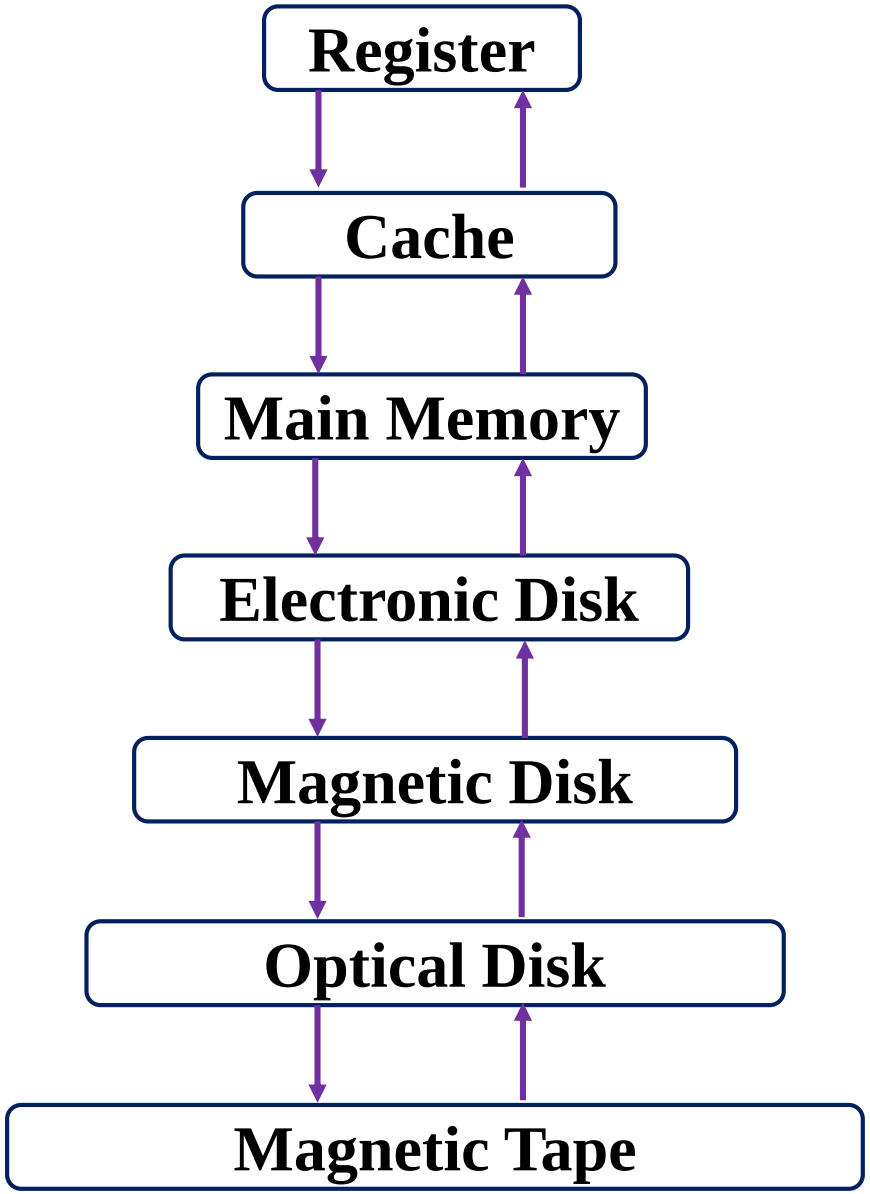
فصل ششم

سلسله مراتب حافظه

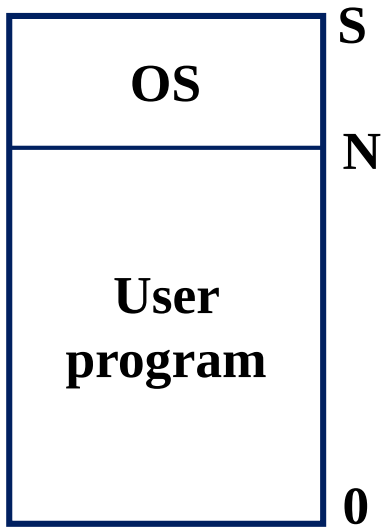


تک برنامگی ساده

- در این حالت، یک برنامه در RAM قرار داده می شود و اجرا می گردد. اگر حافظه کم باشد، برنامه اجرا نمی شود. سیستم عامل ابتدایی DOS از این روش استفاده می کرد.
  - ✓ قسمتی از حافظه برای سیستم عامل استفاده می شده(۶۴۰ کیلو بایت) که از دسترس برنامه کاربر محافظت می شد.
  - ✓ رجیستر Limit Register بالاترین آدرسی که برنامه کاربر می توانسته استفاده کند را نگه می دارد، اگر برنامه ای از این بیشتر آدرسی تولید می کرد ، وقفه نقص برنامه (Program Fault) تولید می شد.

Limit Register | N

Address in User Program > N ⇨ Program Fault



تک برنامگی با سیستم (جایگشت) Overlay

- در این روش برنامه کاربر توسط برنامه نویس به چند قطعه تقسیم می شود. قطعات در حافظه جانبی ذخیره شده و هر دفعه یک قطعه جهت اجرا به حافظه اصلی آورده می شود و در صورت نیاز جابجا می شوند.
- در این روش برنامه می تواند بزرگتر از حافظه باشد.
  - در سیستم عامل DOS استفاده می شده و فایلهای EXE به همراه یک فایل OVL نگهداری می شدند.

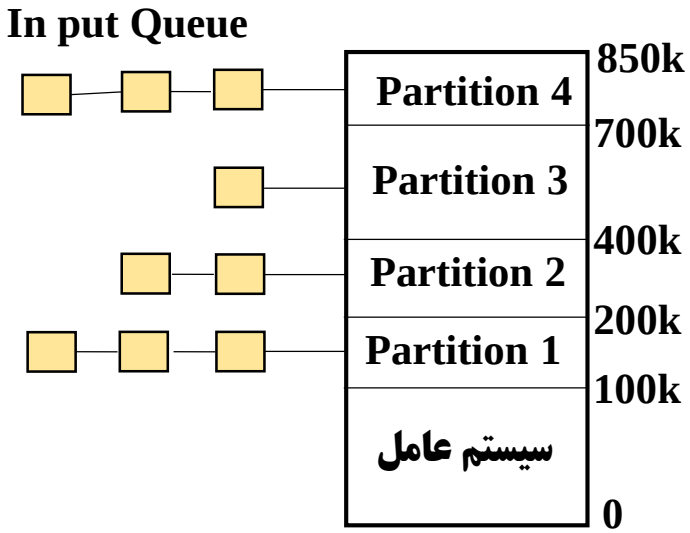
سیستم مبادله (Swapping) ساده

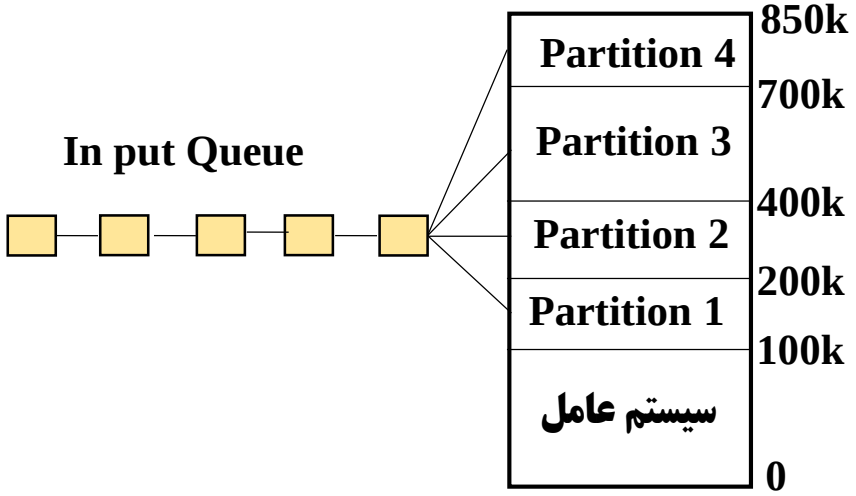
- در این روش یک پردازش در حافظه اجرا می شودو در صورت نیاز با دیگر پردازشها در حافظا جانبی مبادله می شوند. پس زمان پاسخ یک برنامه برابر خواهد بود با زمان اجرا و دوبرابر زمان مبادله(رفت و برگشت)
- در سیستمهای اشتراک زمانی استفاده می شود و این زمان مبادله از مشکلات آن می باشد
- در صد استفاده از CPU از رابطه زیر محاسبه می شود.

$$\text{در صد استفاده از CPU} = \frac{\text{اجرا زمان}}{\text{مبادله زمان} \times 2 + \text{اجرا زمان}} \times 100$$

چند برنامگی با بخش بندی ثابت حافظه

- در این روش، حافظه به چند بخش مساوی یا متفاوت تقسیم می شود. هر JOB به کوچکترین بخش که جا شد، انتقال داده می شود.

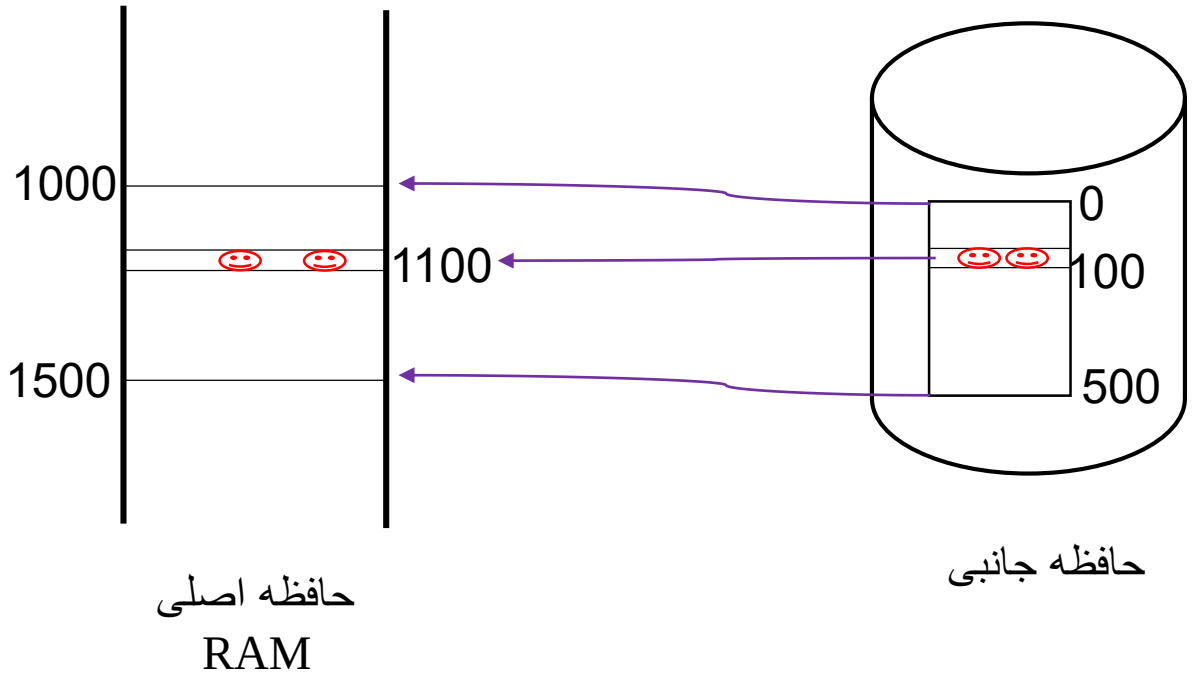




انواع صف برای انتقال به بخشها

تبدیل آدرسها

- برنامه کاربر در حافظه جانبی دارای آدرسی می باشد که این برنامه در حافظه اصلی ممکن است در همان آدرس قرار نگیرد. بنابراین آدرسها به هم پیوند می خورند یا **Bind** می شوند.



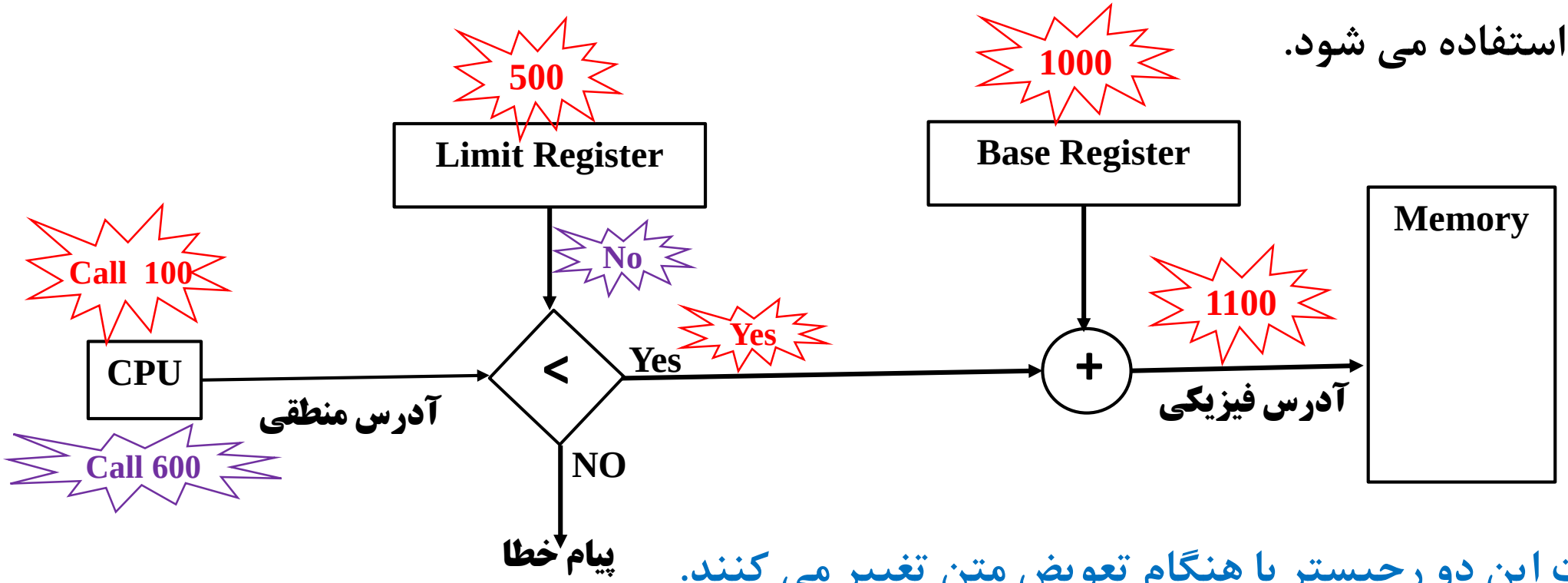
تبدیل آدرسها

❑ آدرس درون برنامه در سه حالت پدید می آید:

- ✓ زمان کامپایل کردن، در این حالت آدرسهای درون برنامه مطلق بوده و بدون پیوند یا Bind شدن، باید در حافظه قرار بگیرد.
- ✓ زمان بار گزاری، آدرسها مطلق نیستند و وقتی برنامه در حافظه Load شد، آدرسهای برنامه به آدرس حافظه پیوند داده می شوند. (Bind می شوند)
- ✓ زمان اجرا، آدرسها در زمان اجرا پدید می آیند که احتیاج به سخت افزار خاصی می باشد.

آدرس فیزیکی و منطقی (Physical & Logical Address)

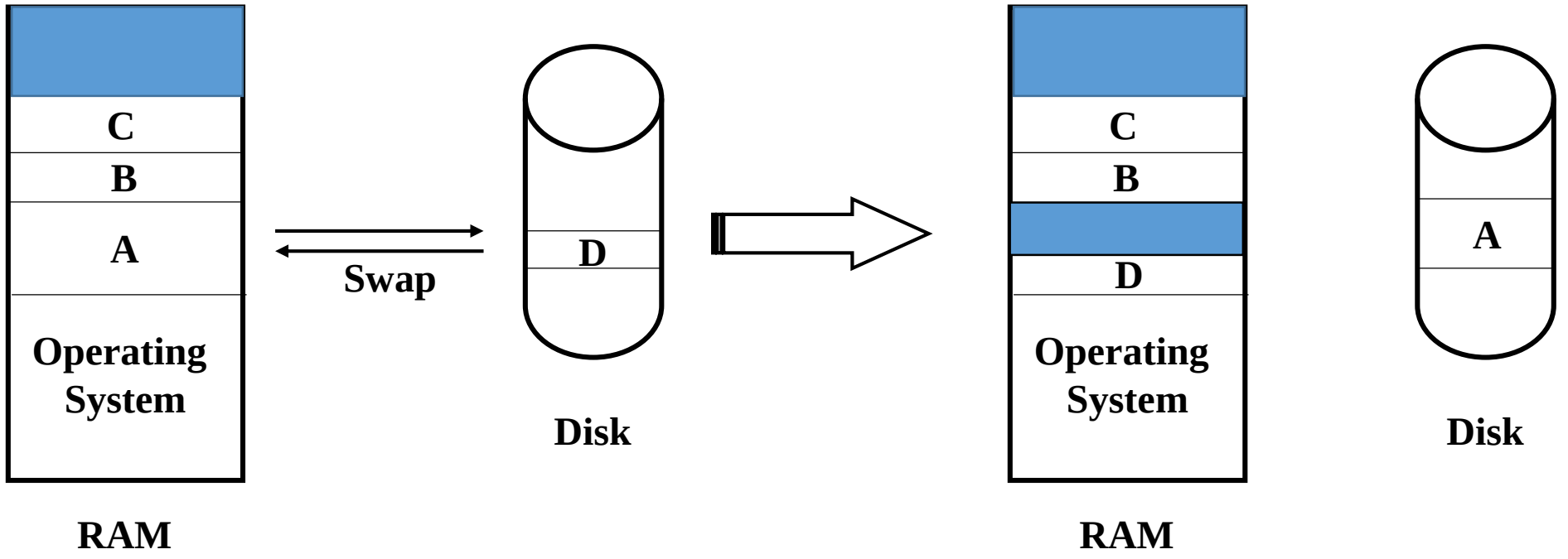
آدرسی که در هنگام اجرای پردازش توسط CPU ایجاد می شود، را آدرس منطقی یا Virtual Address گفته می شود. آدرسی که در آن محل، پردازش در حافظه قرار دارد، آدرس فیزیکی می باشد. برای بالا رفتن سرعت نگاشت این آدرسها به یکدیگر از سخت افزاری به نام واحد مدیریت حافظه Memory Management Unit یا MMU استفاده می شود.



محتویات این دو رجیستر با هنگام تعویض متن تغییر می کنند.

مبادله (Swapping)

❑ در این روش که در سیستم عامل جهت ایجاد قابلیت چند برنامگی ایجاد می شود، یک پردازش بطور کامل از حافظه به حافظه جانبی منتقل می شود و برعکس.



❖ اگر زمان انتقال پردازش از حافظه اصلی به دیسک و برعکس 100ms و زمان مکان یابی در دیسک 8ms باشد و زمان اجرای پردازش 10ms باشد. در صد بکارگیری CPU را محاسبه کنید.

$$\text{زمان بکارگیری } CPU = \frac{10}{10 + 2 \times \ 108} \times 100 = 4.43\%$$

مبادله (Swapping)

- ✓ممکن است قسمتهایی از حافظه خالی بمانند. یعنی حفره ایجاد شود.
- ✓پردازشها باید آدرس دهی جابجا پذیر **Relocatable** داشته باشند.
- ✓پردازشی که منتظر وقفه **I/O** باشد به دیسک منتقل شود ولی اگر برش زمانی تمام شده باشد، در حافظه بماند.

روشهای تخصیص حافظه

- بعد از مدتی که پردازشها اجرا شوند، در حافظه فضاهای خالی مانند حفره بوجود می آید، سیستم عامل به یکی از روشهای زیر این حفره ها را به پردازشها اختصاص می دهد.
  - **First Fit :** سیستم عامل لیست فضاهای آزاد را از ابتدای لیست جستجو می کند، اولین فضای آزاد که به اندازه پردازش یا بیشتر از آن وجود داشت را به پردازش اختصاص می دهد. بنابراین تراکم فضاهای اشغال شده در ابتدای لیست زیاد می شود.
  - **Next Fit :** سیستم عامل لیست فضاهای آزاد را از آنجایی که قبلا اختصاص داده به بعد را جستجو می کند، اولین فضای آزاد که به اندازه پردازش یا بیشتر از آن وجود داشت را به پردازش اختصاص می دهد. بنابراین یکنواختی توزیع فضاهای اشغال شده در لیست بیشتر می شود. احتمال تکه تکه شدن حافظه و پیدا نکردن فضای مناسب بیشتر می شود.

روشهای تخصیص حافظه

- **Best Fit :** سیستم عامل همه لیست فضاهای آزاد را جستجو می کند، اولین فضای آزاد که به اندازه پردازش وجود داشت را به پردازش اختصاص می دهد. بنابراین فضاهای بزرگتر برای پردازشهای بزرگتر باقی می ماند یا حافظه کمتر تکه تکه می شود. اما جستجوی کل لیست زمانگیر خواهد بود.

- **Worst Fit :** سیستم عامل همه لیست فضاهای آزاد را جستجو می کند، بزرگترین فضای آزاد را به پردازش اختصاص می دهد. بنابراین پردازشهای بزرگ منتظر باقی می مانند چون حافظه بیشتر تکه تکه می شود و جستجوی کل لیست زمانگیر خواهد بود.

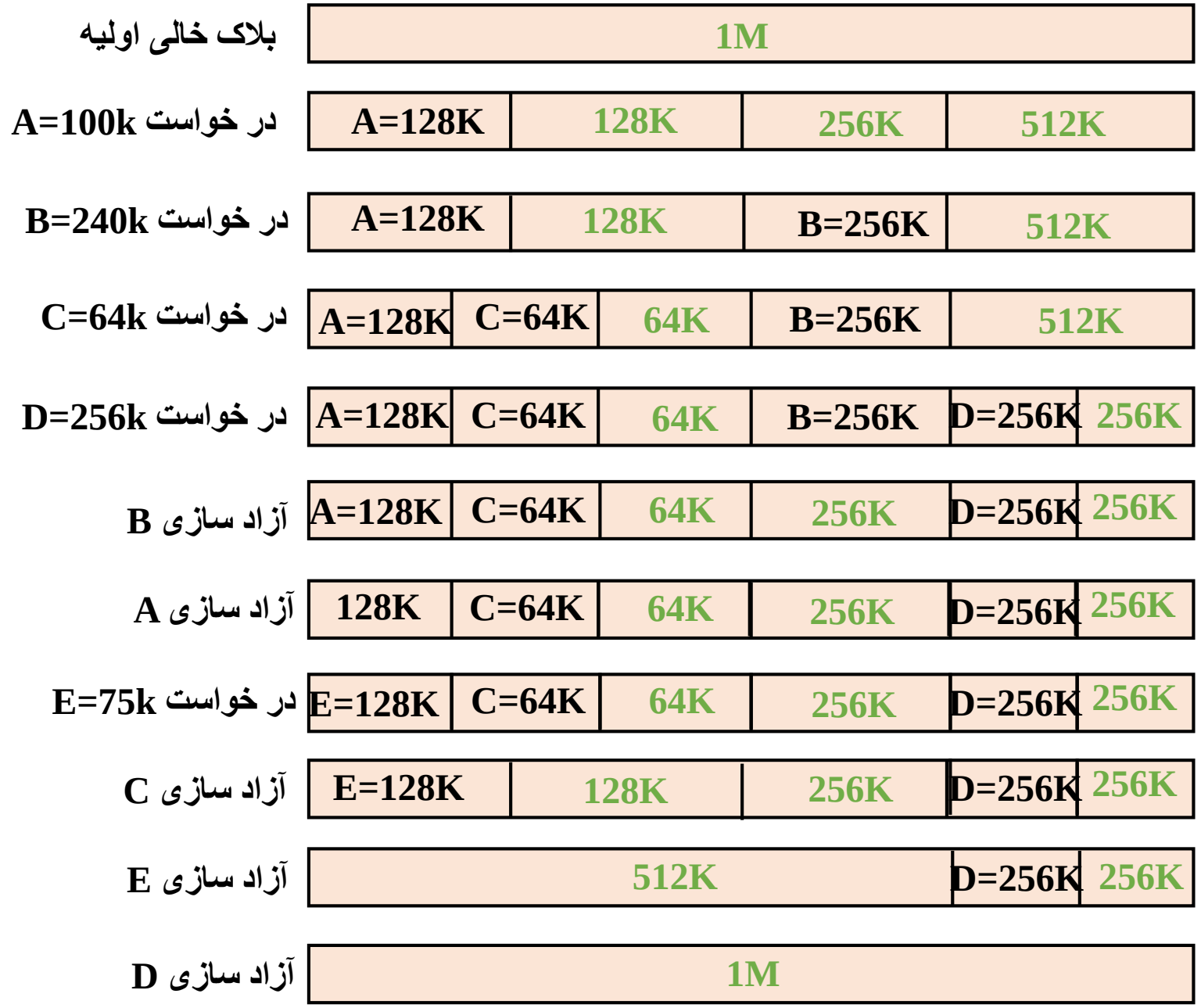
- **Quick Fit :** سیستم عامل لیستی دارد که هر خانه آن به لیست فضاهای یکسان اشاره دارد (**4k** یا **8k** یا **16k**)فضاهای آزاد را جستجو می کند، مناسبترین فضای آزاد را به پردازش اختصاص می دهد. بنابراین خیلی سریع به فضای مناسب دست پیدا می کند، اما برگرداندن فضای خالی شده به لیست مناسب آن زمانگیر می باشد.

روشهای تخصیص حافظه

- **Buddy** : یا الگوریتم رفاقتی : در این روش همه فضای حافظه به قطعاتی از توان ۲ تقسیم می شود . پردازش در فضای مناسب آن قرار داده می شود و پس از اتمام کار پردازشها، قطعات به هم می پیوندند. اینکه قطعات کوچکتر می توان درست کرد و از پیوند قطعات کوچک قطعه بزرگتر ایجاد نمود، باعث بهره وری بالاتری برای این الگوریتم شده است. از ضعفهای آن، طول قعات که باید توانی از ۲ باشند، می باشد.

روشهای تخصیص حافظه

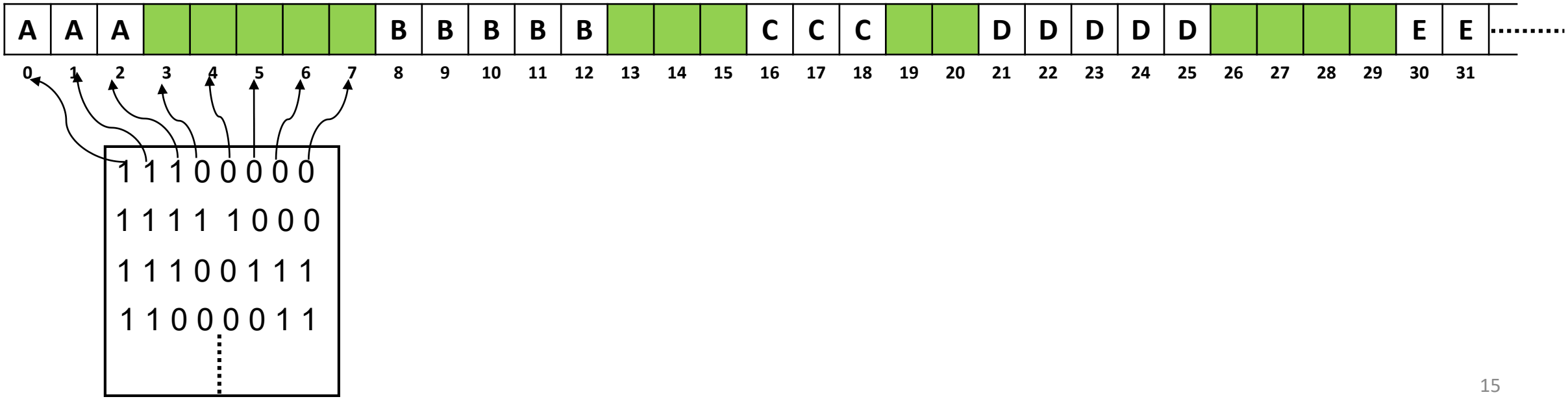
Buddy یا الگوریتم رفاقتی



روشهای تخصیص حافظه

مدیریت فضاهای آزاد

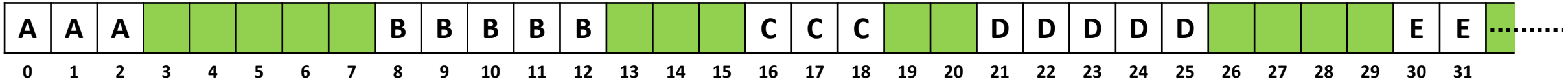
- برای پیدا کردن فضای آزاد در حافظه که بتوان پردازش را درون آن قرار داد. از دو روش استفاده می گردد.
- ✓ روش نگاشت بیتی (Bit Map) حافظه به قطعات کوچکی تقسیم می شود (مثلا 512B) و در قسمتی از حافظه، جدولی وجود دارد که متناظر با هر قطعه یک بیت دارد؛ اگر قطعه خالی باشد، بیت=۱ و اگر قطعه اشغال باشد، بیت=۰ خواهد بود.
- ❖ برای ذخیره هر پردازش باید جدول را جستجو کرد تا تعداد ۱ معادل ظرفیت پردازش پیدا نمود. که کاری زمانبر می باشد.

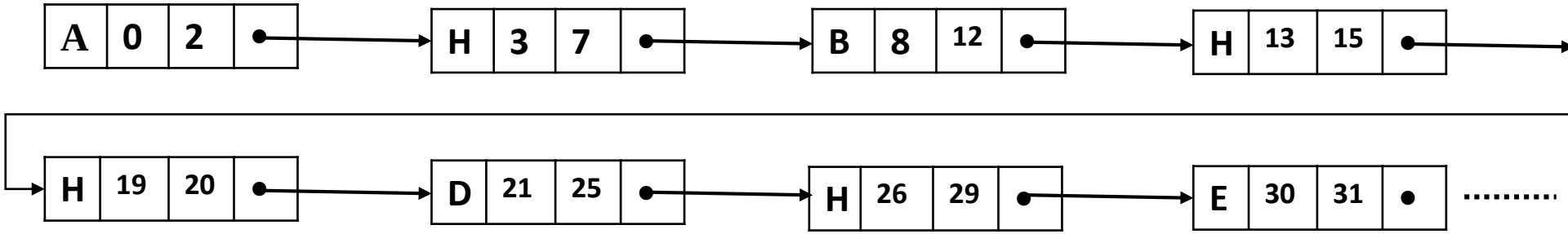


روشهای تخصیص حافظه

مدیریت فضاهای آزاد

- ✓ روش لیست پیوندی(Linked List) لیستی از وضعیت فضا های روی دیسک درست می شود. در قسمت از لیست سه عنصر و یک اشاره گر به قسمت بعد وجود دارد. حرف P و شماره بلاک آغازی و شماره بلاک پایانی و یا حرف H (Hole) قرار داده می شود.
- ❖ لیست سریع تغییر می کند و این روش فقط برای حافظه RAM کاربرد دارد. برای دیسک مناسب نمی باشد.





مشکلات مدیریت حافظه

- بطور منطقی هر پردازش در محلی از حافظه ذخیره شده و به کمک ثباتهای حد و پایه محافظت می شود. روشهای تخصیص حافظه، مشکلاتی خواهند داشت که عبارتند از: پارگی یا پراکندگی داخلی و خارجی، پراکندگی محل برنامه ها و استفاده اشتراکی از داده ها یا کدها.

➢ پارگی داخلی و خارجی (Internal & External Fragmentation)

هنگامیکه فضاهای خالی با ظرفیت کم، همجوار نباشند نمی توانند به هم پیوند زده شده تا فضای بزرگتری ایجاد کنند و پراکنگی خارجی ایجاد می شود. مثلا در الگوریتم FCFS در حدود نیمی از حافظه بدون استفاده می ماند. هنگامیکه فضایی به پردازشی اختصاص داده می شود، ممکن است مقدار کمی فضای زیادی داشته باشد که به علت کم بودن، ارزش تفکیک کردن ندارد. این حالت را پارگی داخلی می گویند.

- ❖ جهت رفع پارگی خارجی، در صورتیکه پردازشها قابلیت جابجایی داشته باشند(Relocatable) یعنی آدرس درون پردازش در هنگام اجرا مشخص شود. آنگاه می توان فضاهای خالی پراکنده را با جابجا کردن پردازشها، یکی کرد.

مشکلات مدیریت حافظه

- **پراکندگی برنامه ها(Sparseness)**

در این حالت دو نوع پراکندگی برنامه ای پیش می آید، استاتیک یا ایستا و دینامیک یا پویا.
در حالت استاتیک: هنگام کامپایل کردن برنامه پیش می آید که کامپایلر فضایی بزرگتر برای برنامه در نظر می گیرد، مثلا برای لیست توابع یا لیست داده ها، این فضای استفاده نشده با برنامه به حافظه برده می شود و راندمان را پایین می آورد.
در حالت پویا : در یک برش زمانی، قسمت کوچکی از برنامه مورد نیاز برای پردازش می باشد ولی قسمتهای زیادی بدون استفاده به حافظه برده شده اند و باعث ایجاد پراکندگی می گردند.

- **استفاده مشترک از کد و داده**

در سیستمهایی **Multitasking** ، چند کاربر از یک کامپایلر استفاده مشترک می کنند. باید کد برنامه ها از داده ها جدا باشند **(Pure Code)** و برنامه ها نتوانند شیوه دسترسی به کامپایلر را تغییر دهند **(Self modifying)** نباشند.