خیر.
  - اگر آن کلمه در حافظه نهان باشد، آن کلمه به پردازنده تحویل داده می شود.
  - در غیراین صورت، آن کلمه از حافظه اصلی بازیابی می گردد.

- بنابراین، کلمه مورد مراجعه می تواند در حافظه نهان، حافظه اصلی یا دیسک باشد.

عملکرد TLB و حافظه نهان (Cache)

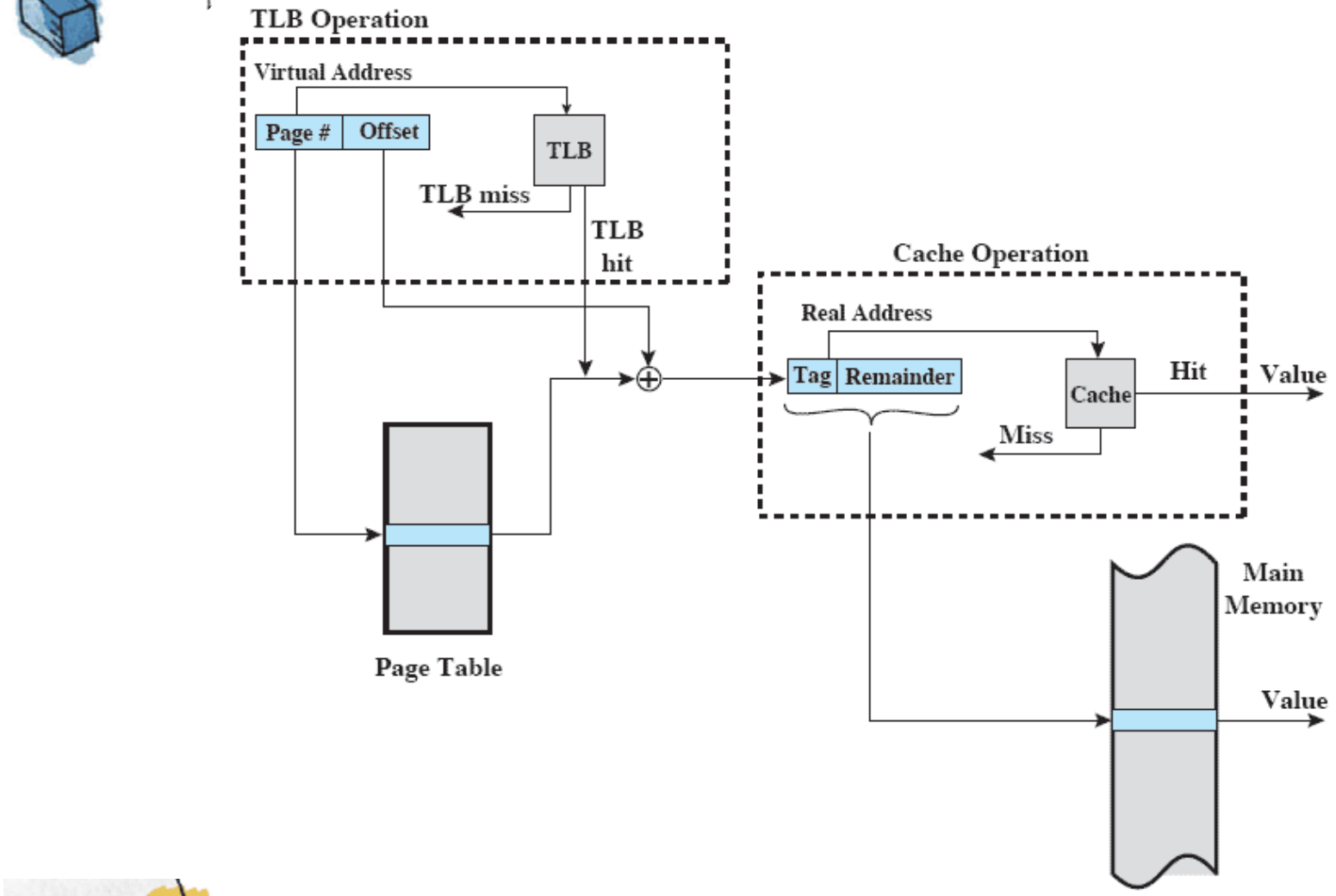


Figure 8.10 Translation Lookaside Buffer and Cache Operation