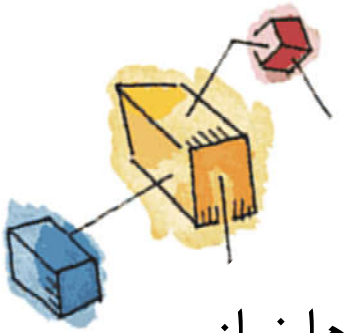
به نام خدا

فصل هفتم:
مدیریت حافظه

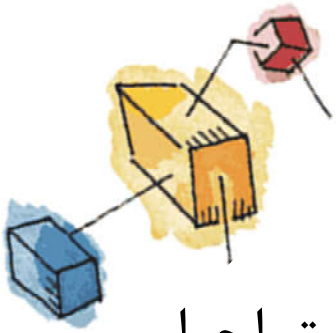
Memory Management



مدیریت حافظه

- امروزه قیمت حافظه بسیار پایین آمده است اما متقابلا برنامه ها نیاز روزافزونی به حافظه دارند و بنابراین، همچنان نیاز به مدیریت کارآمد حافظه است.

- مدیریت حافظه، شامل جابجا کردن بلوک های داده ها و برنامه ها بین حافظه جانبی و حافظه اصلی است.
- این جابجایی در مقایسه با سرعت پردازنده، کند است.
- بنابراین، نیاز است که پردازنده به صورت هوشمندانه ای این جابجایی را برنامه ریزی کند تا کارایی پردازنده را افزایش دهد.

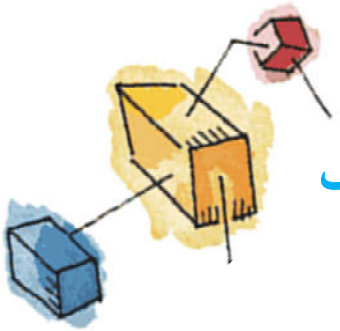


مدیریت حافظه

- وظیفه اصلی مدیریت حافظه انتقال برنامه به داخل حافظه جهت اجرا توسط پردازنده است.

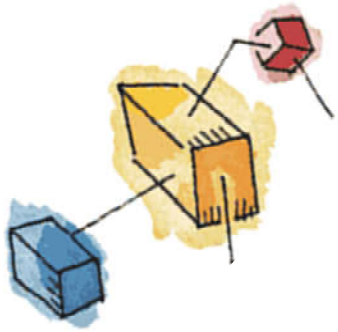
- تقسیم بندی حافظه به گونه ای صورت می گیرد که چندین فرآیند را در خود جای دهد.

- بایستی حافظه به نحوی به فرآیندها تخصیص یابد که تعداد مناسبی از فرآیندهای آماده برای استفاده از وقت پردازنده در حافظه موجود باشند.



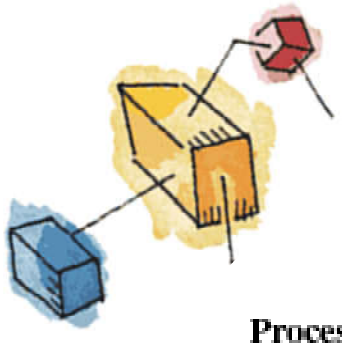
نیازهایی که مدیریت حافظه باید پاسخگو باشد

- جابجایی
- حفاظت
- اشتراک
- سازمان منطقی
- سازمان فیزیکی

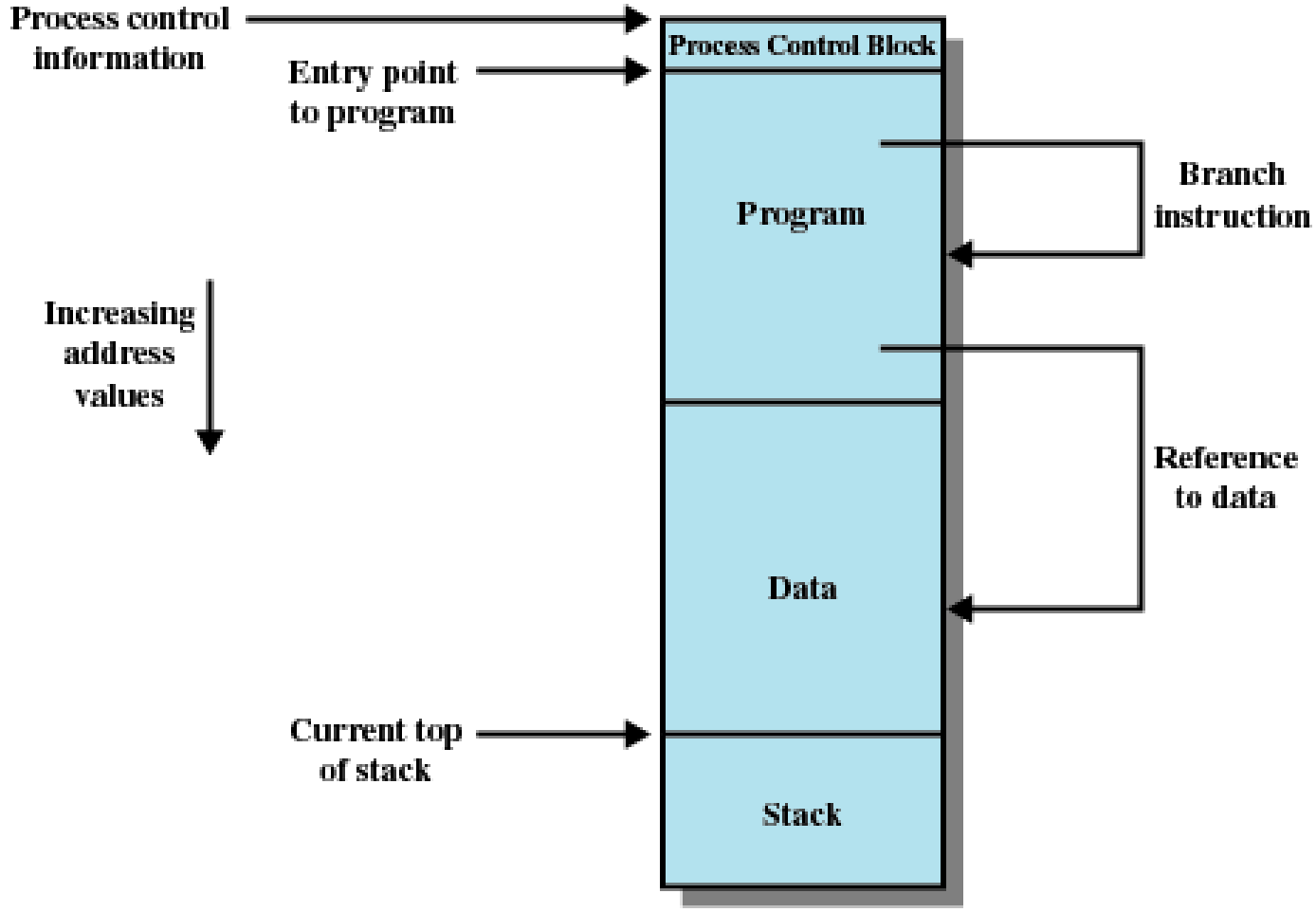


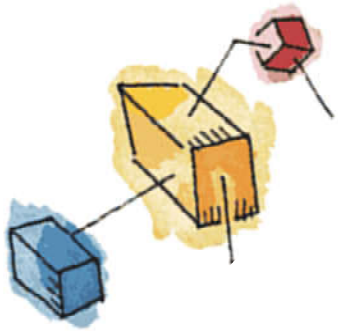
نیازمندی های مدیریت حافظه

- **جابجایی (Relocation)**
  - برنامه نویس نمی داند برنامه هنگام اجرا در کجای حافظه ذخیره می شود.
  - هنگام اجرای برنامه ممکن است برنامه به دیسک منتقل شود و دوباره در مکان دیگری از حافظه قرار گیرد.
  - ارجاع های به حافظه باید به آدرس های فیزیکی ای که منعکس کننده مکان فعلی و حقیقی برنامه در حافظه اند ترجمه شوند.



نیازهای آدرس دهی برای یک فرآیند

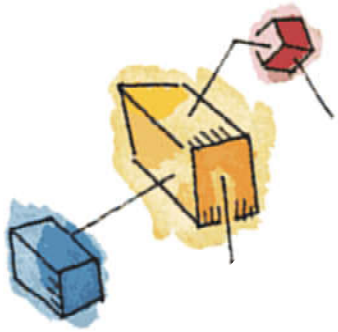




نیازمندی های مدیریت حافظه

- **حفاظت (Protection)**

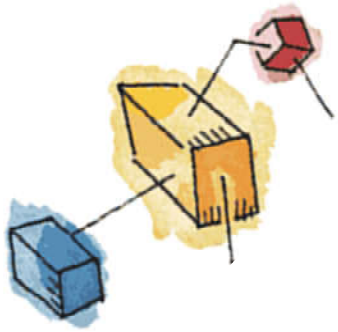
  - فرآیندها نباید بدون اجازه قادر به مراجعه به فضای حافظه مربوط به سایر فرآیندها باشند.
  - یک فرآیند کاربر نباید بتواند به هیچ بخش سیستم عامل اعم از برنامه یا داده دسترسی داشته باشد.
  - بررسی آدرس های فیزیکی در زمان کامپایل (ترجمه) غیرممکن بوده و بایستی در زمان اجرا مورد بررسی قرار گیرند.
  - نیازهای حفاظتی بایستی توسط پردازنده (سخت افزار) برآورده شود نه توسط سیستم عامل (نرم افزار).



نیازمندی های مدیریت حافظه

- **اشتراک (Sharing)**

  - اگر چند فرآیند در حال اجرای یک برنامه هستند، به جای اینکه هر فرآیند کپی جداگانه ای از برنامه داشته باشد بهتر است همه فرآیندها به یک کپی از برنامه دسترسی داشته باشند.
  - هر راهکار حفاظتی باید انعطاف پذیری لازم را داشته باشد و امکان دسترسی چندین فرآیند به یک بخش یکسان از حافظه را بدهد.



نیازمندی های مدیریت حافظه

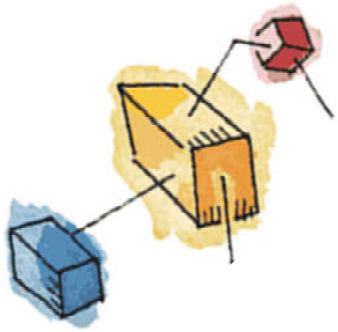
- **سازمان منطقی** (Logical Organization)

– اکثر برنامه ها به صورت مولفه ای سازمان یافته اند.

– مزایای کار کردن با مولفه ها:

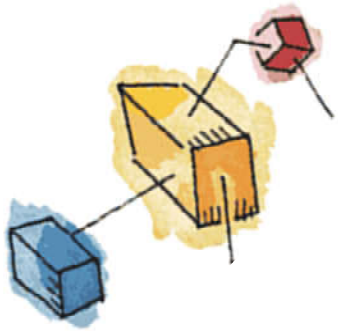
- هر مولفه را می توان به طور مستقل نوشت و ترجمه کرد.
- درجات مختلف حفاظتی (فقط خواندنی، فقط اجرایی) می تواند به مؤلفه های مختلف نسبت داده شود.
- می توان مؤلفه ها را بین فرآیندها به اشتراک گذاشت.

– اگر سیستم عامل و سخت افزار بتوانند به طور موثری با برنامه ها و داده های کاربر به شکل مولفه ای کار کنند، این مزایا برآورده می شود.



نیازمندی های مدیریت حافظه

- **سازمان فیزیکی (Physical Organization)**
  - ممکن است حافظه موجود، برای یک برنامه و داده های آن کافی نباشد.
  - در این شرایط باید جریان اطلاعات بین حافظه اصلی و ثانویه صورت بگیرد و این امر، دغدغه اصلی یک سیستم است.
  - اگر مسئولیت این فرآیند بر عهده برنامه ساز باشد، غیرعملی و نامطلوب است زیرا:
    - در این صورت برنامه ساز باید درگیر روشی به نام جایگذاشت یا روی هم گذاری (Overlaying) شود. جایگذاشت اجازه می دهد یک ناحیه از حافظه به چندین مولفه تخصیص یابد و دائما تعویض صورت بگیرد. این کار برای برنامه ساز، زمان بر است.
    - در یک محیط چندبرنامگی، برنامه ساز در زمان نوشتن برنامه نمی داند چه مقدار و چه محل از فضای حافظه در دسترس خواهد بود.
  - به همین دلیل، این کار به عنوان یک وظیفه سیستمی درنظر گرفته می شود.

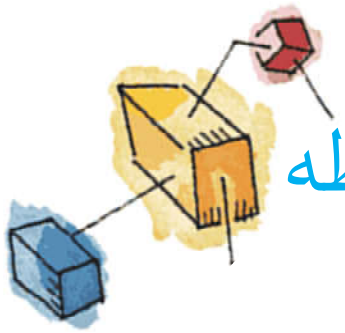


بخش بندی حافظه

- روش های مختلفی برای بخش بندی حافظه ارائه شده اند.
  1. بخش بندی ایستا
  2. بخش بندی پویا
  3. سیستم رفاقتی (Buddy System)
  4. صفحه بندی ساده (Paging)
  5. قطعه بندی ساده (Segmentation)
  6. صفحه بندی حافظه مجازی
  7. قطعه بندی حافظه مجازی

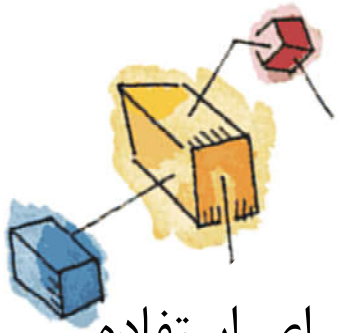
فصل هشتم

- روش های ساده برای بخش بندی حافظه امروزه به کار نمی روند ولی بررسی آنها، به درک روش های جدیدتر کمک می کند.



روش های مختلف برای بخش بندی حافظه

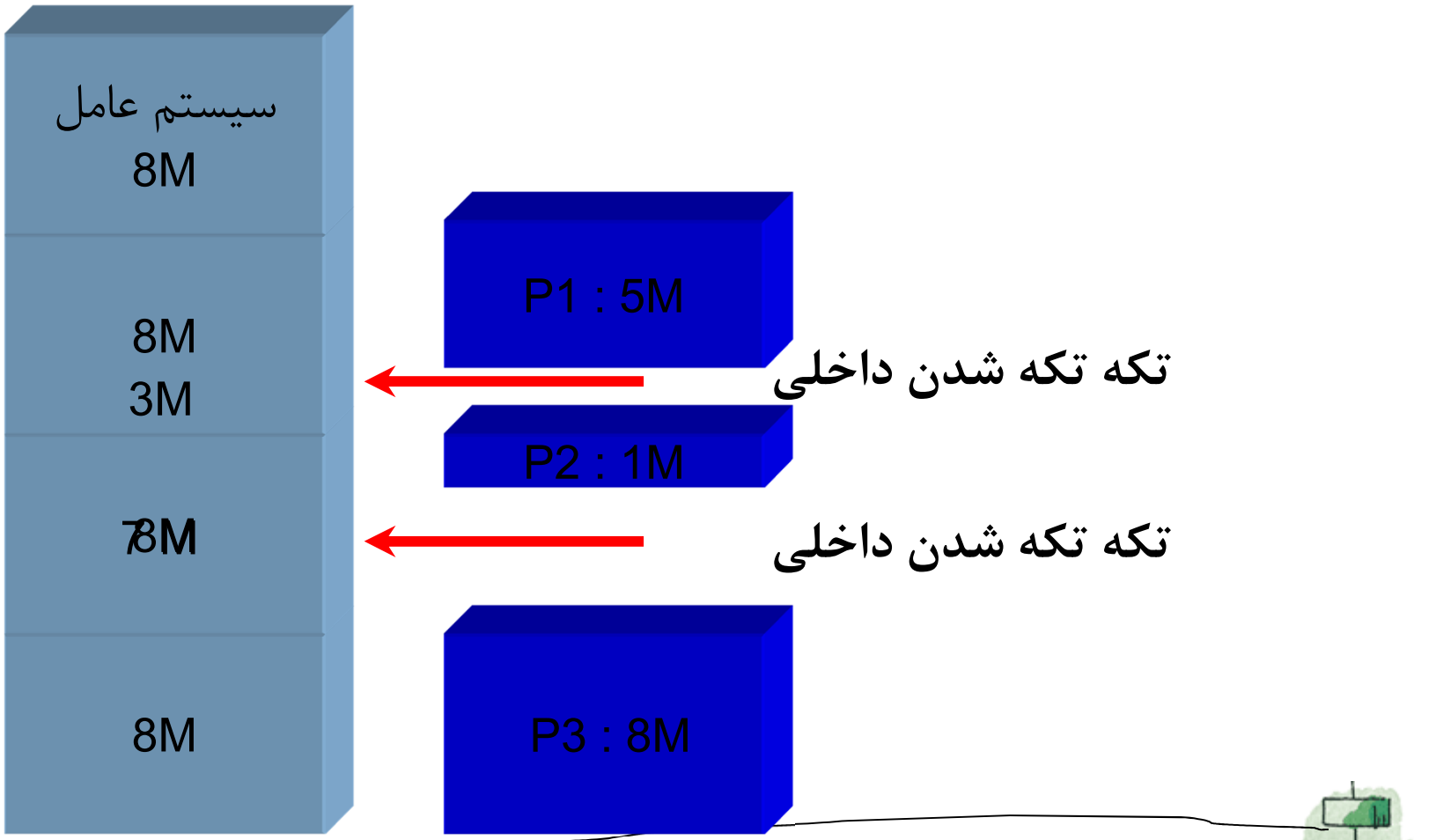
1. **بخش بندی ایستا**
2. بخش بندی پویا
3. سیستم رفاقتی (Buddy System)
4. صفحه بندی ساده (Paging)
5. قطعه بندی ساده (Segmentation)

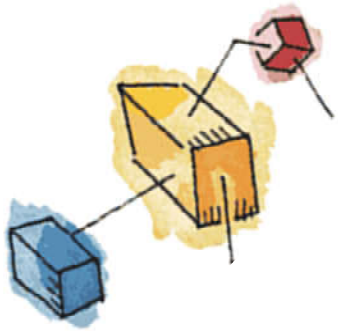


بخش بندی ایستا (Fixed Partitioning)

- سیستم عامل بخش ثابتی از حافظه را اشغال نموده و مابقی حافظه برای استفاده فرآیندها بخش بندی می شود (در همه روش های بخش بندی این فرض درنظر گرفته شده است).
- در روش بخش بندی ایستا، بخش مربوط به فرآیندها به چندین بخش با طول ثابت تقسیم می شود.
- هر فرآیندی که اندازه آن کمتر یا مساوی اندازه بخش باشد می تواند داخل هر بخش موجود بار شود.
- اگر همه بخش ها پر باشد، و هیچ فرآیند مقیمی در حال آماده یا اجرا نباشد، سیستم عامل می تواند فرآیندی را به خارج از حافظه مبادله کند.

مثالی برای درک تکه تکه شدن داخلی





بخش بندی ایستا

- معایب بخش بندی با طول ثابت:

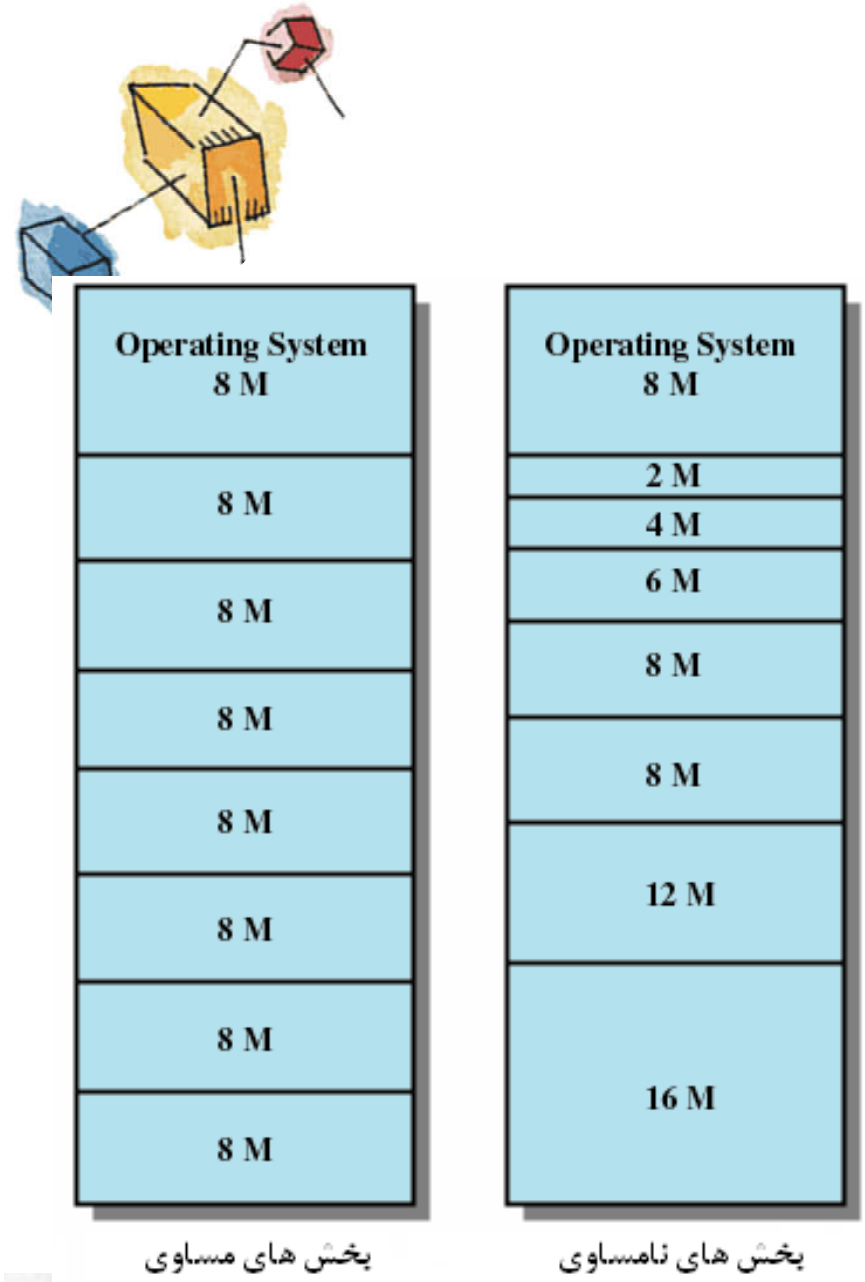
– ممکن است یک برنامه در یک بخش جا نشود، در این صورت برنامه نویس بایستی برنامه را با قابلیت روی هم گذاری (**Overlaying**) طراحی کند.

– استفاده ناکارآمد از حافظه اصلی

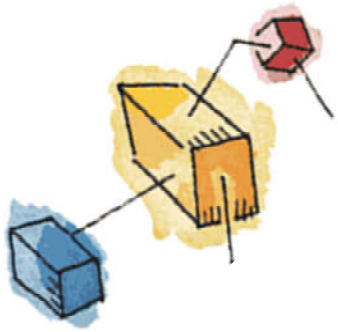
- هر برنامه، هرچقدر هم که کوچک باشد، یک بخش کامل را اشغال می کند. به این پدیده **تکه تکه شدن داخلی** (**Internal Fragmentation**) می گویند.

– تعداد فرآیندهای فعال در سیستم به تعداد بخش های تعریف شده در زمان ایجاد سیستم محدود می گردد.

بخش بندی ایستا



- یک راه حل، استفاده از بخش بندی نامساوی است که دو مساله اول را تا حدی بهبود می دهد ولی به طور کامل حل نمی کند.
- در واقع می توان گفت بخش بندی ایستا به دو صورت قابل انجام است:
  - بخش های مساوی
  - بخش های نامساوی

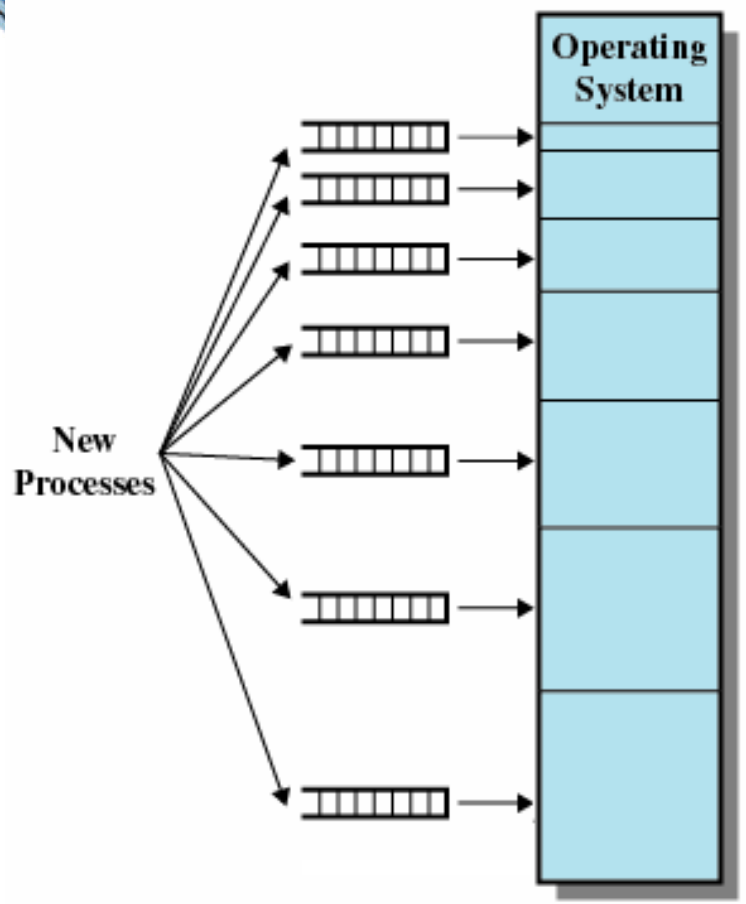


الگوریتم جایگذاری بخش بندی ایستا

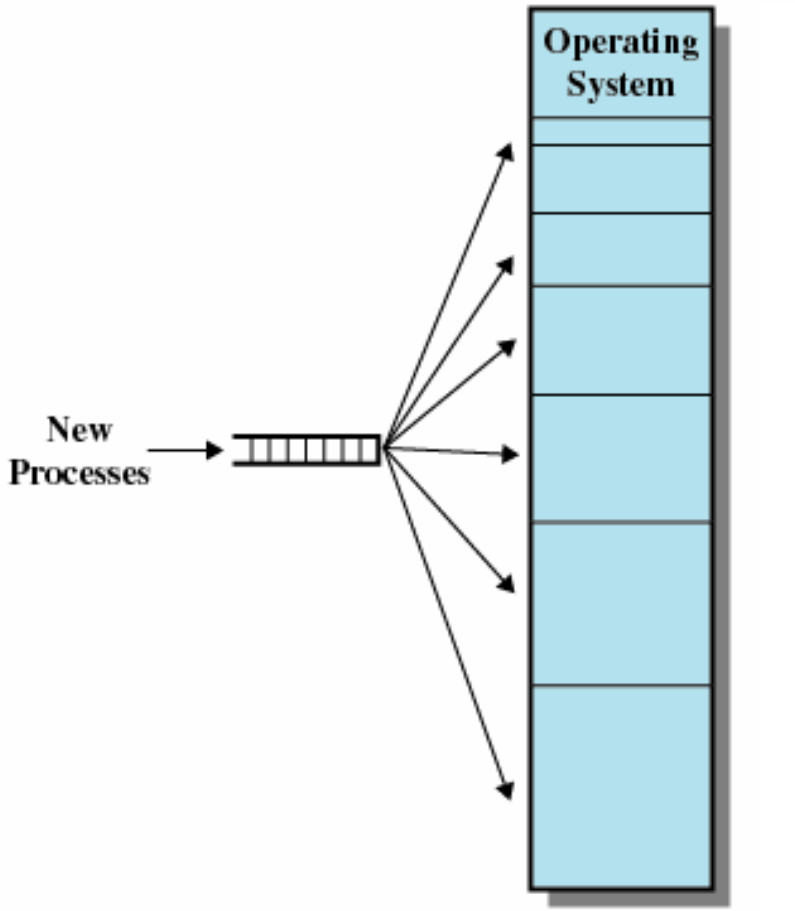
- بخش های مساوی
  - بدلیل مساوی بودن تمام بخش ها، مهم نیست کدام بخش استفاده شود.

- بخش های نامساوی
  - می توان هر فرآیند را در کوچکترین بخشی که در آن جا می شود قرار داد.
    - برای هر بخش یک صف
  - تخصیص حافظه به فرآیندها به نحوی صورت می گیرد که حداقل هدر رفت حافظه داخل هر بخش (تکه تکه شدن داخلی) را به دنبال داشته باشد. ولی در مجموع از دیدگاه سیستم بهینه نیست.
    - یک صف برای همه بخش ها

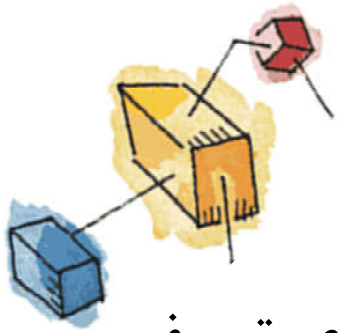
تخصیص حافظه برای بخش بندی ایستا



یک صف برای هر بخش



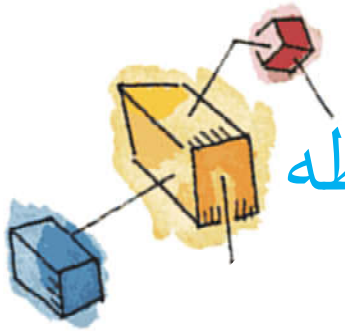
یک صف واحد از فرایند ها



مشکلات باقیمانده از بخش بندی ایستا

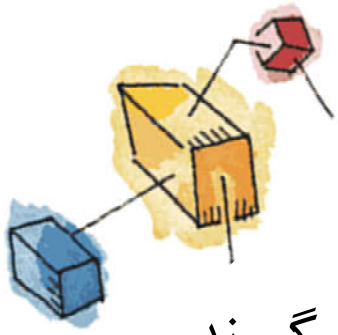
- تعداد فرآیندهای فعال سیستم محدود است به تعداد بخش های تعریف شده در زمان ایجاد سیستم.

- از آنجا که اندازه بخش ها در زمان ایجاد سیستم انتخاب شده اند، کارهای کوچک نمی توانند به صورت کارا از فضای بخش ها استفاده کنند.
  - ممکن است در محیطی که نیازهای برنامه ها برای حافظه اصلی از قبل معلوم است، این مطلب منطقی باشد، ولی در اغلب موارد، روش بخش بندی ایستا ناکارآمد است.



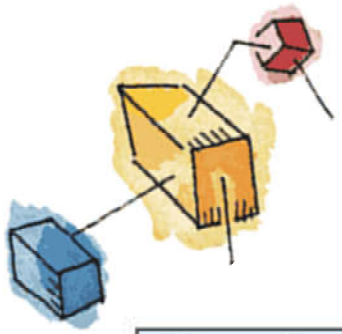
روش های مختلف برای بخش بندی حافظه

1. بخش بندی ایستا
2. **بخش بندی پویا**
3. سیستم رفاقتی (Buddy System)
4. صفحه بندی ساده (Paging)
5. قطعه بندی ساده (Segmentation)

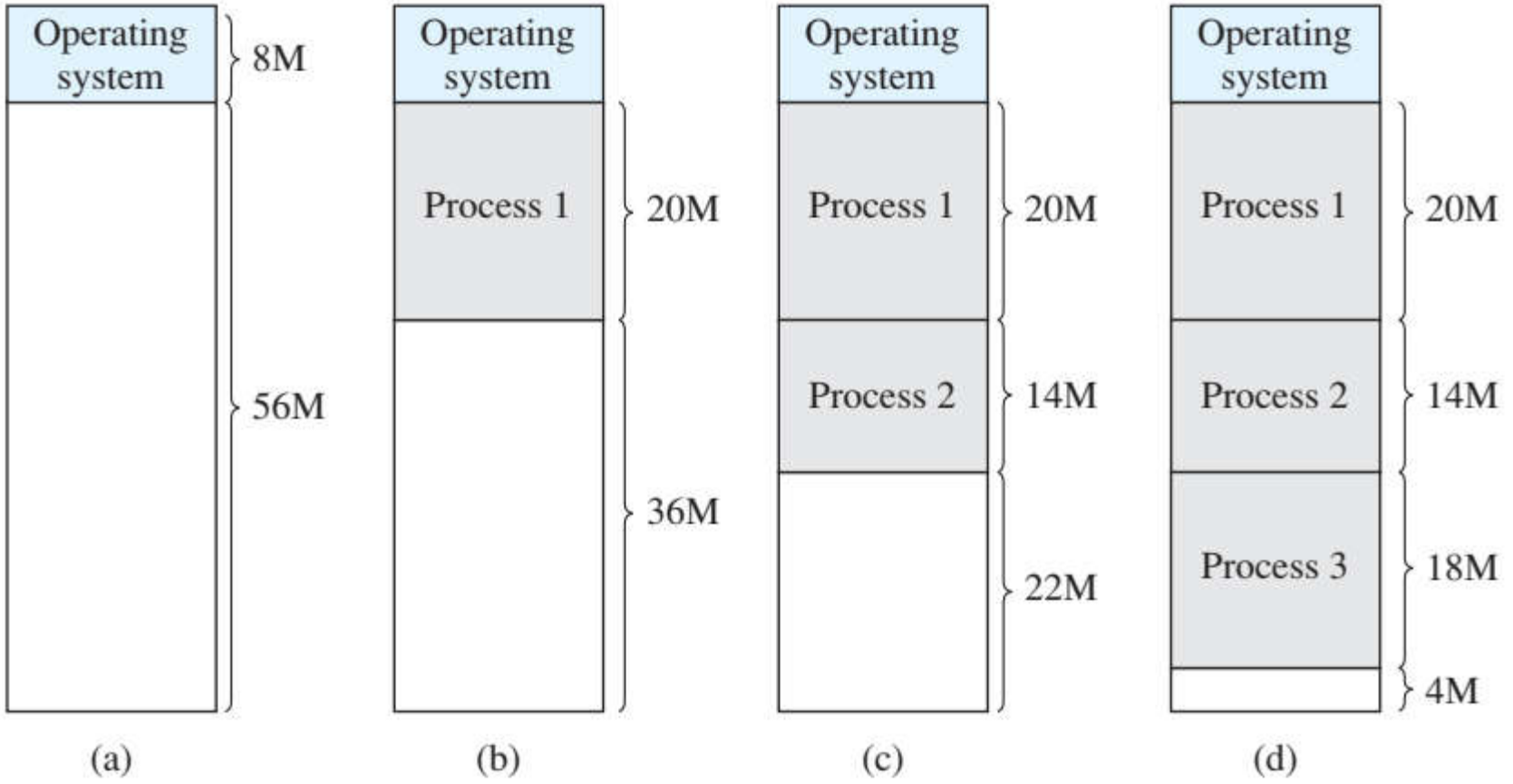


بخش بندی پویا (Dynamic Partitioning)

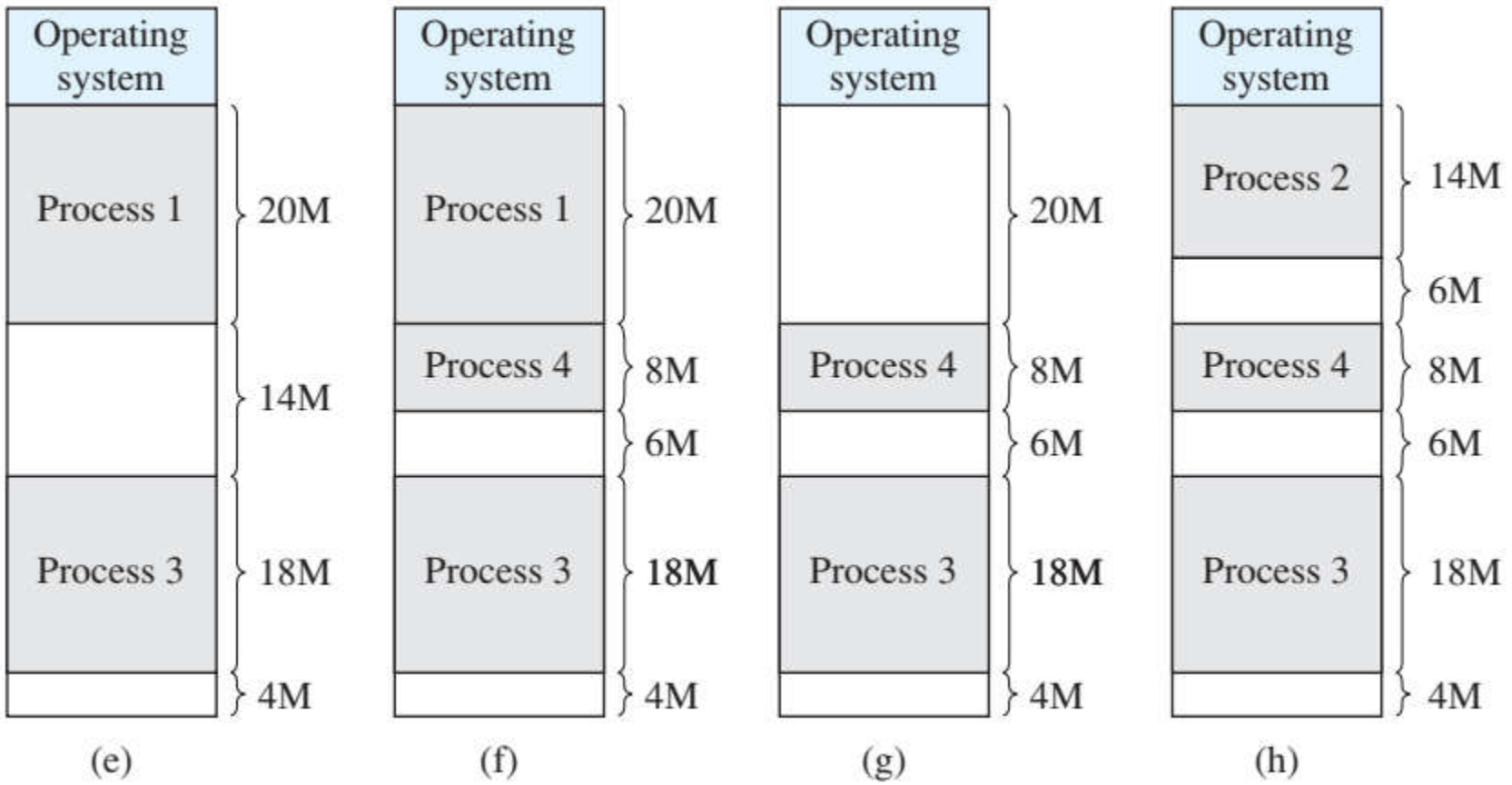
- بخش ها دارای طول و تعداد متغیر هستند و کم کم شکل می گیرند.
- حافظه تخصیص یافته به هر فرآیند، دقیقا برابر میزان نیاز فرآیند است.
- پس از تخصیص و آزادسازی های مکرر، حفره هایی در حافظه پدید می آیند.
  این پدیده **تکه تکه شدن خارجی** (External Fragmentation) نامیده می شود.
- معمولا برای مقابله با تکه تکه شدن خارجی، سیستم عامل فرآیندها را انتقال می دهد تا کنار یکدیگر قرار گرفته و تمام حافظه آزاد در یک بلاک قرار گیرد. به این کار فشرده سازی (compaction) گفته می شود.

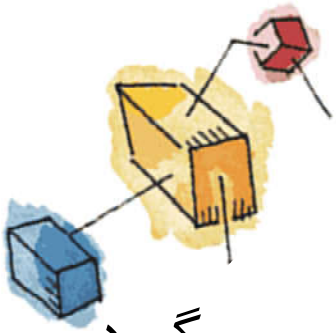


مثالی برای درک تکه تکه شدن خارجی



مثالی برای درک تکه تکه شدن خارجی (ادامه)

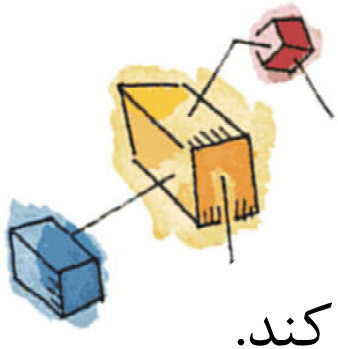




الگوریتم های جایگذاری بخش بندی پویا

- اگر چند بلاک آزاد وجود داشته باشد، سیستم عامل باید تصمیم بگیرد کدام بلاک آزاد را تخصیص دهد.

- برای بخش بندی پویا سه الگوریتم جایگذاری وجود دارد:
  - الگوریتم بهترین برازش (Best-fit)
  - الگوریتم اولین برازش (First-fit)
  - الگوریتم در پی برازش (Next-fit)

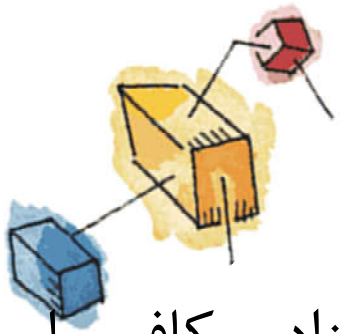


الگوریتم بهترین برازش (Best-fit)

- کوچکترین بلاکی را که فرآیند در آن جا می شود انتخاب می کند.
- برخلاف اسمش، بیشترین هزینه اجرا و بدترین کارایی را دارد.

- به دنبال فضایی می گردد که اندازه آن به اندازه فرآیند بسیار نزدیک باشد، در نتیجه کوچکترین میزان فضای باقیمانده را ایجاد می نماید.

- به دلیل ایجاد قطعات بسیار کوچک فضای آزاد در حافظه، اغلب موجب افزایش مشکل تکه تکه شدن خارجی می شود.
- بنابراین عملیات فشرده سازی باید بیشتر انجام شد (که کاری زمانبر است).

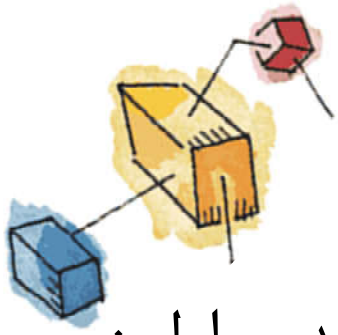


الگوریتم اولین برازش (First-fit)

- این الگوریتم، حافظه را از ابتدا مرور می کند و اولین بلاک آزاد و کافی را به فرآیند اختصاص می دهد.

- ساده ترین و سریعترین الگوریتم است.

- ممکن است قسمت ابتدایی حافظه را از تکه های کوچک انباشته کند که هر بار بایستی برای یافتن فضای خالی مورد نیاز جستجو شوند.



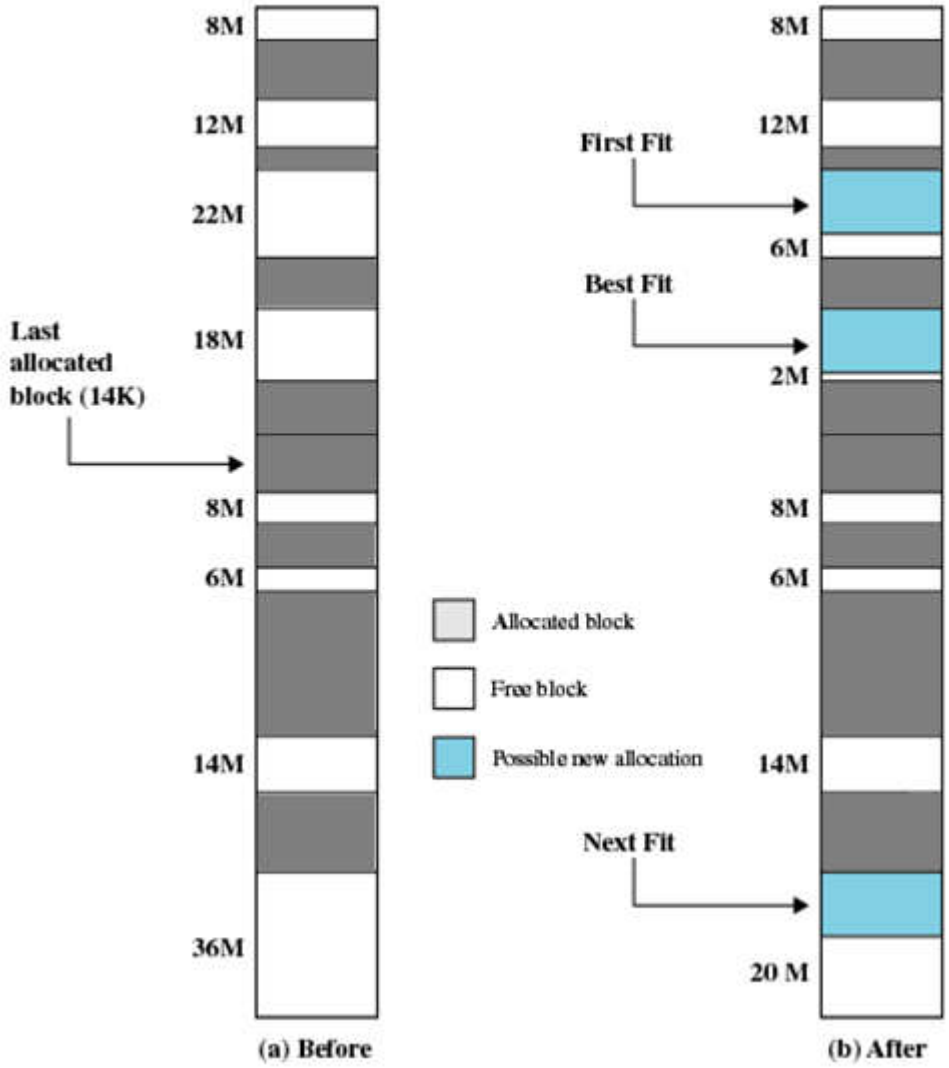
الگوریتم در پی برازش (Next-fit)

- حافظه را از محل آخرین جایگذاری به بعد مرور می کند و اولین بلاک با اندازه کافی را انتخاب می کند.

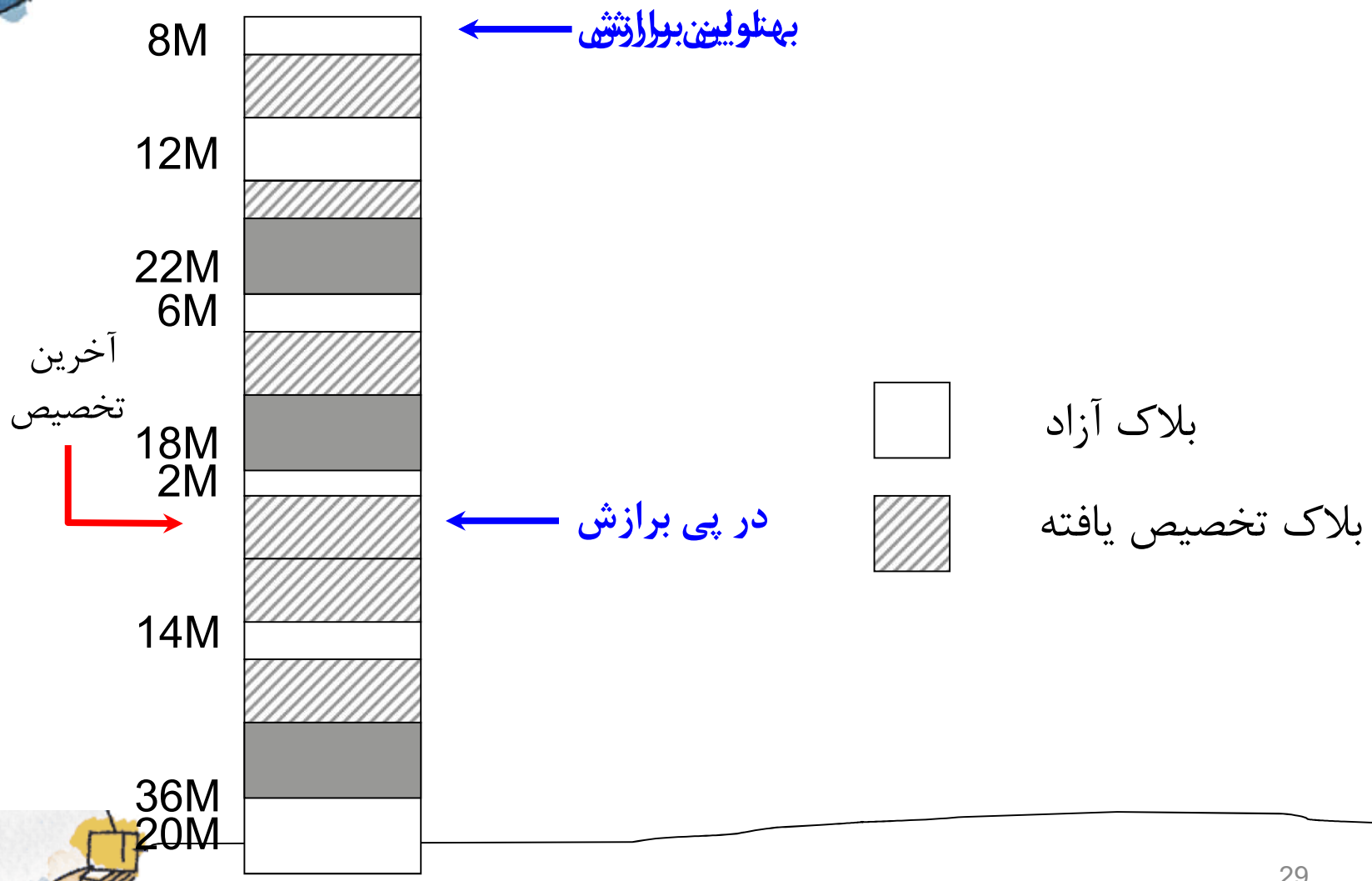
- بلاک های بزرگ حافظه سریعا به بلاک های کوچکتر تقسیم می شوند.

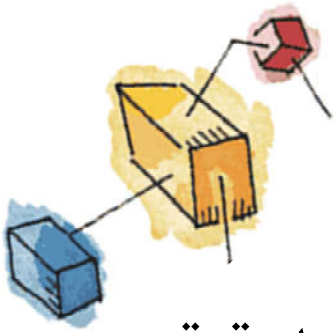
- برای بدست آوردن یک بلاک بزرگ در انتهای حافظه، مکررا نیازمند فشرده سازی است.

پیکر بندی حافظه قبل و بعد از تخصیص یک بلاک ۱۶ مگا بایتی



پیکر بندی حافظه قبل و بعد از تخصیص یک بلاک ۱۶ مگا بایتی

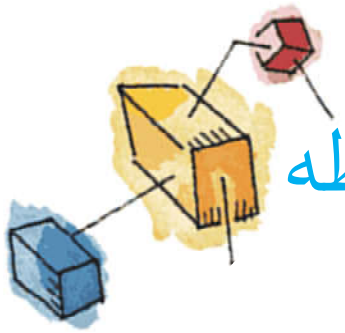




الگوریتم های جایگذاری بخش بندی پویا

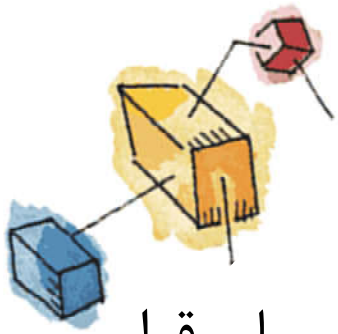
- اینکه کدام الگوریتم تخصیص پویا، مناسب تر است به ترتیب دقیق مبادله فرآیندها و اندازه آنها بستگی دارد.

- ولی در حالت کلی می توان گفت که:
  - الگوریتم اولین برازش بهتر از بقیه است.
  - الگوریتم در پی برازش، کمی بدتر از اولین برازش است.
  - الگوریتم بهترین بزارش، بر خلاف اسمش، از الگوریتم های دیگر بدتر است



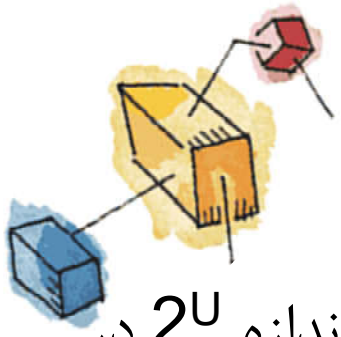
روش های مختلف برای بخش بندی حافظه

1. بخش بندی ایستا
2. بخش بندی پویا
3. سیستم رفاقتی (Buddy System)
4. صفحه بندی ساده (Paging)
5. قطعه بندی ساده (Segmentation)



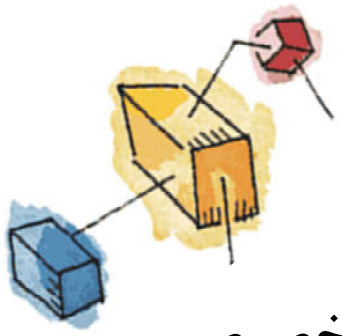
سیستم رفاقتی (Buddy System)

- این روش تعادلی را برای غلبه بر مشکلات بخش بندی ایستا و پویا برقرار می کند.
- در این روش، بلوک های حافظه اندازه هایی به شکل 2^k دارندکه در آن $L<=k<=U$
- و 2^L اندازه کوچکترین بلوک تخصیص یافته و 2^U اندازه بزرگترین بلوک تخصیص یافته است.



سیستم رفاقتی (Buddy System)

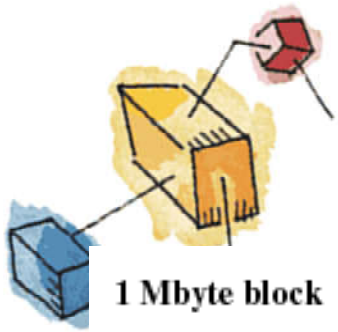
- برای شروع، کل فضای موجود حافظه به صورت یک بلاک واحد به اندازه 2^U در نظر گرفته می شود.
- وقتی درخواستی برای اختصاص حافظه به اندازه S دریافت شود آنگاه:
  - اگر $2^{U-1} < S <= 2^U$ باشد، کل بلاک تخصیص می یابد.
  - در غیر اینصورت، بلاک به دو بخش (رفیق) مساوی با اندازه های 2^{U-1} تقسیم می شود.
  - اگر $2^{U-2} < S <= 2^{U-1}$ باشد، یکی از این دو رفیق به درخواست تخصیص داده می شود. در غیر اینصورت یکی از این دو رفیق، دوباره به دوبخش تقسیم می شود و ...
  - این فرآیند ادامه می یابد تا زمانی که کوچکترین بلاکی که بزرگتر یا مساوی S باشد، تولید گردد.
  - این بلاک به درخواست تخصیص داده می شود.



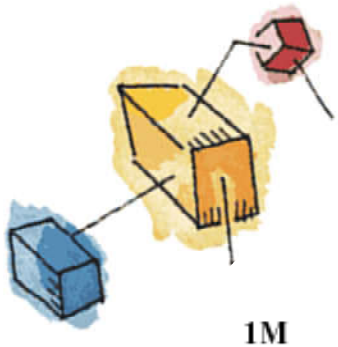
سیستم رفاقتی (Buddy System)

- اگر پس از مدتی حافظه هایی آزاد شود و دو رفیق به صورت تخصیص نیافته درآیند، با هم تلفیق می شوند و به صورت یک بلاک واحد تبدیل می شوند.

مثالی از سیستم رفاقتی



1 Mbyte block	1 M					
Request 100 K	A = 128 K	128 K	256 K	512 K		
Request 240 K	A = 128 K	128 K	B = 256 K	512 K		
Request 64 K	A = 128 K	C = 64 K	64 K	B = 256 K	512 K	
Request 256 K	A = 128 K	C = 64 K	64 K	B = 256 K	D = 256 K	256 K
Release B	A = 128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Release A	128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Request 75 K	E = 128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Release C	E = 128 K	128 K	256 K	D = 256 K	256 K	
Release E	512 K			D = 256 K	256 K	
Release D	1 M					



مثالی از سیستم رفاقتی – نمای درختی

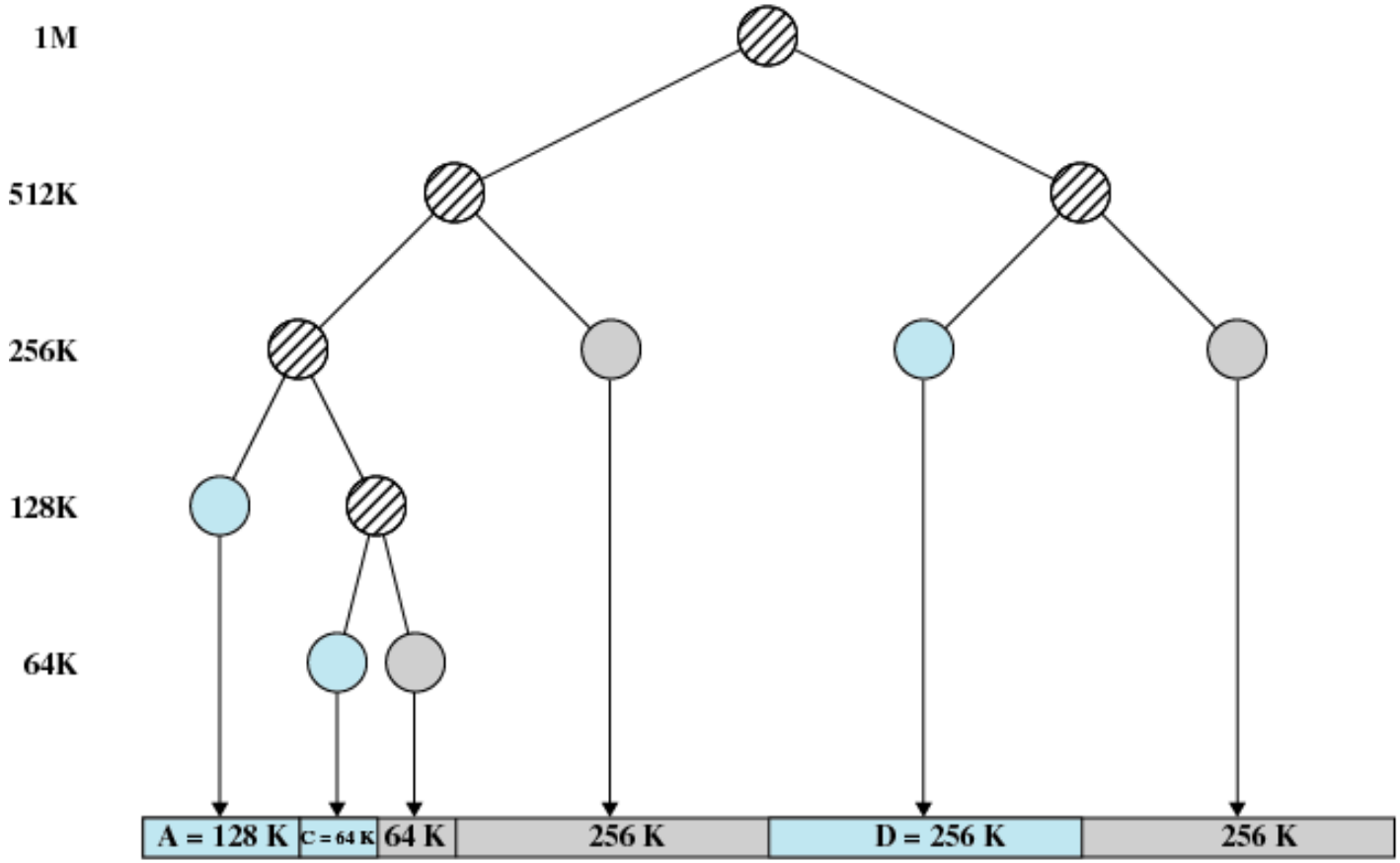
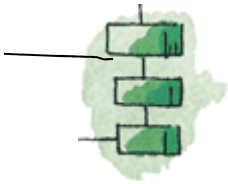
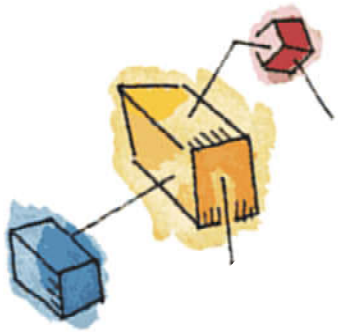
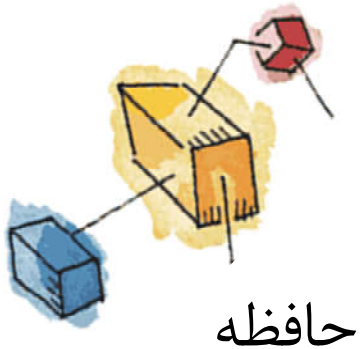


Figure 7.7 Tree Representation of Buddy System





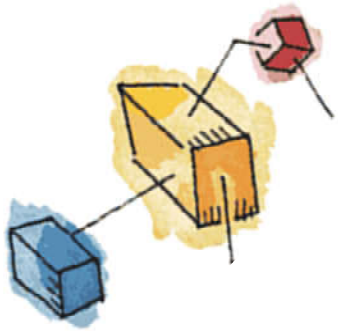
جابجایی (Relocation)



جابجایی (Relocation)

- زمانی که یک برنامه در حافظه بار می شود محل های واقعی حافظه تخصیص یافته به برنامه تعیین می شوند.
- یک برنامه ممکن است در حین اجرا، بخش های مختلفی از حافظه و بنابراین مکان های واقعی مختلفی از حافظه را اشغال کند.
- فشرده سازی موجب انتقال بخش های فرآیند می شود و این به معنای اشغال کردن محل های واقعی مختلفی از حافظه در طول اجرا است.

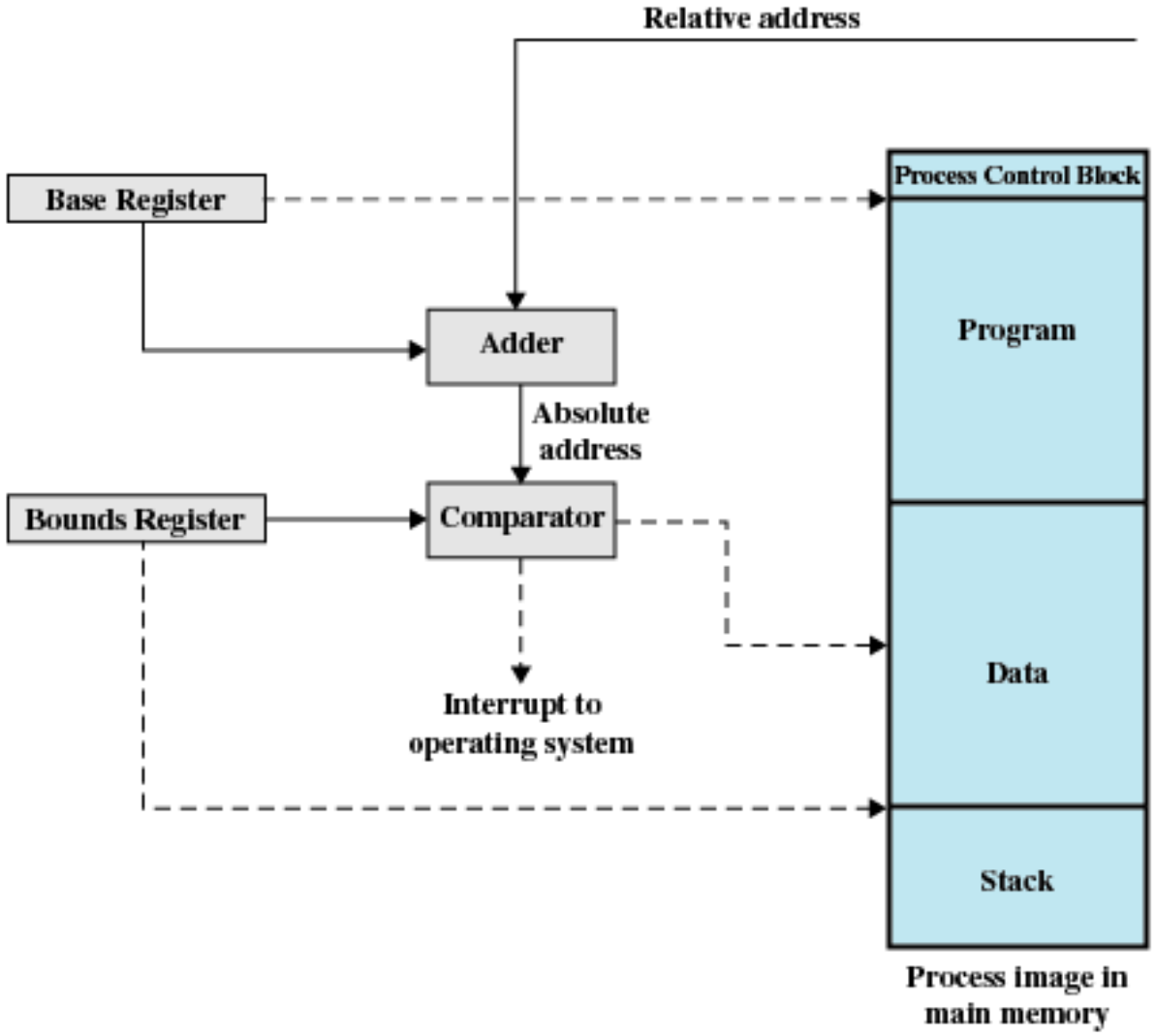
- برای حل این مساله که برنامه هر بار در جاهای مختلفی از حافظه قرار می گیرد، باید میان آدرس ها تفاوت قائل شویم و چند نوع آدرس داشته باشیم (آدرس فیزیکی، آدرس منطقی، آدرس نسبی)

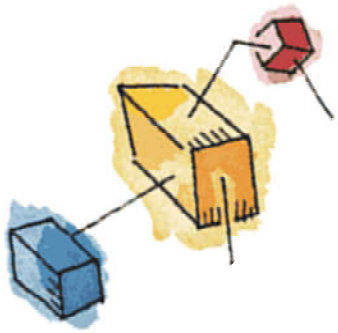


انواع آدرس

- **منطقی** (Logical)
  - ارجاع به محلی از حافظه، مستقل از چگونگی اختصاص داده جاری به حافظه.
  - آدرسی که برنامه کاربر همواره با آن سر و کار دارد.
  - برای دسترسی به حافظه، آدرس منطقی باید به آدرس فیزیکی ترجمه شود.
- **نسبی** (Relative)
  - آدرس بر اساس فاصله مکان مورد نظر نسبت به یک نقطه معلوم (مثلاً ابتدای برنامه) بیان می شود.
- **فیزیکی** (Physical or Absolute)
  - آدرس واقعی یا مطلق در حافظه اصلی، که توسط واحد حافظه قابل رویت است.

پشتیبانی سخت افزاری برای جابجایی

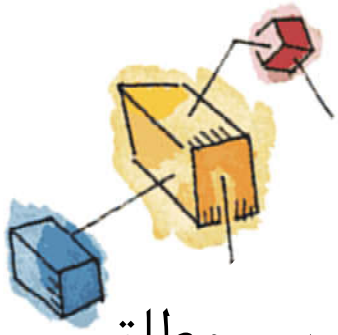




ثبات های مورد استفاده در هنگام اجرا

- ثبات پایه (Base register):
  - آدرس شروع فرآیند
- ثبات محدوده (Bounds register):
  - آدرس پایان فرآیند

- این ثبات ها هنگام بار شدن فرآیند به حافظه اصلی یا مبادله فرآیند به داخل، مقدار دهی می شوند.

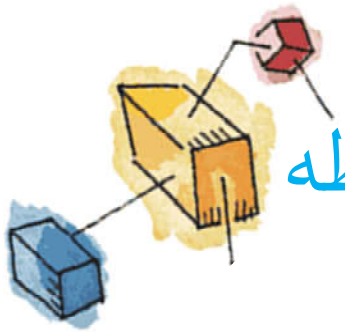


ثبات های مورد استفاده در هنگام اجرا

- مقدار ثبات پایه به آدرس نسبی افزوده می شود تا یک آدرس مطلق بدست آید.

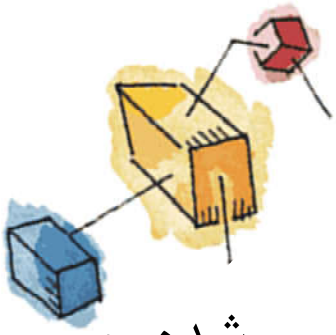
- آدرس بدست آمده با مقدار ثبات محدوده مقایسه می شود.

- اگر در محدوده باشد، اجرای دستورالعمل ها ادامه می یابد، در غیر این صورت خطایی از طریق وقفه به سیستم عامل داده می شود.



روش های مختلف برای بخش بندی حافظه

1. بخش بندی ایستا
2. بخش بندی پویا
3. سیستم رفاقتی (Buddy System)
4. **صفحه بندی ساده (Paging)**
5. قطعه بندی ساده (Segmentation)

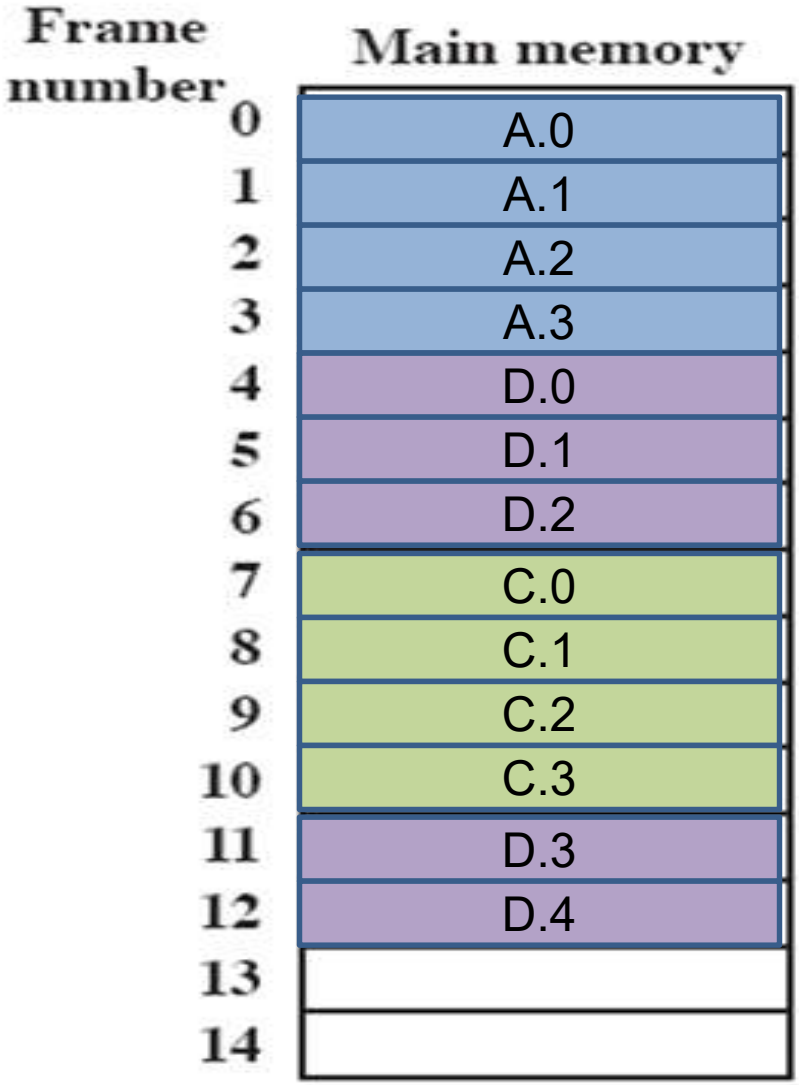


صفحه بندی (Paging)

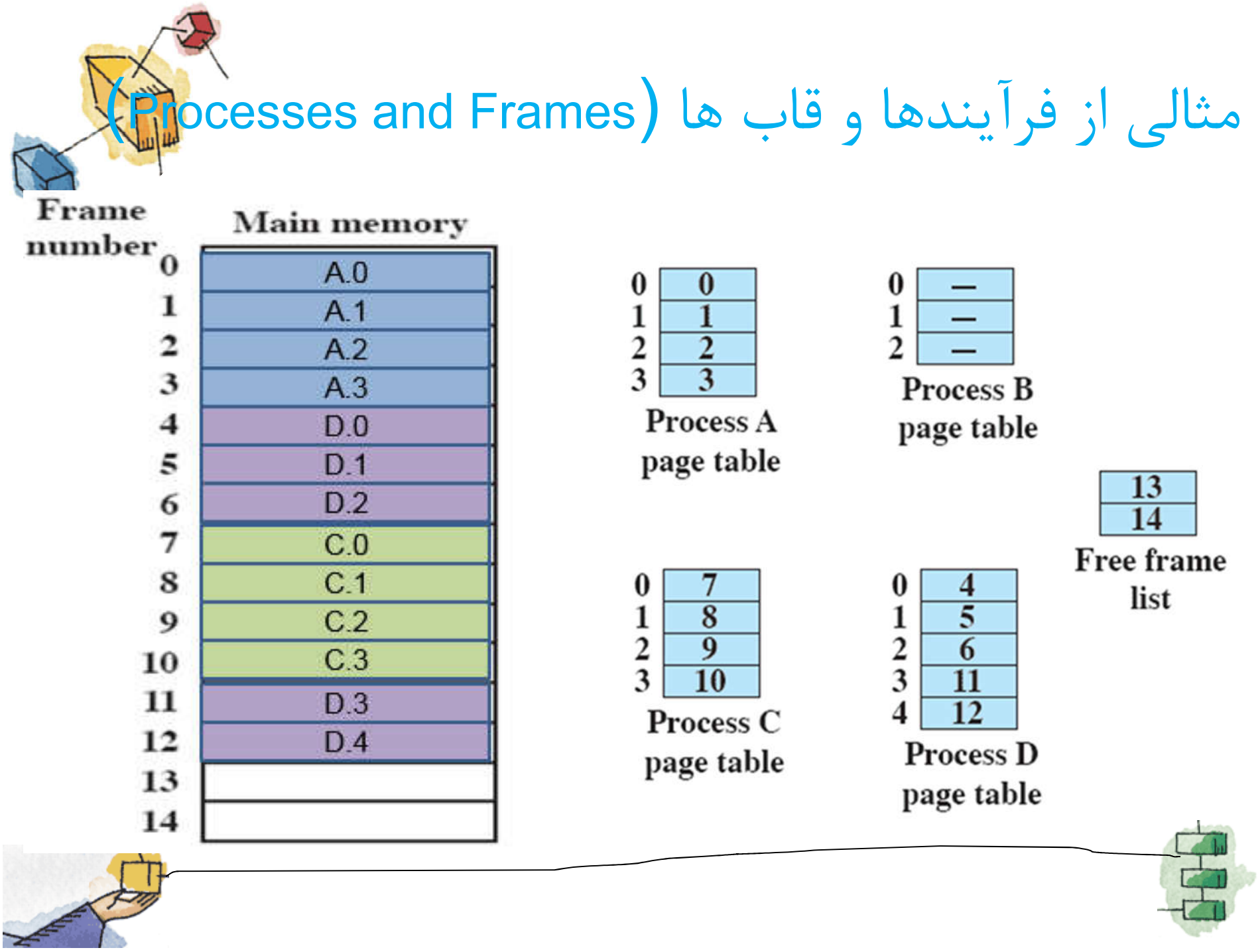
- بخش های حافظه به قطعات کوچکی با اندازه ثابت تقسیم شده و فرآیندها نیز به قطعاتی با همان اندازه تقسیم می شوند.
- قطعات یک فرآیند، **صفحه** (page) نامیده شده و قطعات مربوط به حافظه **قاب** (frame) نام دارند.

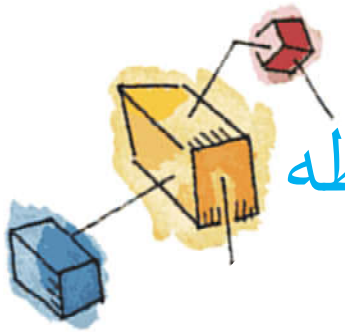
- سیستم عامل برای هر فرآیند یک جدول صفحه نگه می دارد.
  - حاوی موقعیت قابی است که هر صفحه از فرآیند را در بر دارد.
  - هر آدرس منطقی حافظه شامل شماره صفحه و یک انحراف در صفحه است.

مثالی از فرآیندها و قاب ها (Processes and Frames)



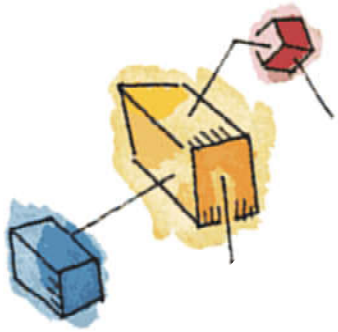
مثالی از فرآیندها و قاب ها (Processes and Frames)





روش های مختلف برای بخش بندی حافظه

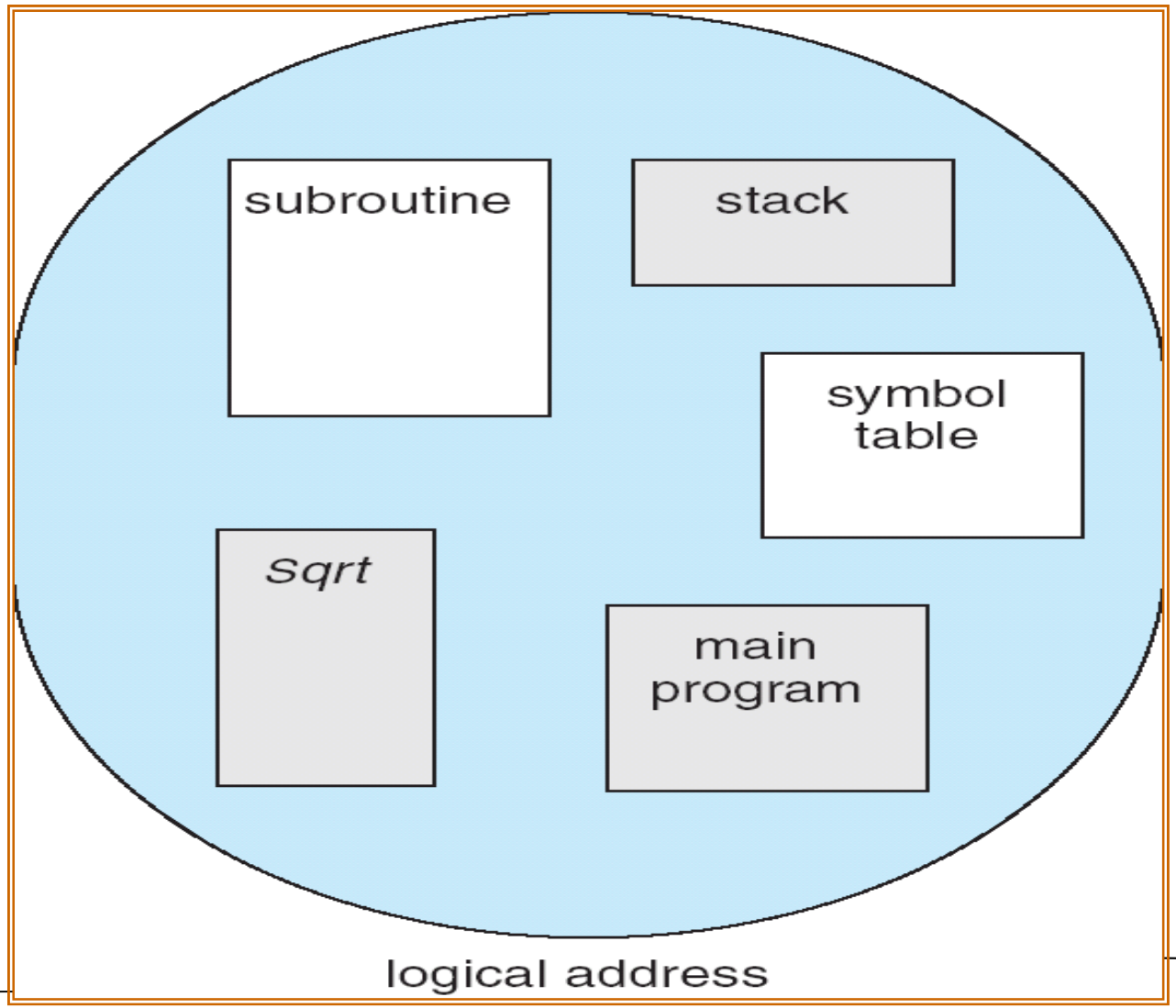
1. بخش بندی ایستا
2. بخش بندی پویا
3. سیستم رفاقتی (Buddy System)
4. صفحه بندی ساده (Paging)
5. **قطعه بندی ساده (Segmentation)**

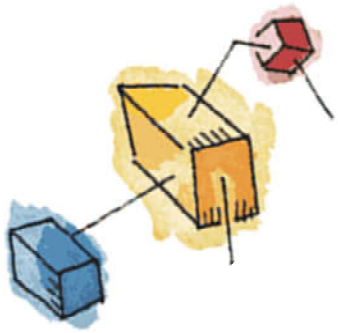


قطعه بندی (Segmentation)

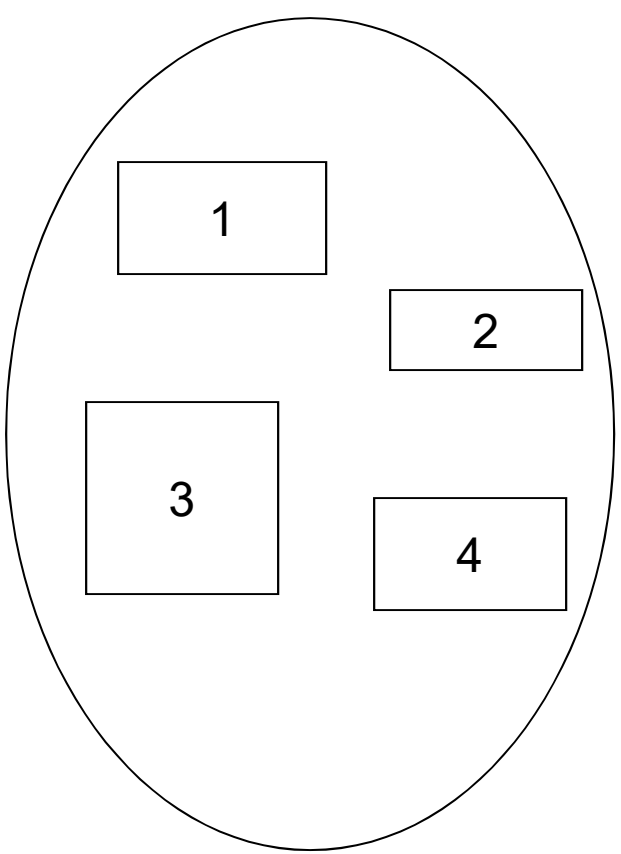
- لزومی ندارد همه قطعات تمام برنامه ها دارای اندازه یکسان باشند.
- فرآیند از لحاظ منطقی و به صورت معنی دار به بخش هایی تقسیم می شود.
- قطعه بندی، طرحی برای مدیریت حافظه است که از دید کاربر در مورد حافظه پیروی می کند.
- یک برنامه مجموعه ای از قطعه ها است. یک قطعه یک واحد منطقی است مثل:
  - برنامه ی اصلی
  - رویه
  - تابع
  - متد
  - شیء
  - متغیرهای محلی، متغیرهای سراسری
  - استک
  - جدول علائم
  - آرایه ها
  - ....
- حداکثر مقداری برای طول قطعه وجود دارد.

دید کاربر از یک برنامه



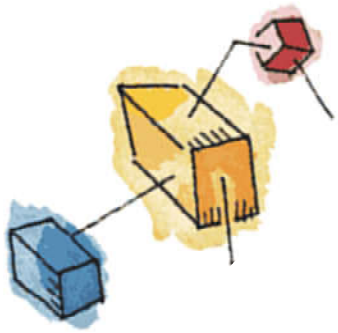


دید منطقی قطعه بندی



user space

physical memory space

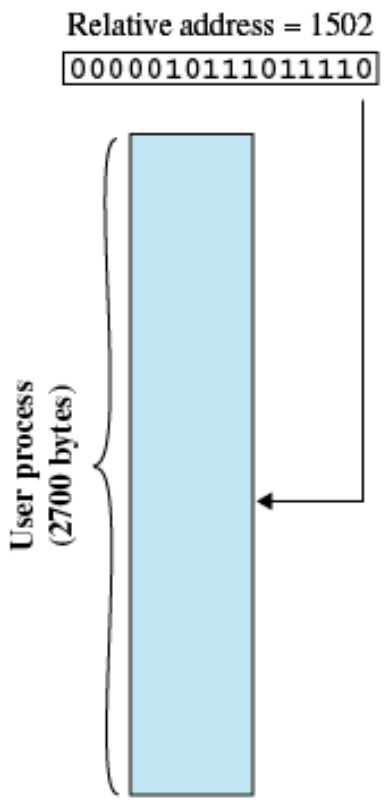


قطعه بندی (Segmentation)

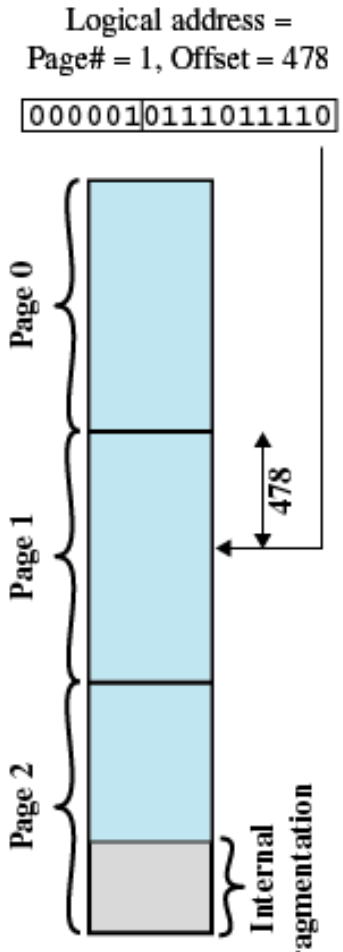
- آدرس منطقی شامل دو بخش است:
  - شماره قطعه
  - انحراف داخل قطعه

- ازآنجایی که قطعه ها دارای طول یکسانی نیستند، قطعه بندی شبیه بخش بندی پویا است.

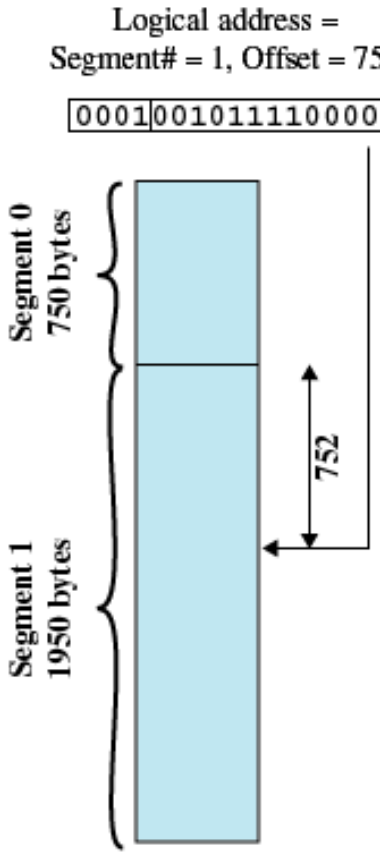
آدرس های منطقی در سه روش بخش بندی، صفحه بندی، قطعه بندی



(a) Partitioning



(b) Paging
(page size = 1K)



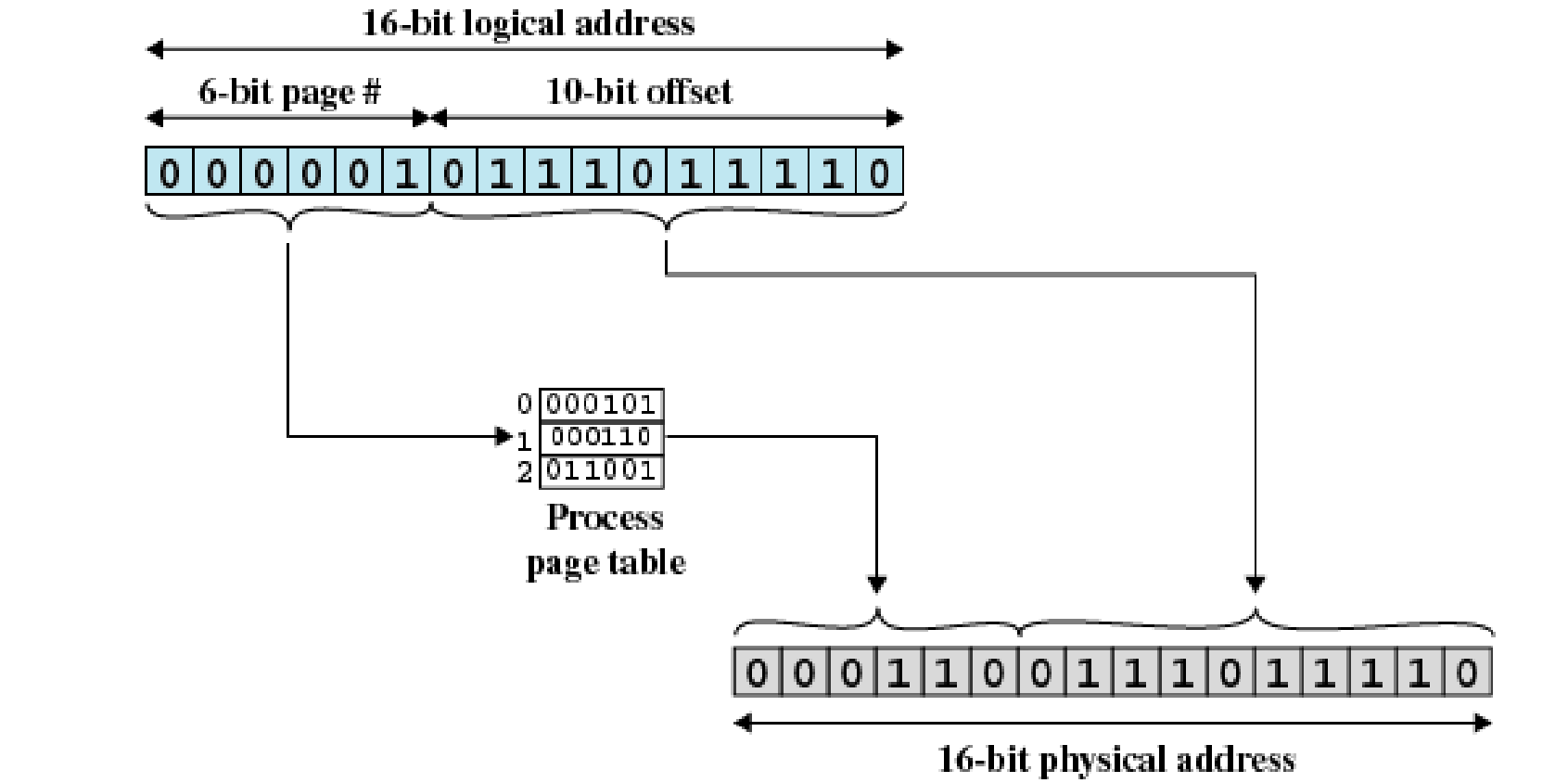
(c) Segmentation

روش بخش بندی
(ایستا یا پویا)

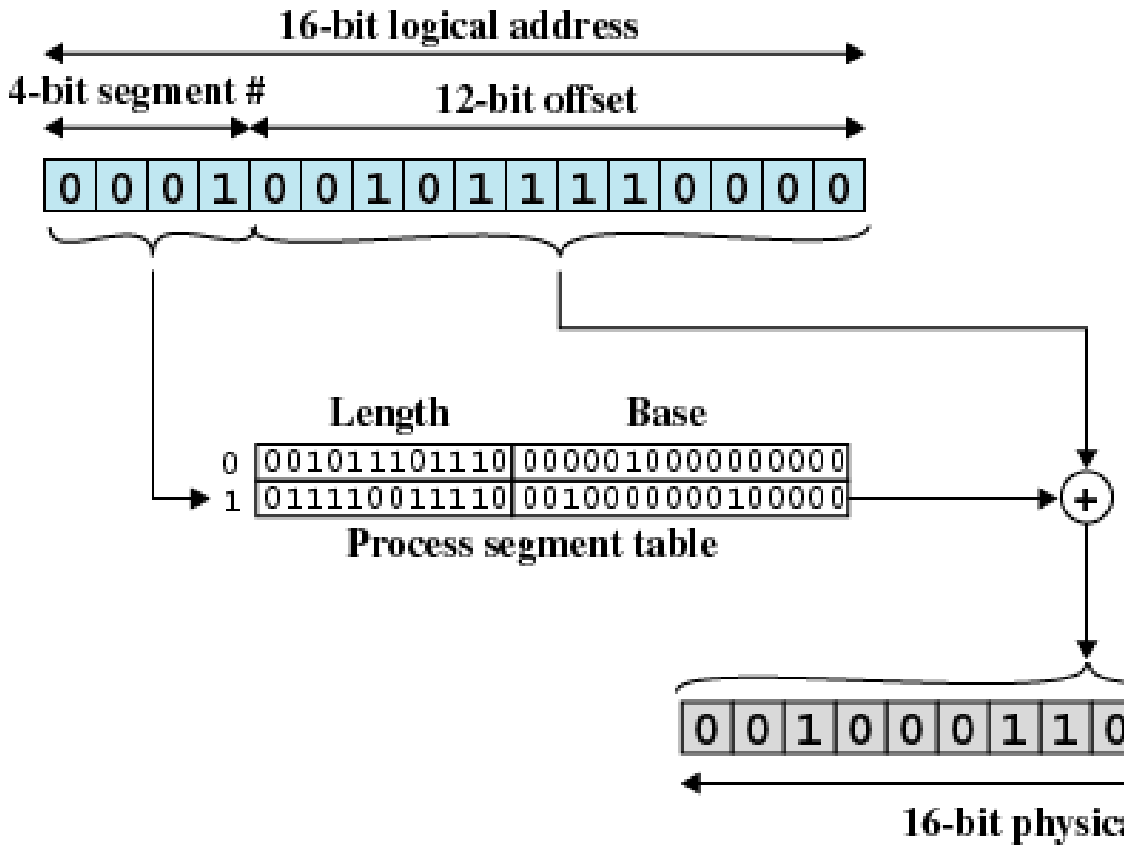
روش صفحه بندی

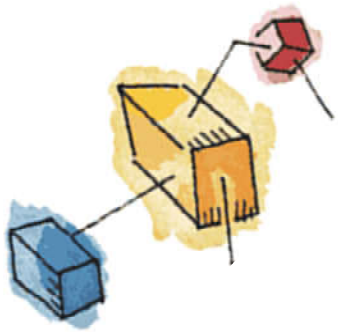
روش قطعه بندی

ترجمه آدرس منطقی به فیزیکی در صفحه بندی



ترجمه آدرس منطقی به فیزیکی در قطعه بندی





پایان فصل هفتم