فصل هفتم

مدیریت حافظه

وحید رنجبر

پائیز ۹۸

مباحث این فصل:

- نیازهای مدیریت حافظه
  - جابجایی، حفاظت، اشتراک، سازمان منطقی، سازمان فیزیکی
- بخش بندی حافظه
  - بخش بندی ایستا
  - بخش بندی پویا
  - سیستم رقابتی
  - جابجایی
- صفحه بندی
- قطعه بندی

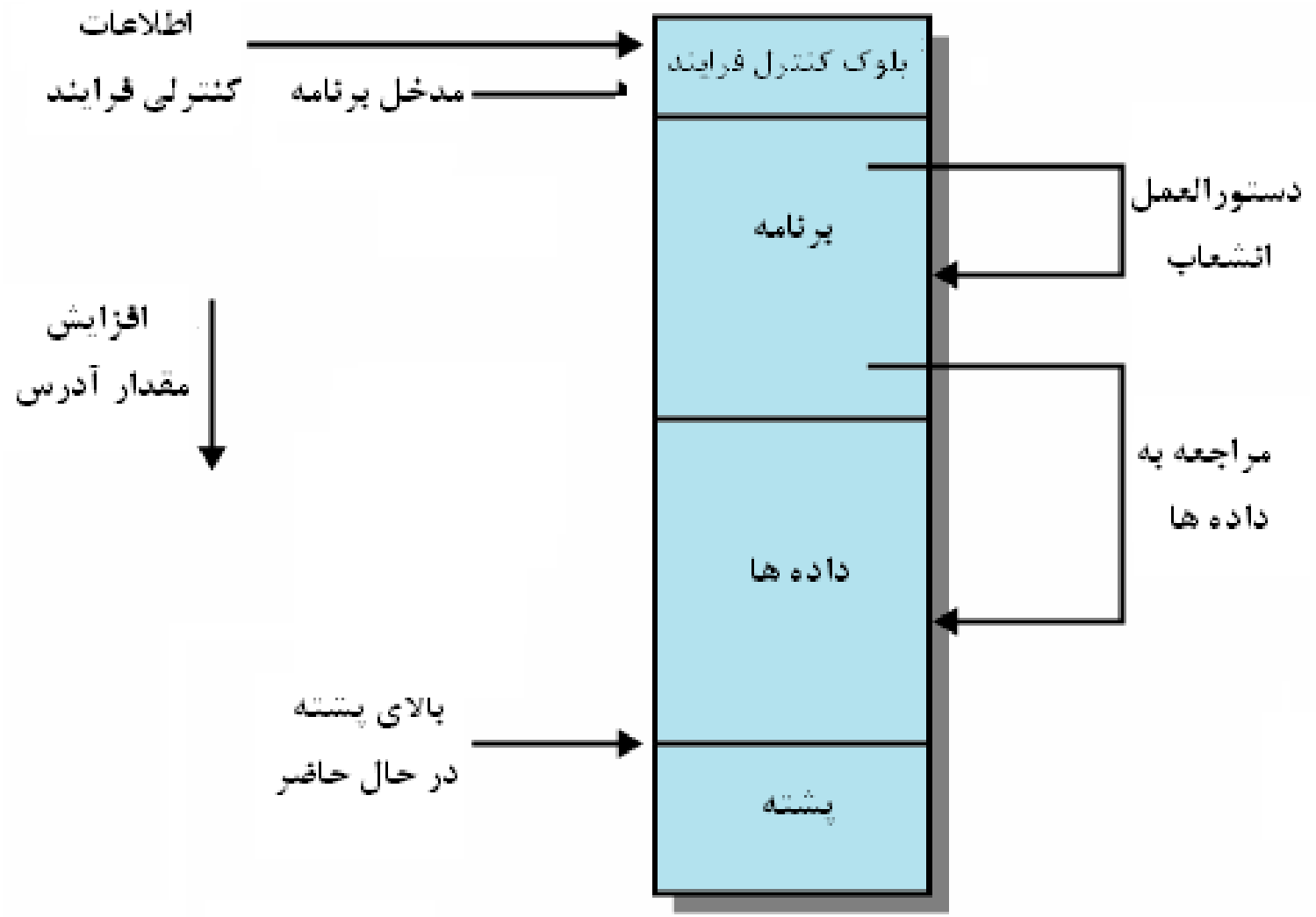
مدیریت حافظه:

- در یک سیستم تک برنامه ای حافظه به دو بخش تقسیم می شود:
  - یک بخش برای سیستم عامل (ناظر، مقیم، هسته)
  - یک بخش برای برنامه در حال اجرای کاربر
- در یک سیستم چند برنامه ای بخش کاربر باید تقسیم بندی شود تا چندین برنامه را همزمان در خود جای دهد.
- وظیفه تقسیم بندی حافظه به زیر بخشها به صورت پویا و توسط سیستم عامل صورت میگیرد و به این عمل مدیریت حافظه میگویند.
- حافظه باید به گونه ای تخصیص یابد که فرایندهای آماده بیشتری در آن مجتمع شوند.

نیازمندی های مدیریت حافظه:

- **جابجایی**
  - برنامه نویس نمی داند برنامه هنگام اجرا در کجای حافظه ذخیره میشود.
  - هنگام اجرای برنامه ممکن است برنامه به دیسک منتقل شود و دوباره در مکان دیگری از حافظه قرار گیرد.
  - ارجاعهای به حافظه باید به آدرسهای فیزیکی که منعکس کننده مکان فعلی برنامه در حافظه اند ترجمه شوند.

نمایی از نیازهای آدرس دهی فرایند:



نیازمندی های مدیریت حافظه:

- **حفاظت**
  - فرایندها نباید بدون اجازه قادر به مراجعه به اطلاعات سایر فرایندها باشند
  - به دلایل زیر بررسی آدرسهای فیزیکی در زمان ترجمه غیر ممکن است:
    - مکان برنامه در حافظه اصلی در زمان ترجمه مشخص نیست.
    - اکثر زبانهای برنامه سازی محاسبه پویای آدرس در زمان اجرا (مثل محاسبه شاخص یک آرایه یا اشاره گر به یک ساختمان داده) را اجازه میدهند.
  - یک فرایند کاربر نمیتواند به هیچ بخش سیستم عامل اعم از برنامه یا داده دسترسی داشته باشد.
  - نیازهای حفاظتی باید توسط پردازنده (سخت افزار) برآورده شود نه توسط سیستم عامل (نرم افزار).

نیازمندی های مدیریت حافظه:

- **اشتراک:**
  - اشتراک امکان دسترسی چندین فرایند به یک بخش یکسان از حافظه را میدهد.
  - اگر چند فرایند در حال اجرای یک برنامه هستند، به جای اینکه هر فرایند کپی جداگانه ای از برنامه داشته باشد بهتر است همه فرایندها به یک کپی از برنامه دسترسی داشته باشند.

نیازمندی های مدیریت حافظه:

- **سازمان منطقی:**
  - اکثر برنامه ها به صورت مؤلفه ای سازمان یافته اند.
  - مزایای کار کردن با مؤلفه ها:
    - هر مؤلفه را میتوان به طور مستقل نوشت و ترجمه کرد.
    - با یک سربار مختصر مراتب مختلف حفاظتی (فقط خواندنی، فقط اجرایی) میتواند به مؤلفه های مختلف نسبت داده شود.
    - اشتراک مؤلفه ها بین فرایندها

نیازمندی های مدیریت حافظه:

- سازمان فیزیکی:
  - جریان اطلاعات بین حافظه اصلی و ثانویه.
  - اگر مسئولیت به کاربر داده شود:
    - ممکن است حافظه موجود، برای یک برنامه و داده های آن کافی نباشد. در این صورت از روی هم گذاری استفاده میشود.
      - روی هم گذاری اجازه میدهد یک ناحیه از حافظه به چندین مؤلفه تخصیص یابد
    - در یک محیط چندبرنامگی، برنامه ساز در زمان نوشتن برنامه نمیداند چه مقدار از فضا در دسترس است و آن فضا کجاست.

بخش بندی ایستا:

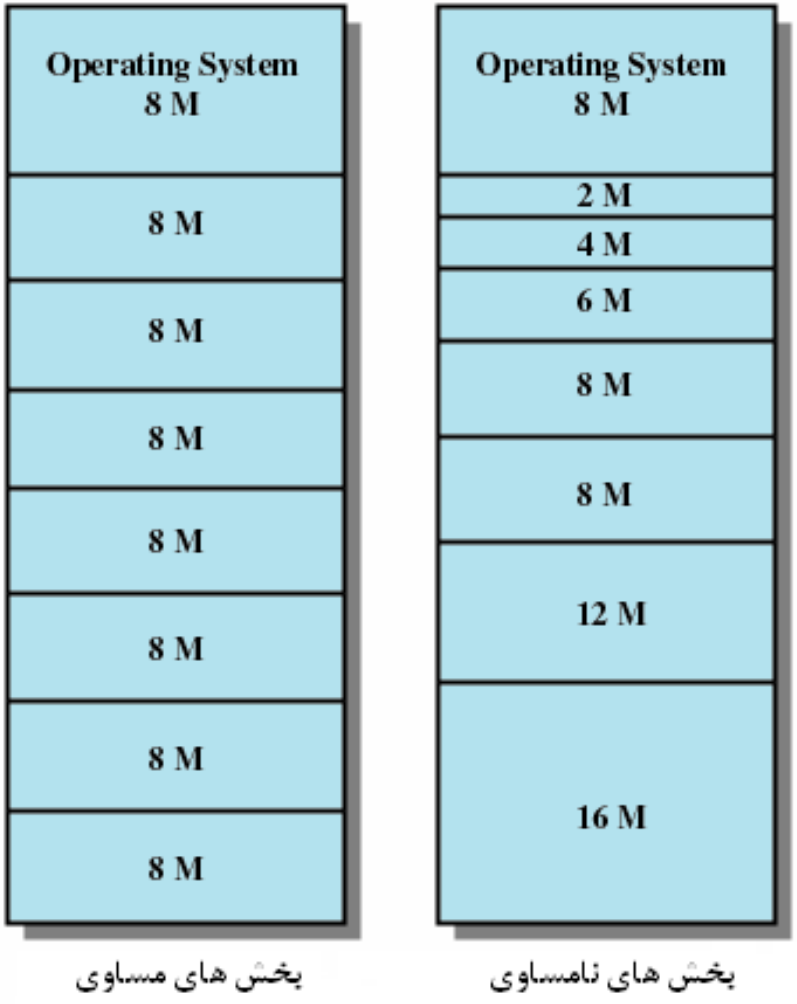
- در اغلب طرحهای مدیریت حافظه سیستم عامل، بخش ثابتی از حافظه را اشغال میکند و بقیه برای استفاده فرایندها است.
- در این روش بخش مربوط به فرایند ها به چندین بخش با طول ثابت تقسیم میشود.
- هر فرایندی که اندازه آن کمتر یا مساوی اندازه بخش باشد میتواند داخل هر بخش موجود بار شود.
- اگر همه بخش ها پر باشد، و هیچ فرایند مقیمی در حال آماده یا اجرا نباشد، سیستم عامل میتواند فرایند را به خارج مبادله کند.

بخش بندی ایستا:

- معایب بخش بندی طول ثابت:
  - ممکن است یک برنامه در یک بخش جا نشود، در این صورت برنامه نویس باید از روی هم گذاری استفاده کند.
  - استفاده ناکارمد از حافظه اصلی حافظه اصلی : هر برنامه صرف نظر از کوچکی آن یک بخش کامل را اشغال میکند به این پدیده پراکندگی داخلی میگویند.

بخش بندی ایستا:

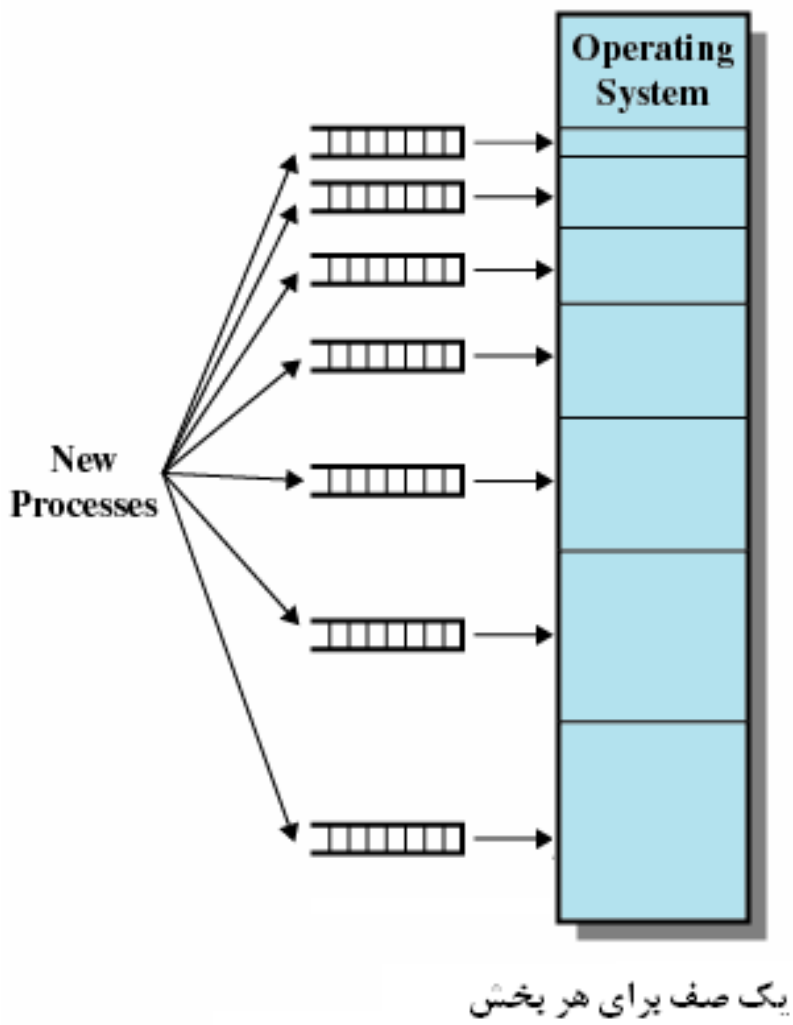
- بخش بندی طول ثابت بر اساس اندازه بخشها دو نوع است:
  - بخشهای مساوی
  - بخش های نامساوی



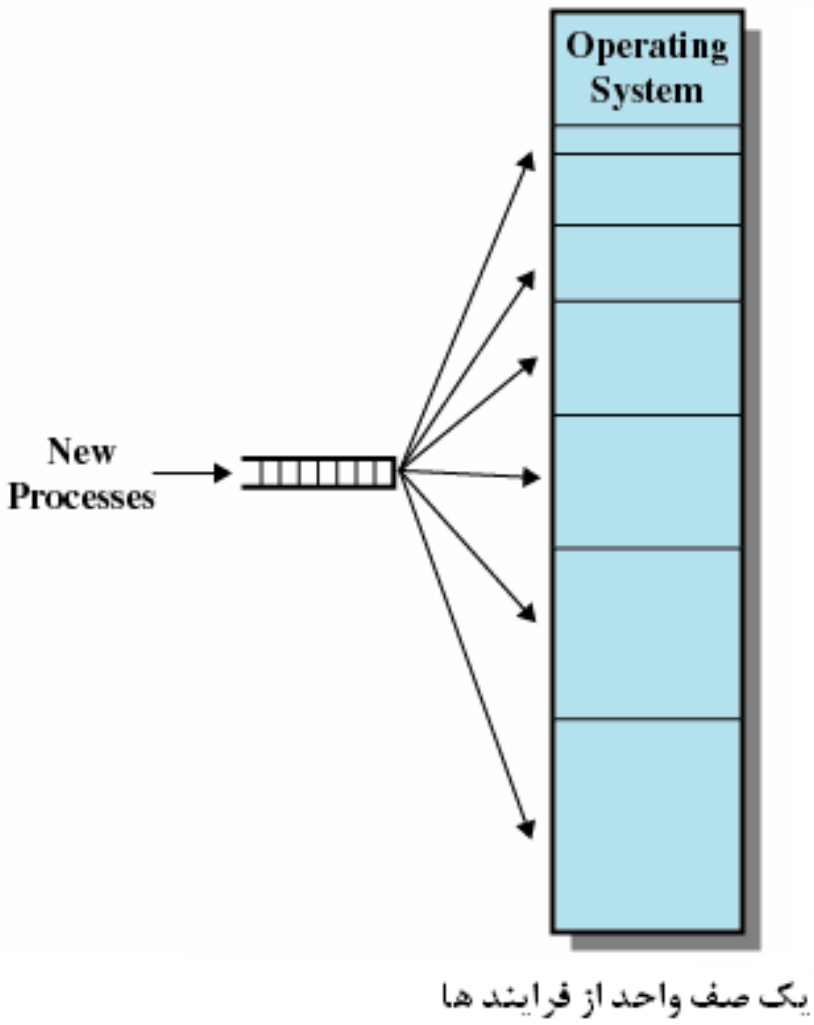
الگوریتم جاگذاری بخش بندی ایستا:

- بخشهای مساوی:
  - بدلیل مساوی بودن تمام بخشها، مهم نیست کدام بخش استفاده شود.
- بخش های نامساوی:
  - میتوان هر فرایند را در کوچکترین بخشی که در آن جا میشود قرار داد. در این صورت باید از صف زمانبندی استفاده شود.
    - برای هر بخش یک صف: حداقل هدر رفتن حافظه داخل هر بخش
    - یک صف برای همه بخشها:

تخصیص حافظه برای بخش بندی ایستا:

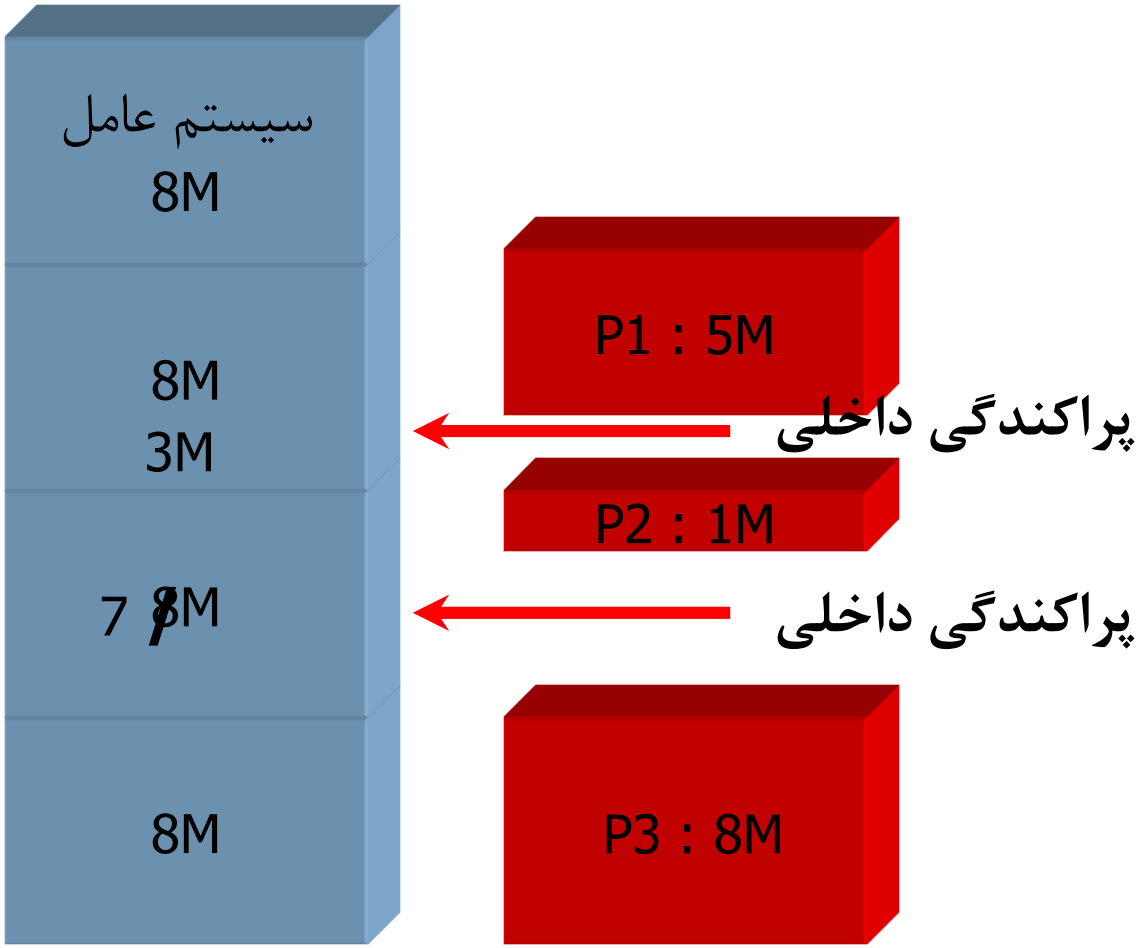


یک صف برای هر بخش



یک صف واحد از فرایند ها

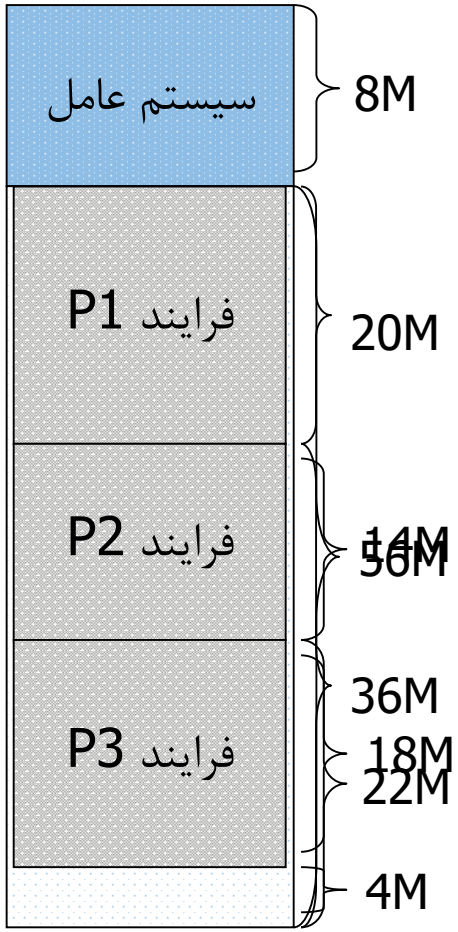
مثالی برای درک تکه تکه شدن داخلی:



بخش بندی پویا:

- بخشها دارای طول و تعداد متغیر هستند.
- حافظه تخصیص یافته به هر فرایند، دقیقاً برابر میزان نیاز فرایند است.
- پس از تخصیص و آزاد سازی های مکرر حفره هایی در حافظه پدید می آیند که پراکندگی خارجی نامیده میشوند.
- **فشرده سازی:** معمولا برای مقابله با پراکندگی خارجی، سیستم عامل فرایند ها را انتقال میدهد تا کنار یکدیگر قرار گیرند و تمام حافظه آزاد در یک بلوک قرار گیرد.

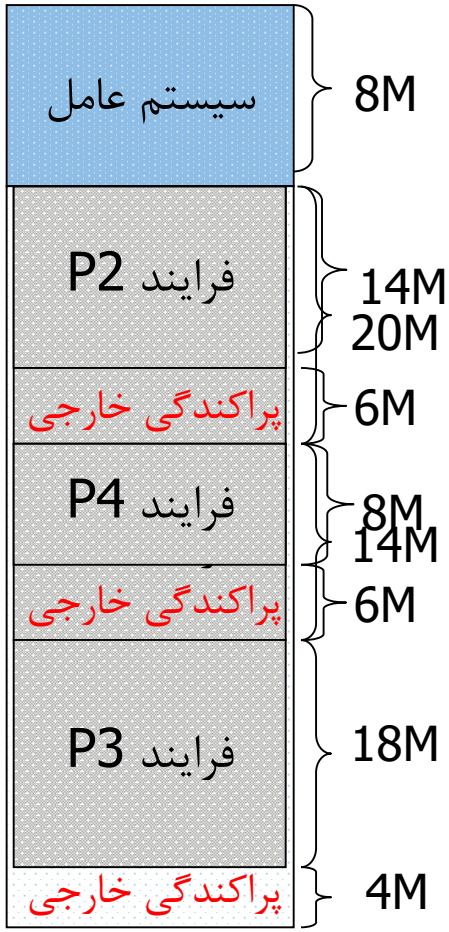
مثالی برای درک پراکندگی خارجی:



- ✓در ابتدا سیستم عامل 8M فضا اشغال میکند و 56Mفضای آزاد برای تخصیص به فرایند ها وجود دارد.
- ✓فرایند۱، 20M فضا اشغال میکند و 36M فضای آزاد باقی میماند.
- ✓فرایند۲، 14M فضا اشغال میکند و 22M فضای آزاد باقی میماند.
- ✓فرایند۳، 18M فضا اشغال میکند و 4M فضای آزاد باقی میماند.

Press Enter

مثالی برای درک پراکندگی خارجی:



✓فرایند۲،حذف میشود و 14M فضا آزاد میکند.

✓فرایند۴، 8M فضا اشغال میکند و در مکان قبلی فرایند۲ قرار میگیرد و 6M فضای آزاد باقی میماند.

✓فرایند۱، حذف میشود و 20M فضا آزاد میکند.

✓فرایند۲، 14M فضا اشغال میکند و در مکان قبلی فرایند۱ قرار میگیرد 6M فضای آزاد باقی میماند.

Press Enter

الگوریتم جاگذاری بخش بندی پویا:

- اگر چند بلوک آزاد وجود داشته باشد سیستم عامل باید تصمیم بگیرد کدام بلوک آزاد را تخصیص دهد.
- ۳ الگوریتم جایگذاری وجود دارد:
  - بهترین برازش
  - اولین برازش
  - در پی برازش

الگوریتم جاگذاری بخش بندی پویا:

- بهترین برازش:
  - کوچکترین بلوکی را که فرایند در آن جا میشود انتخاب میکند.
  - بیشترین هزینه اجرا و بدترین کارایی را دارد.
  - از آنجا که کوچکترین بلوک به فرایند اختصاص میابد، کمترین پراکندگی را بوجود می آورد.
  - اما معمولاً فضای باقیمانده ۱۰۰٪ پراکندگی است، چرا که خیلی کوچک است، بنابراین فشرده سازی بیشتر باید انجام شود.

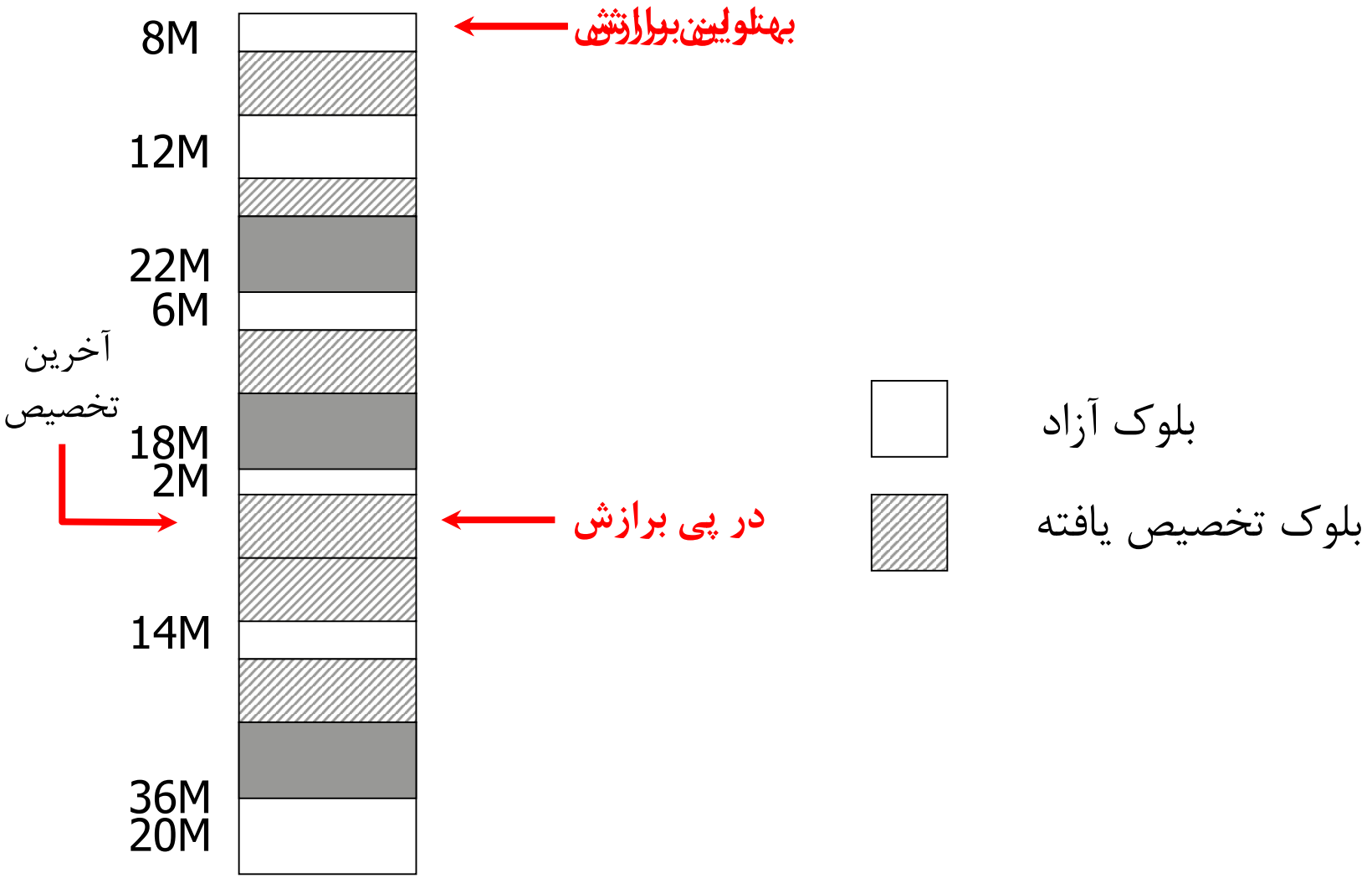
الگوریتم جاگذاری بخش بندی پویا:

- اولین برازش:
  - حافظه را از ابتدا مرور میکند و اولین بلوک آزاد و کافی را به فرایند اختصاص میدهد.
  - ساده ترین، سریعترین و بهترین الگوریتم است.
  - ممکن است قسمت ابتدایی حافظه را از تکه های کوچک انباشته کند که هر بار باید جستجو شوند.

الگوریتم جاگذاری بخش بندی پویا:

- در پی برازش:
  - حافظه را از محل آخرین جایابی به بعد مرور میکند. و اولین بلوک با اندازه کافی را انتخاب میکند.
  - معمولاً به تخصیص یک بلوک آزاد در انتهای حافظه (جایی که بلوک های بزرگتری پیدا میشوند) می انجامد.
  - بلوکهای بزرگ حافظه سریعاً به بلوکهای کوچکتر تقسیم میشوند.
  - فشرده سازی در این الگوریتم با بسامد بیشتری انجام میشود.

پیکر بندی حافظه قبل و بعد تخصیص یک بلو ۱۶ مگا بایتی



سیستم رفاقتی:

در یک سیستم رفاقتی بلوک های حافظه بصورت زیر هستند:

$$\text{Allocated Blook} = S = 2^K$$

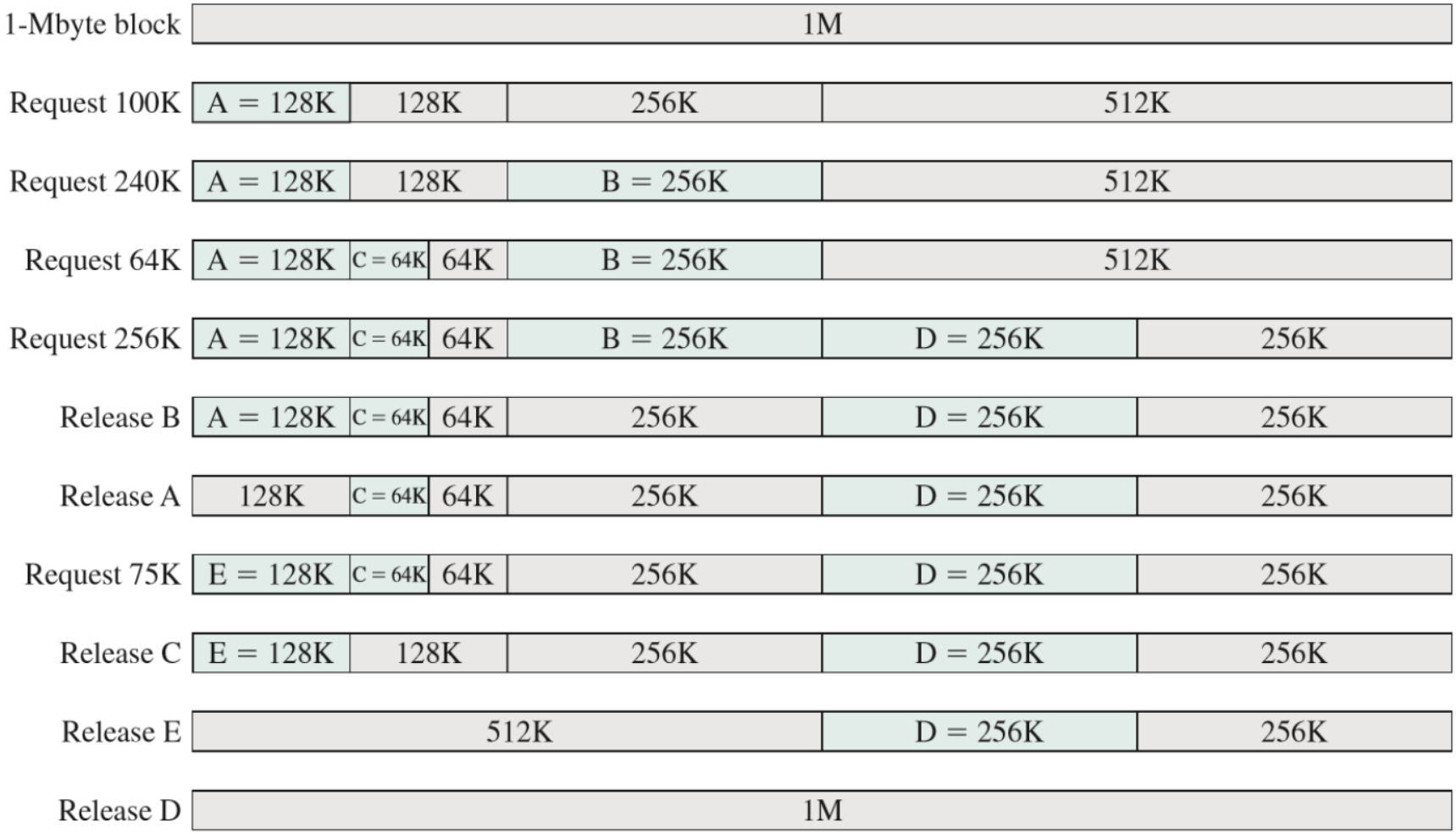
$$L \leq K \leq U$$

$$2^U = \text{Size Of Largest Blook}$$

$$2^L = \text{Size Of Smallest Blook}$$

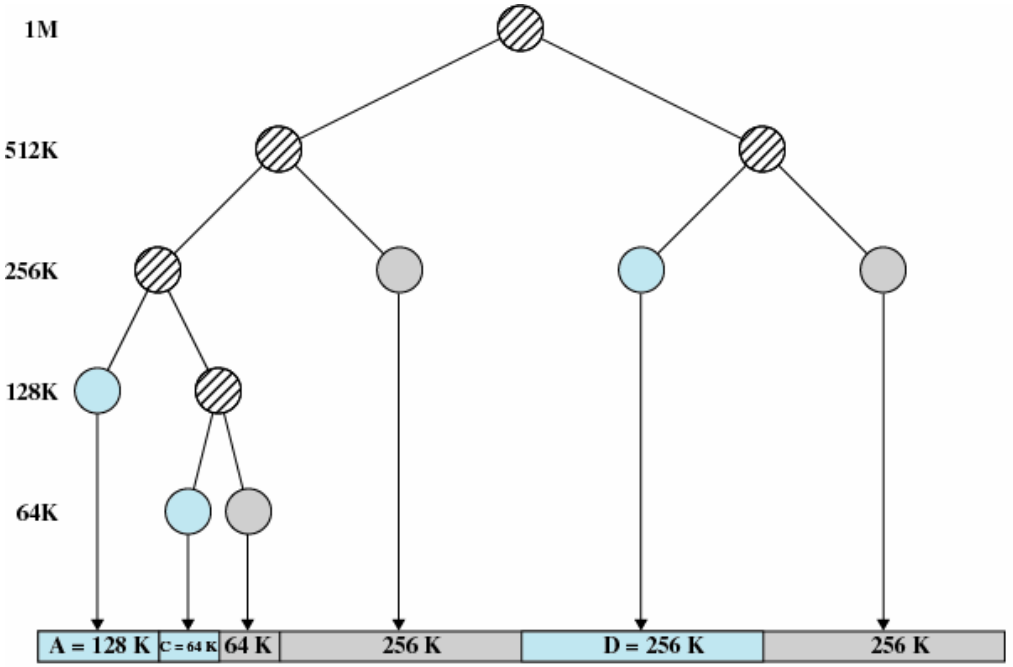
مراحل تقسیم تا زمانی که بلوکی با اندازه بزرگتر یا مساوی به فرایند تخصیص یابد تکرار میشود.

سیستم رفاقتی:



سیستم رفاقتی:

```
void get_hole(int i)
{
  if (i == (U + 1)) <failure>;
  if (<i_list empty>) {
      get_hole(i + 1);
      <split hole into buddies>;
      <put buddies on i_list>;
  }
  <take first hole on i_list>;
}
```



جابجایی:

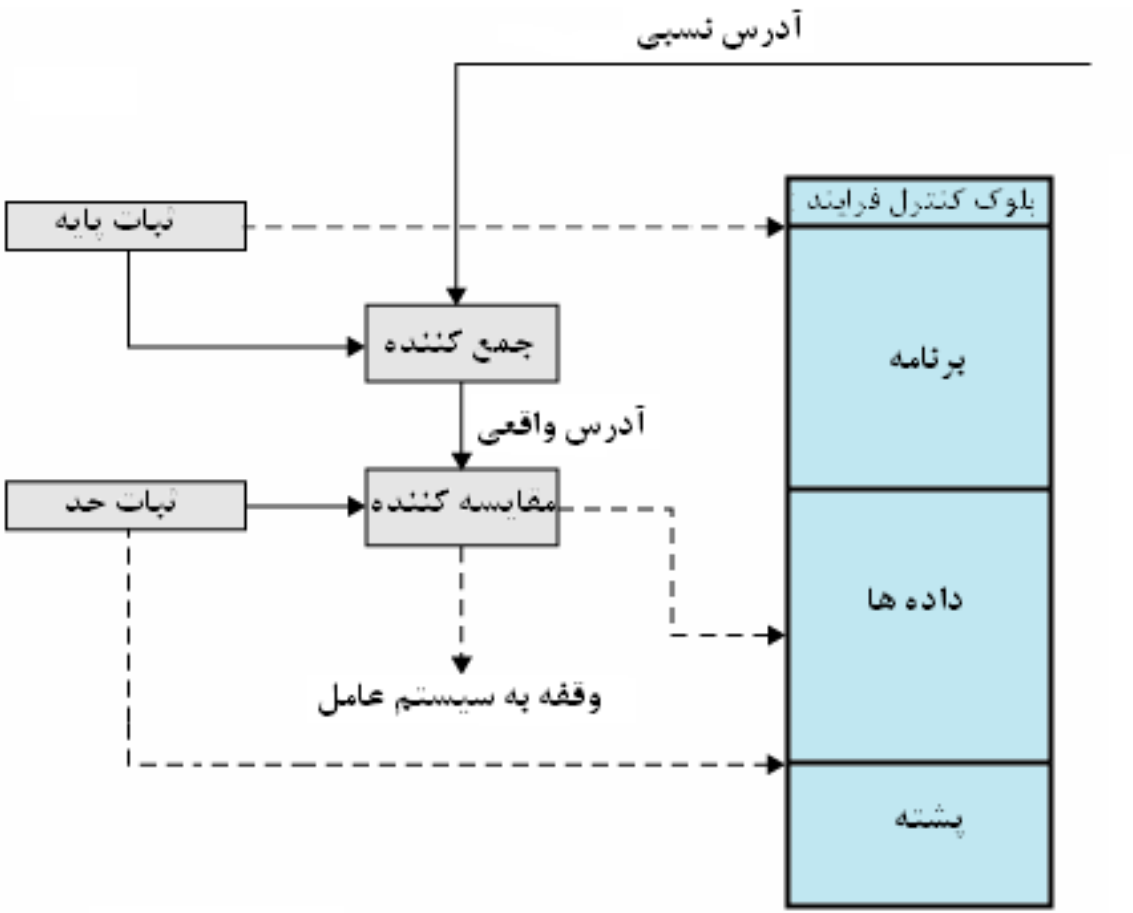
- زمانی که یک برنامه در حافظه بار میشود محلهای واقعی حافظه تخصیص یافته به برنامه تعیین میشوند.
- یک برنامه ممکن است در طول اجرا بخش های مختلفی از حافظه و بنابراین مکان های واقعی مختلفی از حافظه را اشغال کند.
- فشرده سازی موجب انتقال بخشهای فرایند میشود و این به معنای اشغال کردن محل های واقعی مختلفی از حافظه در طول اجراست.

انواع آدرس:

- منطقی:
  - یک مراجعه به حافظه مستقل از اختصاص داده جاری به حافظه است.
  - برای دسترسی به حافظه آدرس منطقی باید به آدرس فیزیکی ترجمه شود.
- نسبی:
  - آدرس به صورت مکان نسبی به یک نقطه معلوم (مثلاً ابتدای برنامه) بیان میشود.
- فیزیکی:
  - آدرس های وقعی یا مطلق در حافظه اصلی

حمایت سخت افزاری برای جابجایی:

- برای ترجمه آدرس های نسبی به آدرس های فیزیکی حافظه اصلی در زمان اجرای دستورالعمل حاوی مراجعه راهکار های سخت افزاری مورد نیاز است.



تصویر فرایند در حافظه اصلی

ثبات های بکار رفته حین اجرای دستورالعمل:

- ثبات پایه:
  - آدرس شروع فرایند
- ثبات حد:
  - آدرس پایان فرایند
- این ثباتها هنگام بار شدن فرایند به حافظه اصلی یا مبادله تصویر فرایند به داخل مقدار دهی میشوند.

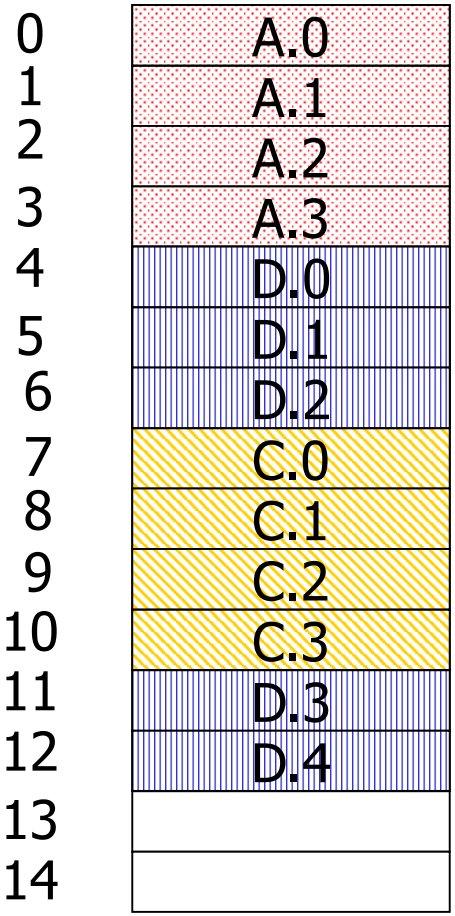
ثبات های بکار رفته حین اجرای دستورالعمل:

- مقدار ثبات پایه به آدرس نسبی افزوده میشود تا یک آدرس مطلق بدست آید.
- آدرس بدست آمده با مقدار ثبات حد مقایسه میشود.
- اگر در محدوده باشد، اجرای دستورالعمل ادامه میابد. در غیر این صورت خطایی به سیستم عامل داده میشود.

صفحه بندی:

- بخش های حافظه به قطعات کوچکی با اندازه ثابت تقسیم میشوند. برنامه ها نیز به قطعاتی با همان اندازه تقسیم میشوند.
- قطعات یک برنامه، صفحه نامیده میشوند و قطعات مربوط به حافظه قاب نام دارند.
- سیستم عامل برای هر فرایند یک جدول صفحه نگه می دارد.
  - محل قابی که هر صفحه از فرایند را در بر دارد، نگه می دارد.
  - هر آدرس منطقی شامل شماره صفحه و یک انحراف در صفحه است.

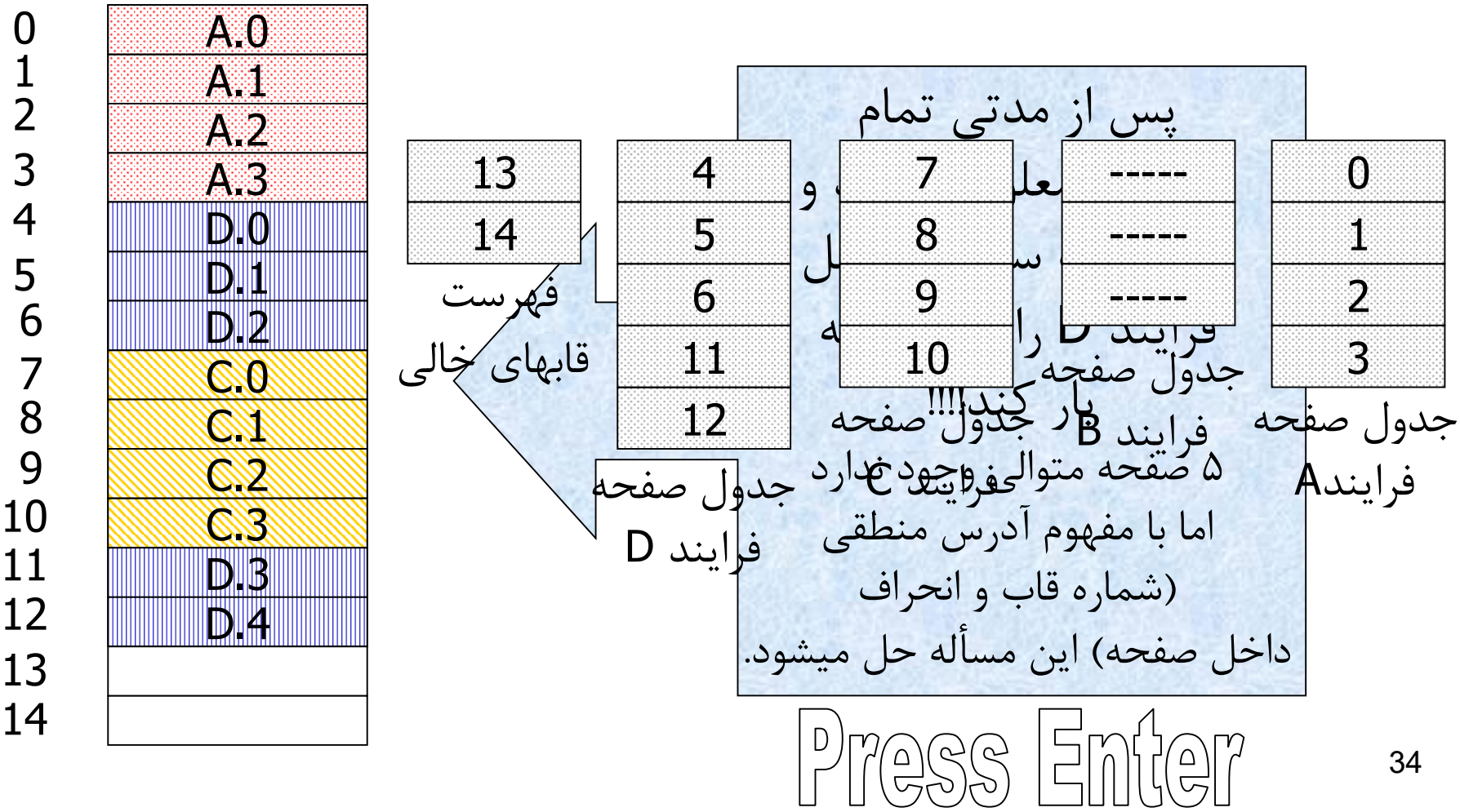
مثالی از اختصاص قاب های آزاد به فرایندها:



پس از مدتی تمام
فرایندها معلق میشوند و
نیاز است [illegible] عامل
فرایند D [illegible] ۵ صفحه
به دیسک انتقال میابد
۵ صفحه متوالی وجود ندارد
اما با مفهوم آدرس منطقی
(شماره قاب و انحراف
داخل صفحه) این مسأله حل میشود.

Press Enter

مثالی از اختصاص قاب های آزاد به فرایندها:



ترجمه آدرس در صفحه‌بندی:

1. استخراج شماره صفحه به عنوان n بیت سمت چپ آدرس منطقی
2. استفاده از شماره صفحه به عنوان شاخص به جدول صفحه فرآیند برای یافتن شماره قاب (k)
3. آدرس فیزیکی شروع قاب $K*2^M$ و آدرس فیزیکی بایت مورد نظر همین عدد به علاوه انحراف است

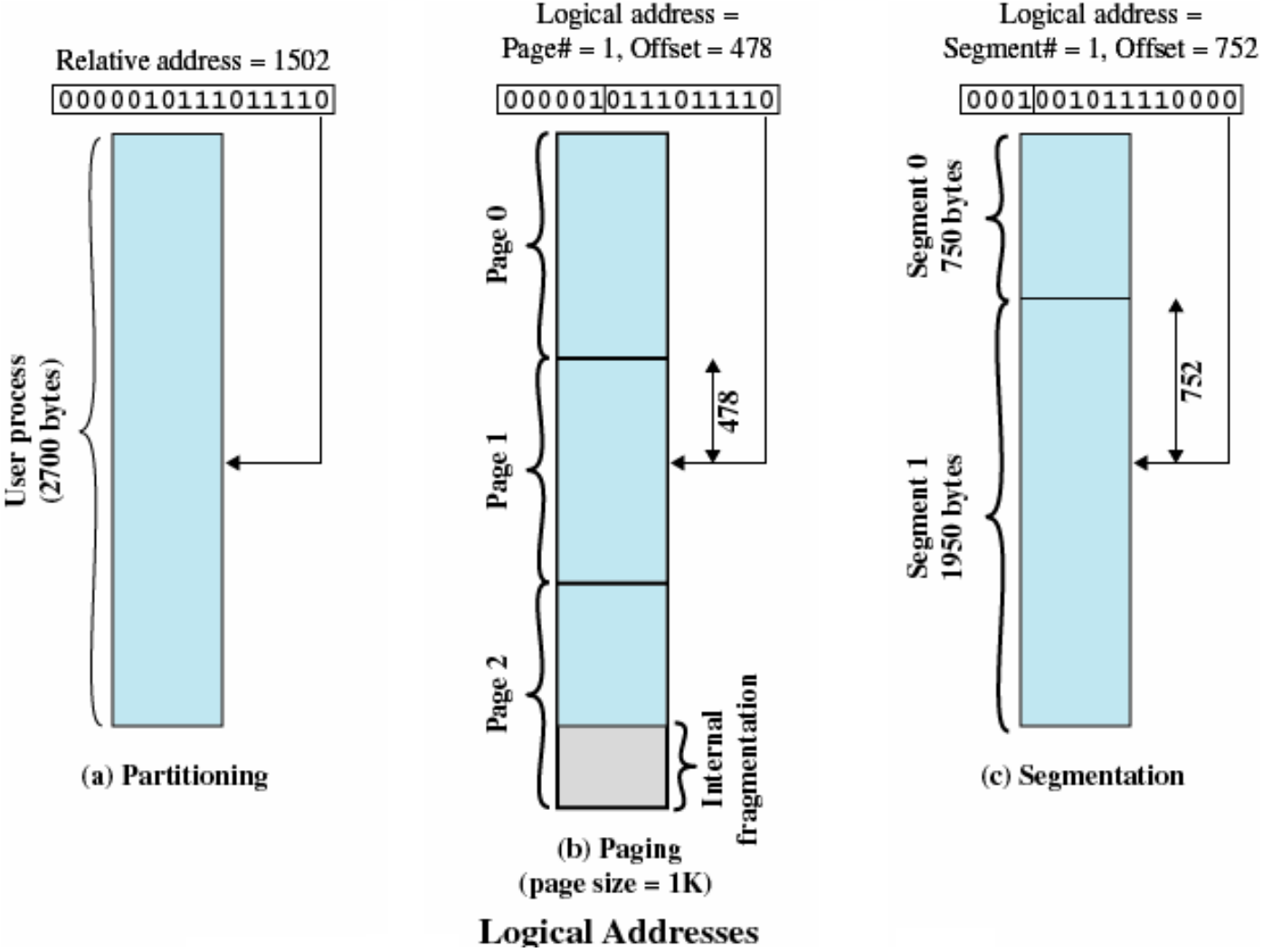
قطعه بندی:

- برنامه و داده‌های مربوط به تعدادی قطعه تقسیم می‌شوند.
- لزومی ندارد همه قطعات تمام برنامه ها دارای اندازه یکسان باشند
- حداکثری برای طول قطعه وجود دارد.
- آدرس منطقی شامل دو بخش است: شماره قطعه و انحراف داخل قطعه
- ازآنجایی که قطعه ها دارای طول متغیراند قطعه بندی مانند بخشبندی پویا است.
- قطعه بندی تکه تکه شدن داخلی را حذف می‌کند ولی همانند بخشبندی پویا دارای اشکال تکه تکه شدن خارجی است.

ترجمه آدرس در قطعه بندی:

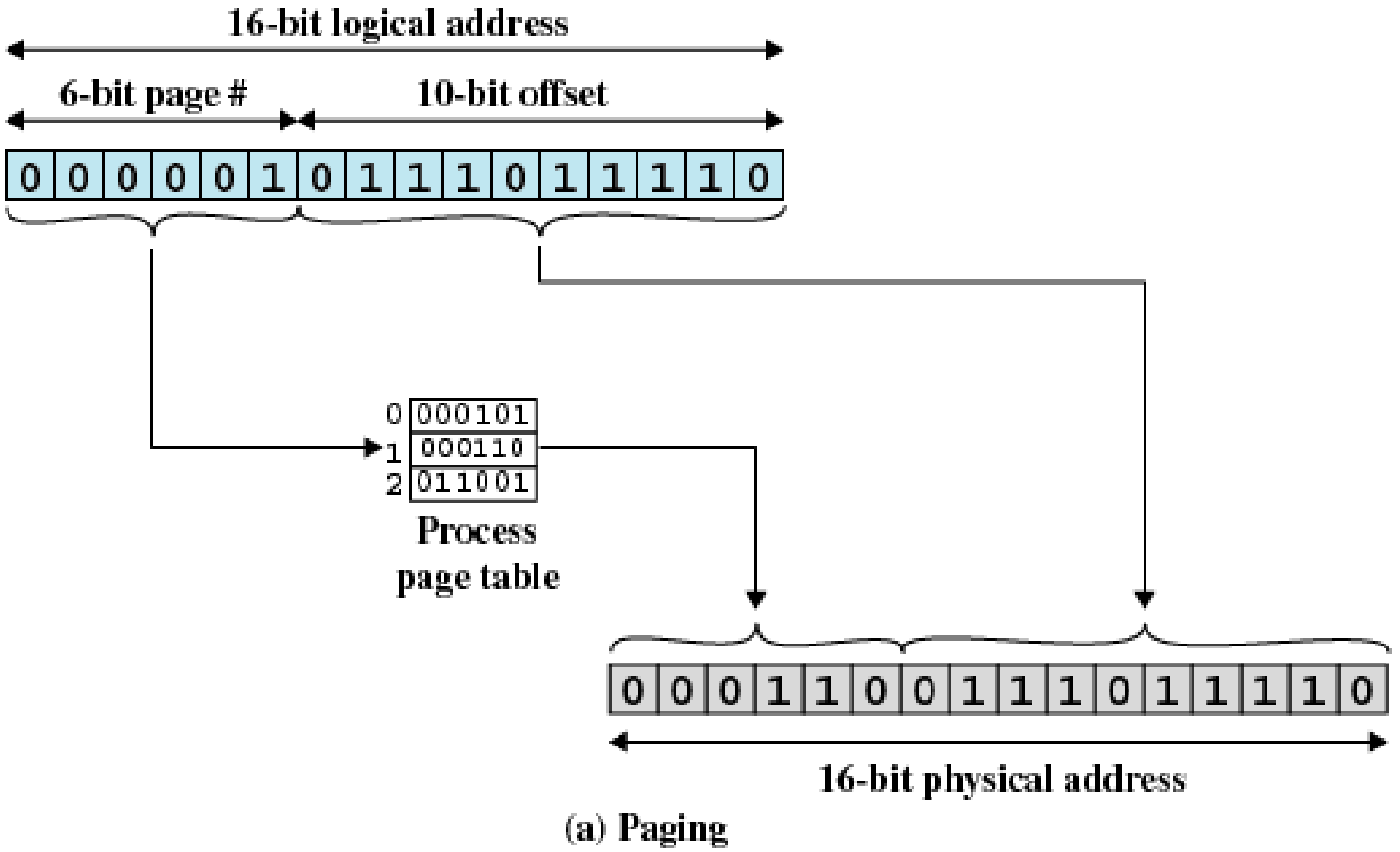
1. استخراج شماره قطعه از n بیت سمت چپ آدرس منطقی
2. استفاده از شماره قطعه به عنوان شاخص به جدول قطعه فرآیند برای یافتن آدرس فیزیکی شروع آن قطعه
3. مقایسه انحراف موجود در m بیت سمت راست با طول قطعه، اگر انحراف بزرگتر از طول قطعه باشد، آدرس معبر نیست.
4. آدرس فیزیکی مورد نظر عبارت است از **مجموع** آدرس فیزیکی شروع قطعه و انحراف.

آدرسهای منطقی :



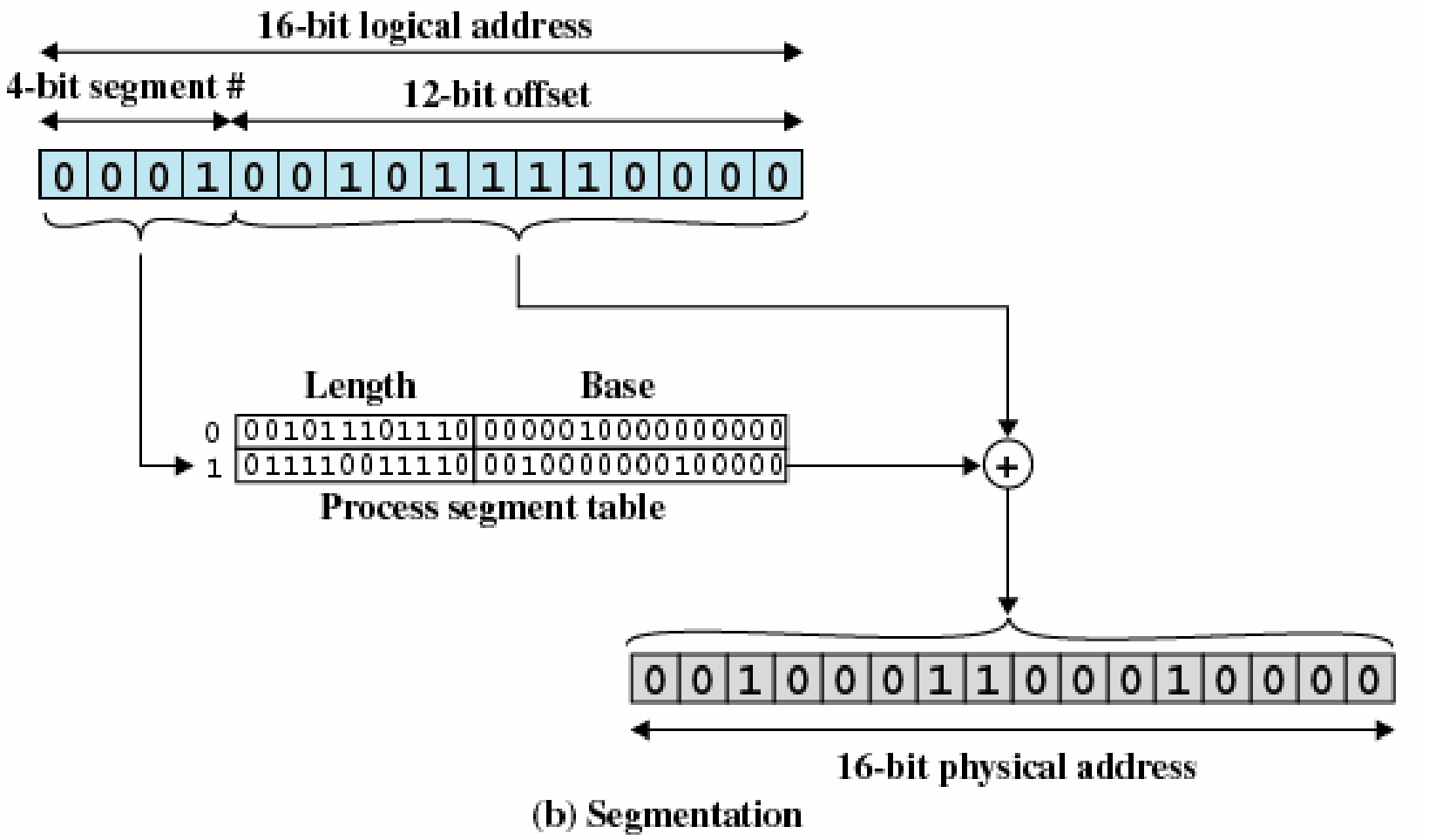
Logical Addresses

ترجمه آدرس منطقی به فیزیکی در صفحه بندی:



(a) Paging

ترجمه آدرس منطقی به فیزیکی در قطعه بندی:



(b) Segmentation