فصل سوم:
حافظه (Memory)

مریم تقی زاده

مهرماه ۱٤۰٤

Email: taghizadehmail@gmail.com

رئوس مطالب فصل ۳

- حافظه اصلی
- اصول مدیریت حافظه
  - تک برنامگی
  - چندبرنامگی
  - مدیریت دینامیک حافظه
- روش‌های تخصیص حافظه مبتنی بر همجواری
  - الگوریتم های برازش اول، بعدی، بهترین، بدترین، سریع، رفاقتی
- صفحه‌بندی (Paging)
  - مفاهیم، جدول صفحه، بیت های کنترلی، پیاده سازی سخت افزاری TLB
- قطعه بندی (Segmentation)
- حافظه مجازی

حافظه

- حافظه یکی از بخش‌های اصلی سیستم‌های کامپیوتر است. دارای تعدادی کلمه است که اطلاعات (داده‌ها، دستورات و برنامه‌ها) در آن ذخیره می‌شود. هر کلمه نیز دارای آدرسی است.

- حافظه اصلی (RAM) سریع است اما نسبت به CPU بسیار کُندتر است. بنابراین برای کاهش انتظار زمانی، از حافظه سریع به نام حافظه کش (نهان) استفاده می‌شود که بین CPU و حافظه اصلی قرار می‌گیرد.
  - امروزه، سیستم های کامپیوتری از سلسله مراتب حافظه استفاده می کنند.

- بخشی از سیستم عامل که سلسله مراتب حافظه را اداره می کند مدیر حافظه نامیده می شود.
  - کدام قسمت از حافظه اصلی خالی و کدام قسمت استفاده شده است
  - اگر چندین برنامه همزمان بخواهند اجرا شوند، باید همه آن ها به حافظه منتقل شود و OS از تداخل آنها در حافظه جلوگیری کند.

اصول مدیریت حافظه

- سیستم های مدیریت حافظه
  - مبتنی بر جابجایی فرآیندهای در حال اجرا بین حافظه اصلی و دیسک
  - رویکرد بدون جابجایی

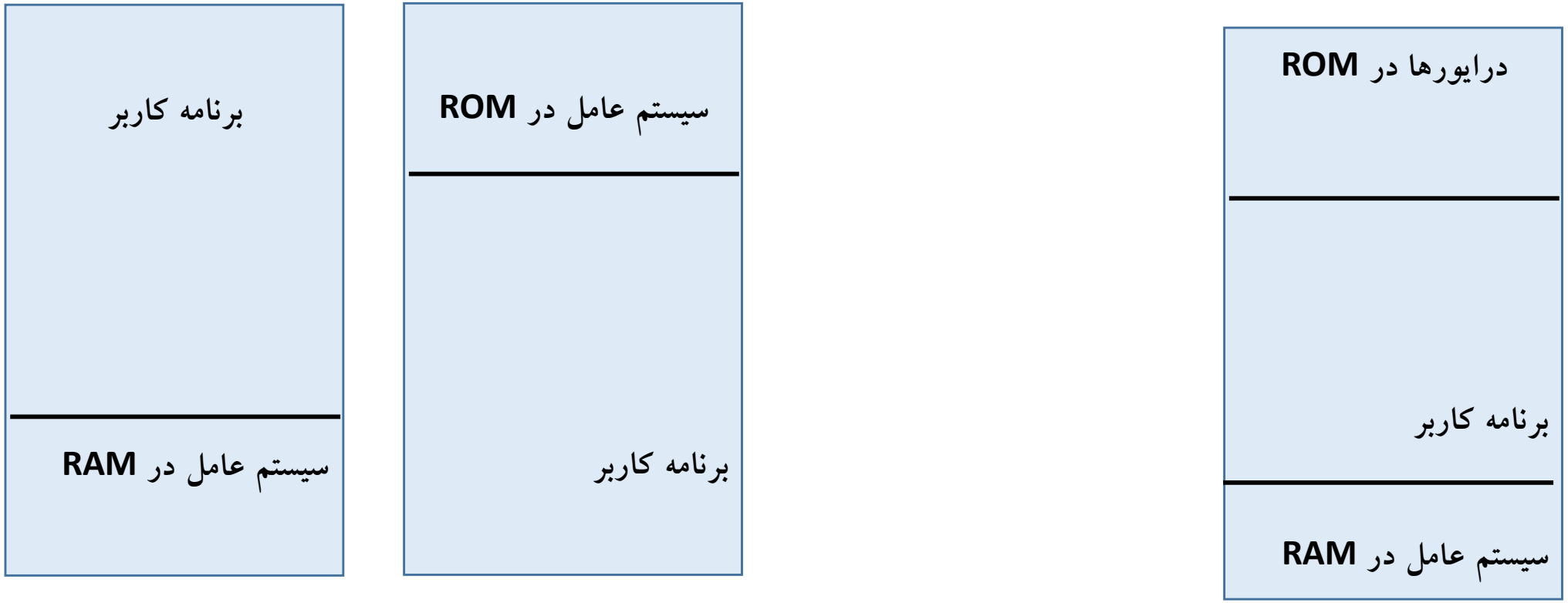
- رویکرد بدون جابجایی

۱). تک برنامگی بدون معاوضه (swapping)

۲). چندبرنامگی

تک برنامگی

- ساده‌ترین روش مدیریت حافظه است. در هر لحظه فقط یک برنامه اجازه اجرا شدن را داشته باشد : –سیستم عامل در پایین RAM قرار دارد. – سیستم عامل در بالای RAM قرار دارد.
  – بخشی از سیستم عامل در پایین و بخش دیگر در بالا قرار می گیرد.



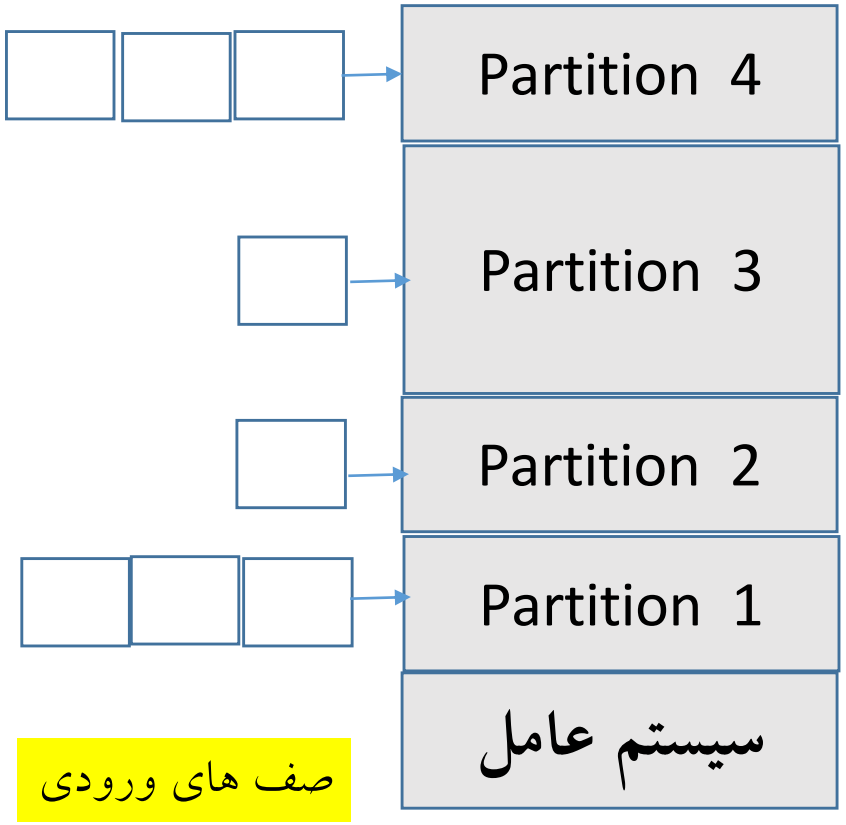
چندبرنامگی با بخش‌بندی ثابت

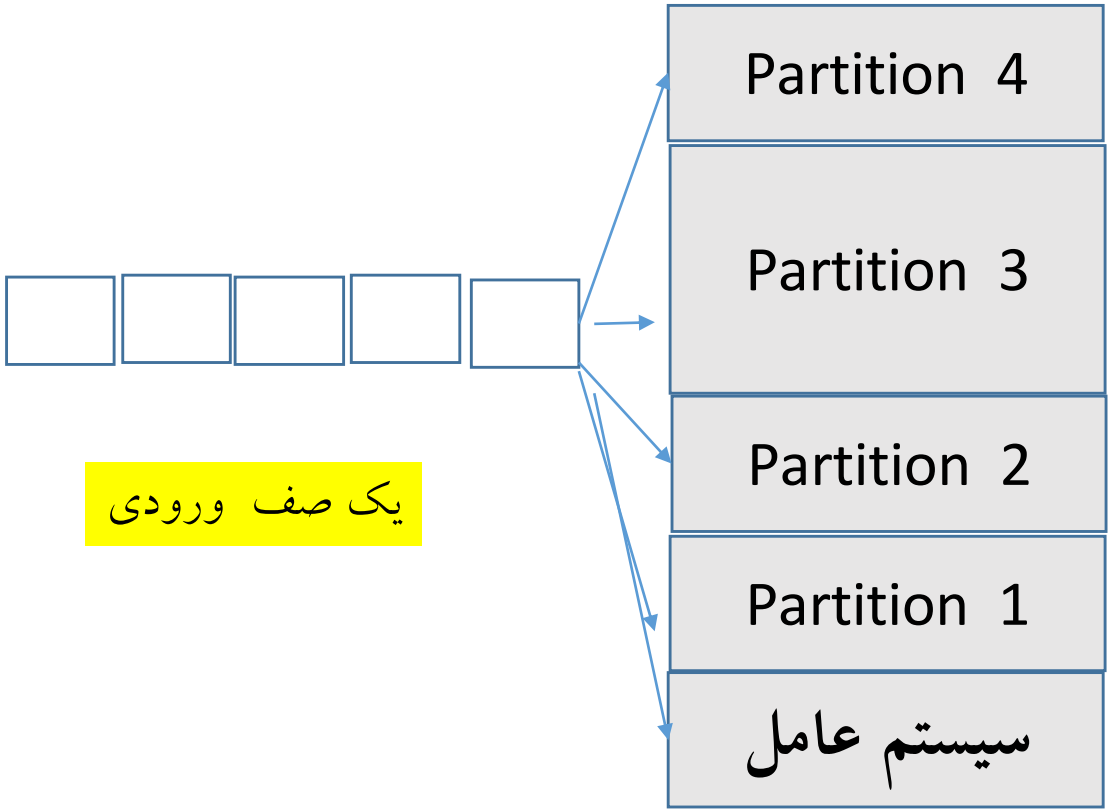
- همزمان چند برنامه در حافظه بارگذاری می‌شود. برنامه‌ها نباید با هم تداخلی داشته باشند.

- حافظه به n بخش (partition) تقسیم می‌شود. هر فرآیند در یکی از این بخش‌ها قرار می‌گیرد.
- وقتی یک فرآیند وارد حافظه می‌شود در یک صف قرار می‌گیرد تا در کوچکترین بخش که مناسب آن است قرار گیرد. البته ممکن است آن بخش دقیقا هم اندازه برنامه نباشد و بدین‌ترتیب مقداری از فضای حافظه هدر می‌رود.

برای هر بخش می‌توان از (۱) یک صف مجزا یا (۲) یک صف برای تمام بخش‌ها استفاده کرد.

چندبرنامگی با بخش‌بندی ثابت



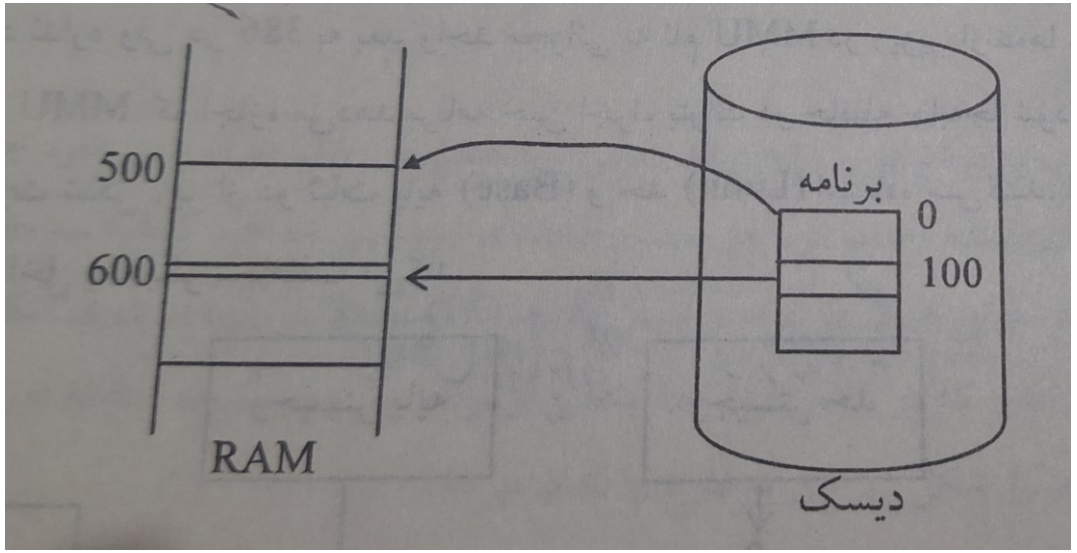


چندبرنامگی با بخش‌بندی ثابت

- مبتنی بر صف ورودی مستقل
- هر فرآیند جدیدی که وارد می شود در صف کوچکترین بخشی که در آن جا می گیرد، قرار داده می شود.
- ممکن است فضایی در هر بخش بیکار و بدون استفاده باقی بماند.
- صف ورودی بخشی از بخش ها، می تواند بسیار شلوغ باشد و صف های دیگر بکلی خالی باشد.
- مبتنی بر یک صف ورودی مشترک
- وقتی یک بخش خالی می‌شود ، نزدیکترین فرآیند به ابتدای صف که در این بخش جا می‌شود، انتخاب شده و در بخش حافظه بارگذاری می‌شود.

چند برنامگی

- در چندبرنامگی دو مساله وجود دارد:
  - جابجایی (Relocation)
  - حفاظت (Protection)
- هنگام ورود یک فرآیند به بخش حافظه، آن فرآیند می‌تواند در هر بخشی از حافظه اصلی قرار گیرد. مثلا برنامه شامل متغیری باشد که در آدرس ۱۰۰ از ابتدای برنامه قرار گرفته باشد. ممکن است در حافظه در آدرسی مشابه تصویر زیر قرار گیرد.
- اگر آدرس تولید شده در دستور پرشی در یک برنامه، سبب دسترسی به بخشی از یک فرآیند دیگر در حافظه شود، مساله حفاظت رخ می دهد.

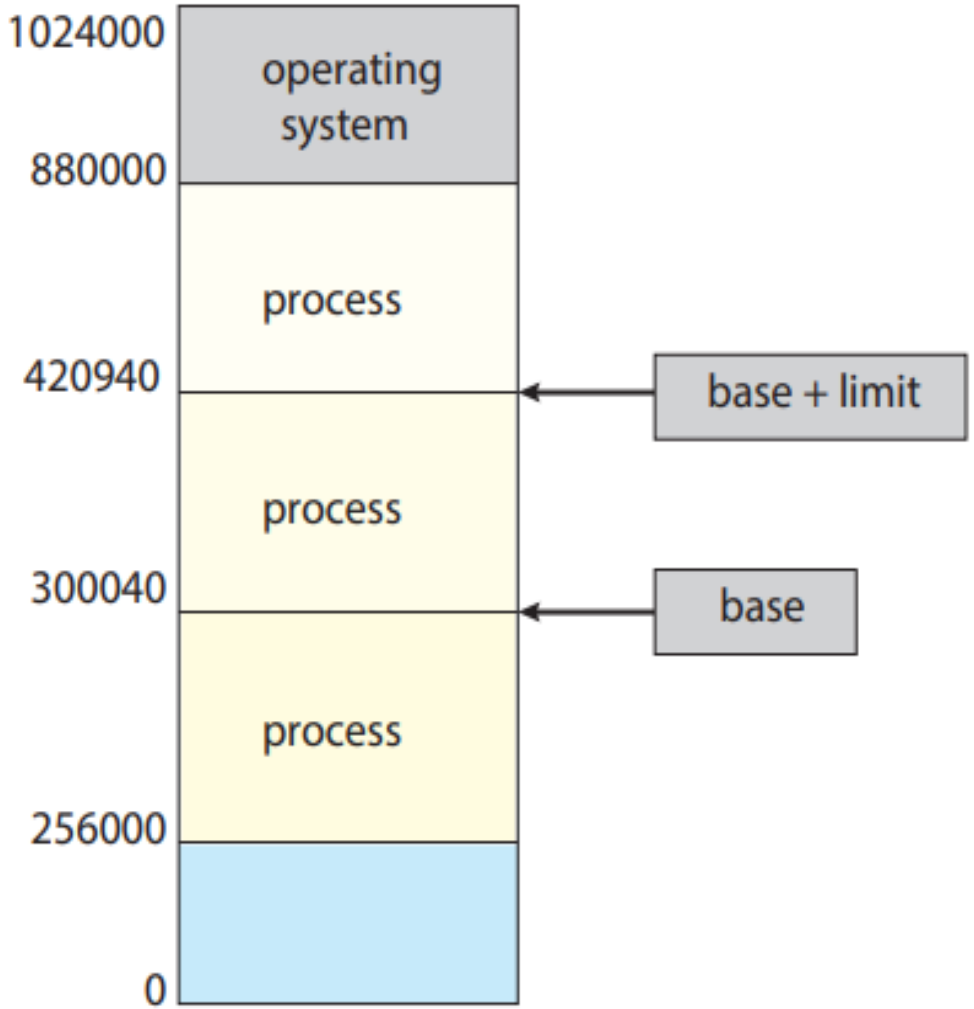


چند برنامگی

- برای جابجایی و حفاظت، دو ثبات پایه (base) و حد (limit) در نظر گرفته می‌شود.

- با بارگذاری فرآیند در حافظه، آدرس شروع بخش حافظه در ثبات پایه قرار می‌گیرد.
- اندازه بخش حافظه در ثبات حد قرار می‌گیرد.
- برای محاسبه آدرس واقعی حافظه، سیستم‌عامل ثبات مبنا را به تمام ارجاعات برنامه به حافظه جمع می‌کند.

حافظه



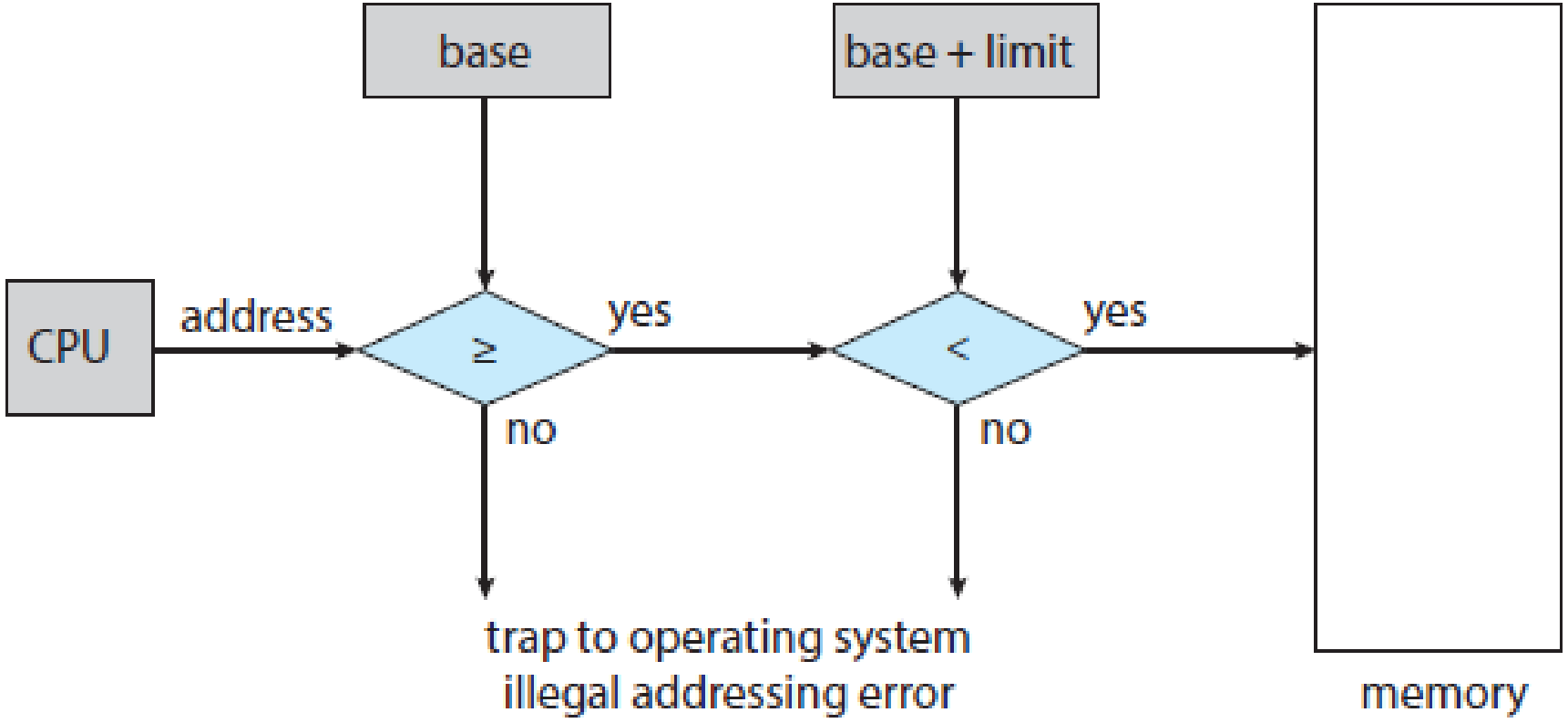
- برای فرآیند مورد نظر، ثبات پایه برابر با ۳۰۰۰۴۰ و ثبات حد برابر با ۱۲۰۹۰۰ باشد، فرآیند می‌تواند آدرسهایی مطابق با این آدرس‌ها را ایجاد نماید.
- [۳۰۰۰۴۰, .., ۴۲۰۹۴۰]

فضای آدرس فیزیکی و منطقی

- آدرس تولید شده توسط CPU را آدرس منطقی می گویند.

- آدرس مربوط به واحد حافظه، که در رجیستر آدرس حافظه قرار می گیرد را آدرس فیزیکی می نامند.

فضای آدرس فیزیکی و منطقی



برای جلوگیری از مشکل جابجایی، هر آدرس حافظه‌ای که تولید می‌شود، قبل از ارسال به حافظه مقدارش به صورت خودکار با محتوای ثبات پایه (Base) جمع می‌شود.

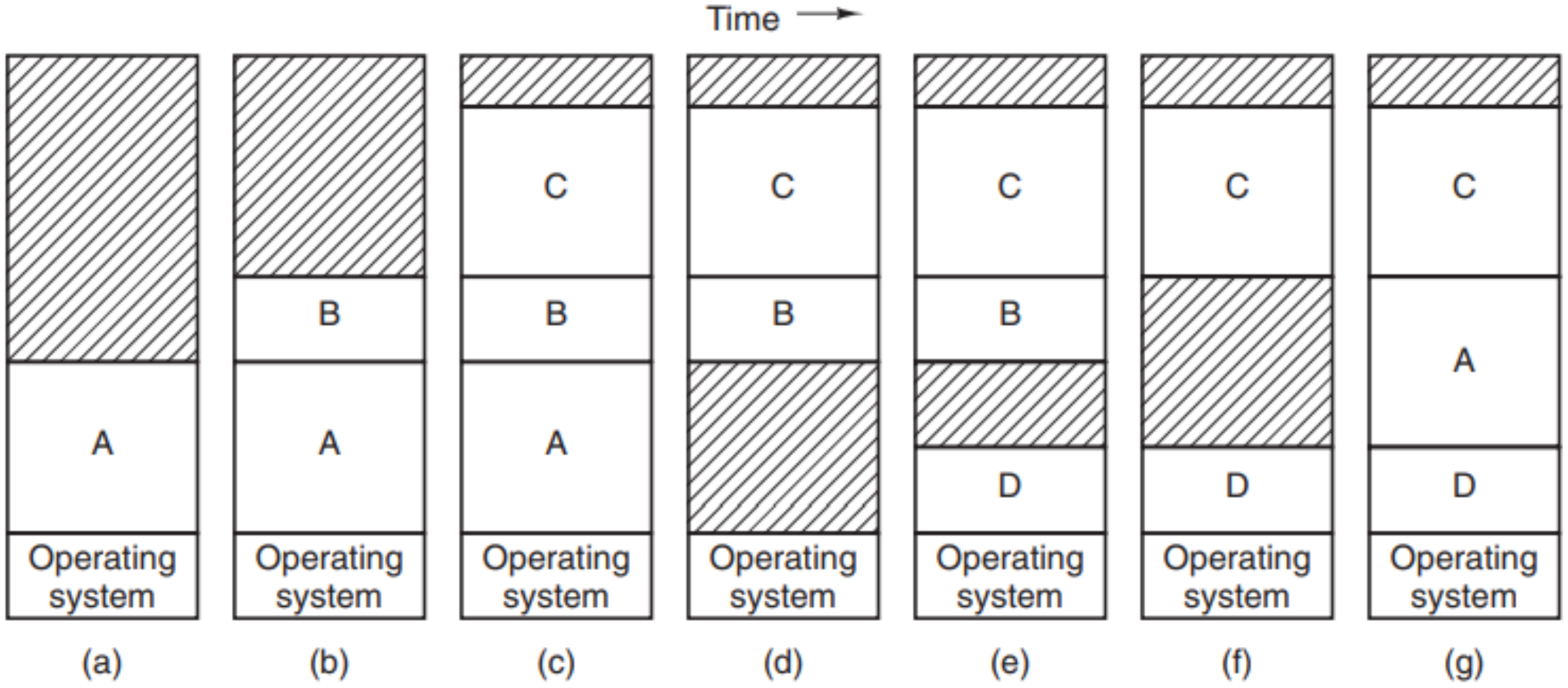
برای حفاظت، آدرس‌های تولیدشده در برنامه، با ثبات Limit مقایسه می‌شوند که مجوز استفاده از فضای خارج از بخش حافظه (Partition) را نداشته باشند.

مبادله (Swapping)

- در سیستم‌های اشتراک زمانی، ممکن است در سیستم حافظه کافی برای نگهداری تمامی فرآیندهای فعال موجود نباشد. بنابراین در مدیریت حافظه، از روش (۱) مبادله و (۲) حافظه مجازی استفاده می‌شود.

- **مبادله**
  - هر فرآیند بطور کامل به حافظه منتقل شده، اجرا می شود و بعد دوباره به دیسک برگردانده می‌شود.
- **حافظه مجازی**
  - در این روش، به فرآیندها اجازه داده می‌شود که فقط بخشی از آنها در حافظه بارگذاری شده و اجرا شوند.

مبادله


Time
A
Operating system
(a)
B
A
Operating system
(b)
C
B
A
Operating system
(c)
C
B
Operating system
(d)
C
B
D
Operating system
(e)
C
D
Operating system
(f)
C
A
D
Operating system
(g)

مبادله

- بر اثر عملیات مبادله، ممکن است حفره های متعددی در حافظه بوجود آید.
- مجموع فضاهای آزاد (حفره ها) مقدار زیادی است اما به دلیل پراکندگی حفره‌ها و اینکه همجوار نیستند، نمی‌توان استفاده بهینه‌ای از آنها داشت و پردازش جدیدی در آن بارگذاری کرد. این مساله را **External Fragmentation (تکه‌تکه شدن- پارگی خارجی)** می‌گویند.

- از معایب مبادله، ایجاد حفره‌های متعدد در حافظه است.
- **حفره** به مناطق کوچکی از حافظه گویند که خالی و بدون استفاده مانده‌اند و نمی‌توان فرآیندی را در آن جا داد.

متراکم کردن حافظه

- برای برطرف کردن مشکل External fragmentation، می‌توان از ایده فشرده‌سازی استفاده کرد. در این روش، حفره‌های کوچک با یکدیگر ادغام‌شده و حفره یکپارچه بزرگی ایجاد می‌شود. در واقع فرآیندها تا جایی که امکان دارد، به آدرس‌های پایین‌تر حافظه جابجا می‌شوند.

- این روش بسیار وقت CPU را می‌گیرد.

- در روش بخش‌بندی ثابت، فضاهایی از حافظه (حفره‌ها) که بدون استفاده می‌ماندند و قابل استفاده برای بارگذاری فرآیندهای دیگر نیستند را تکه‌تکه شدن (پارگی) داخلی (Internal fragmentation) گویند.
- در روش دینامیک، در مقایسه با روش ثبت (ایستا) استفاده بهتری از حافظه می‌شود. اما مدیریت حافظه پیچیده تر است.

مدیریت حافظه در روش دینامیک (مبادله)

- مدیر حافظه باید در هر لحظه از زمان از قسمت های آزاد و استفاده شده حافظه مطلع باشد.

- روش های مدیریت حافظه آزاد و استفاده شده

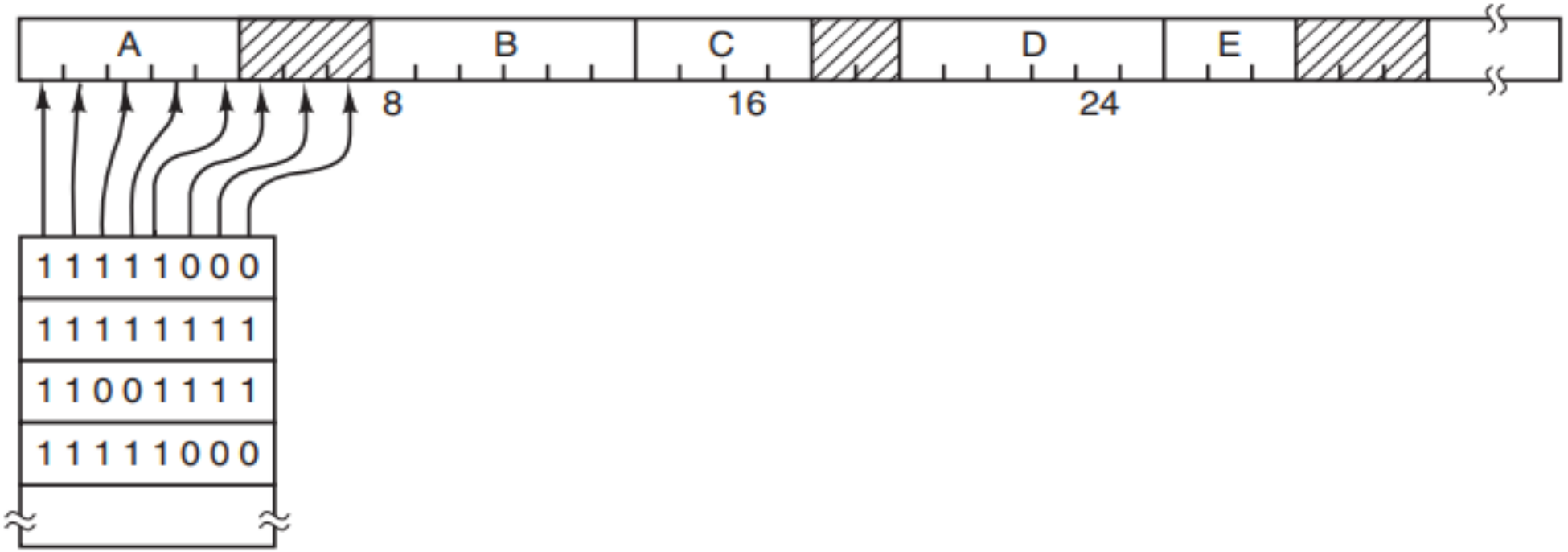
1. مدیریت حافظه با نقشه بیتی (Bit-map) (**نگاشت بیتی**)
2. مدیریت حافظه با لیست پیوندی

مدیریت حافظه با نقشه بیتی

- حافظه به تعدادی واحد تخصیص تقسیم می‌شود.
- متناظر با هر واحد تخصیص یک بیت در نقشه بیتی وجود دارد.
- بیت استفاده شده، متناظر با **یک** و استفاده نشده، متناظر با **صفر** مقداردهی می‌شود.

- **ویژگی ها**
- پیاده‌سازی ساده
- سرعت پایین

مدیریت حافظه با نقشه بیتی

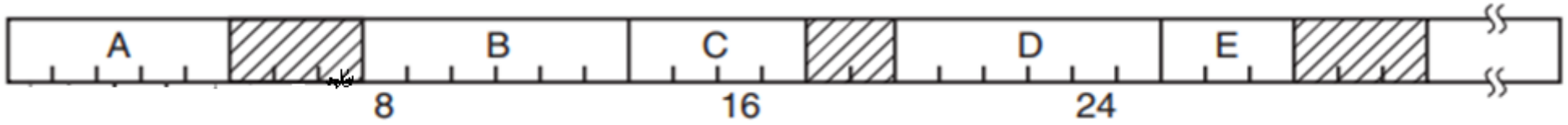


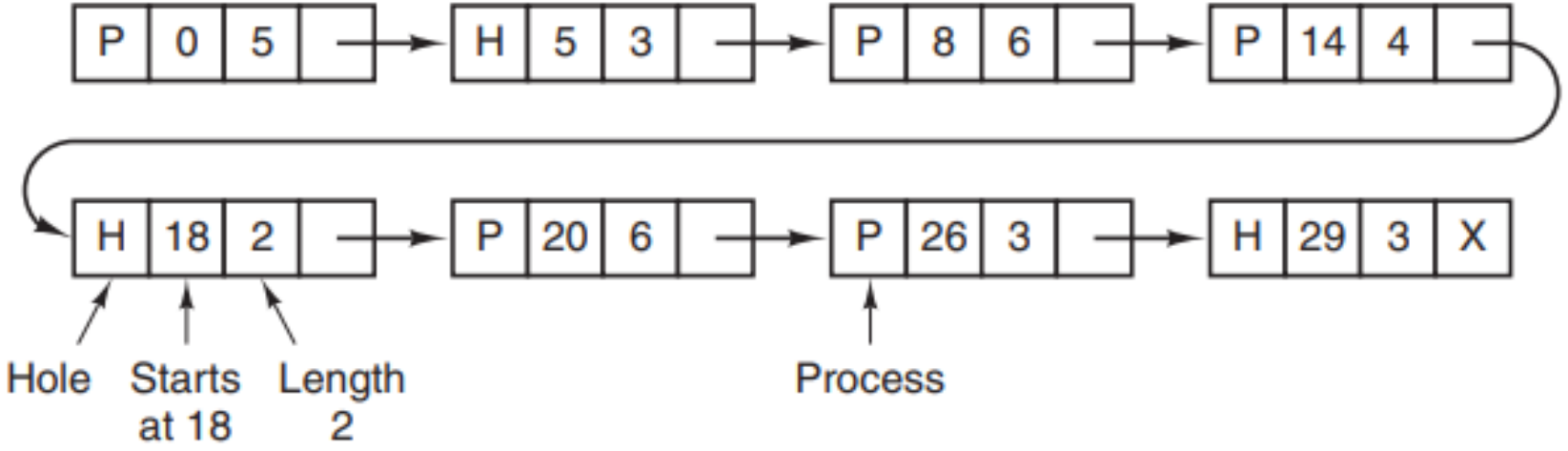
مدیریت حافظه با لیست پیوندی

- لیست پیوندی از قطعه‌های آزاد و تخصیص یافته تشکیل شده است.
- هر گره در لیست پیوندی معادل فرآیند یا حفره است.
- حفره با H نشان داده شده است.
- فرآیند را با P نشان می‌دهند.

- **ویژگی‌ها**
- هنگام مبادله فرآیند یا اتمام آن، به روزرسانی لیست پیوندی به سادگی انجام می‌شود.

مدیریت حافظه با لیست پیوندی





روش‌های تخصیص حافظه در مدل همجواری (Contiguous)

- اولین برازش (First fit)
- برازش بعدی (Next fit)

- بهترین برازش (Best fit)
- بدترین برازش (worst fit)

- برازش سریع (Quick fit)
- رفاقتی (Buddy algorithm)

روش‌های تخصیص حافظه

- **اولین برازش (First fit)**
- سیستم عامل از ابتدای حافظه لیست فضاهای خالی حافظه را بررسی می کند و فرآیند در اولین حفره ای قرار داده می شود که در آن جا شود.
- **برازش بعدی (Next fit)**
  - مشابه روش First fit است. با این تفاوت که جستجو از آخرین محل تخصیص شروع می شود.
- **بهترین برازش (Best fit)**
  - تمام لیست جستجو می شود و فرآیند در کوچکترین حفره ای قرار داده می شود که در آن جا می شود.
- **بدترین برازش (worst fit)**
  - تمام لیست جستجو می شود و فرآیند در بزرگترین حفره ای قرار داده می شود که در آن جا می شود.

روش‌های تخصیص حافظه (برازش سریع)

- **برازش سریع (Quick fit)**
- برای هر دسته از فرآیندها با اندازه‌های متداول یک لیست جداگانه در نظر گرفته می‌شود.
- جدولی با n خانه است که:
  - خانه اول اشاره گری به ابتدای لیستی شامل حفره‌های ٤ کیلوبایتی آزاد است.
  - خانه دوم به لیست حفره های ٨ کیلوبایتی و ... اشاره می کند.
- برای درخواست ورودی طبق جدول اندازه های متداول، مکانی درنظر گرفته می شود یا از فضای آزاد عمومی طبق روش های دیگر، مکانی برای جایگذاری درخواست فرض می‌شود.

- امتیاز: پیدا کردن حفره ای با اندازه مناسب بسیار سریع است.
- ایراد: با خاتمه فرآیند باید فضای آزاد شده آن به لیست مناسبی اضافه شود که این کار زمان‌بر است.

مثال

- فرض کنید فضاهای مبتنی بر اندازه های رایج 8K، 16K و 32K است.
- فضای آزاد عمومی حافظه شامل اندازه حافظه های بزرگتر است.
- با فرض لیست آزاد زیر و درخواست های ورودی از راست به چپ 12K، 8K و 20K، با روش Quick fit مساله را حل نمایید. همچنین x، y و z آدرس های حافظه هستند.

- 8K: [x ,y]
- 16K: [z]
- 32K:[]

مثال

- برای 12K از فضای آزاد z استفاده می کنیم. مقدار 4K هدر می رود.
- برای 8K از فضای آزاد x استفاده می کنیم.
- لیست آزاد حافظه

- 8K:[y]
- 16K: []
- 32K:[]

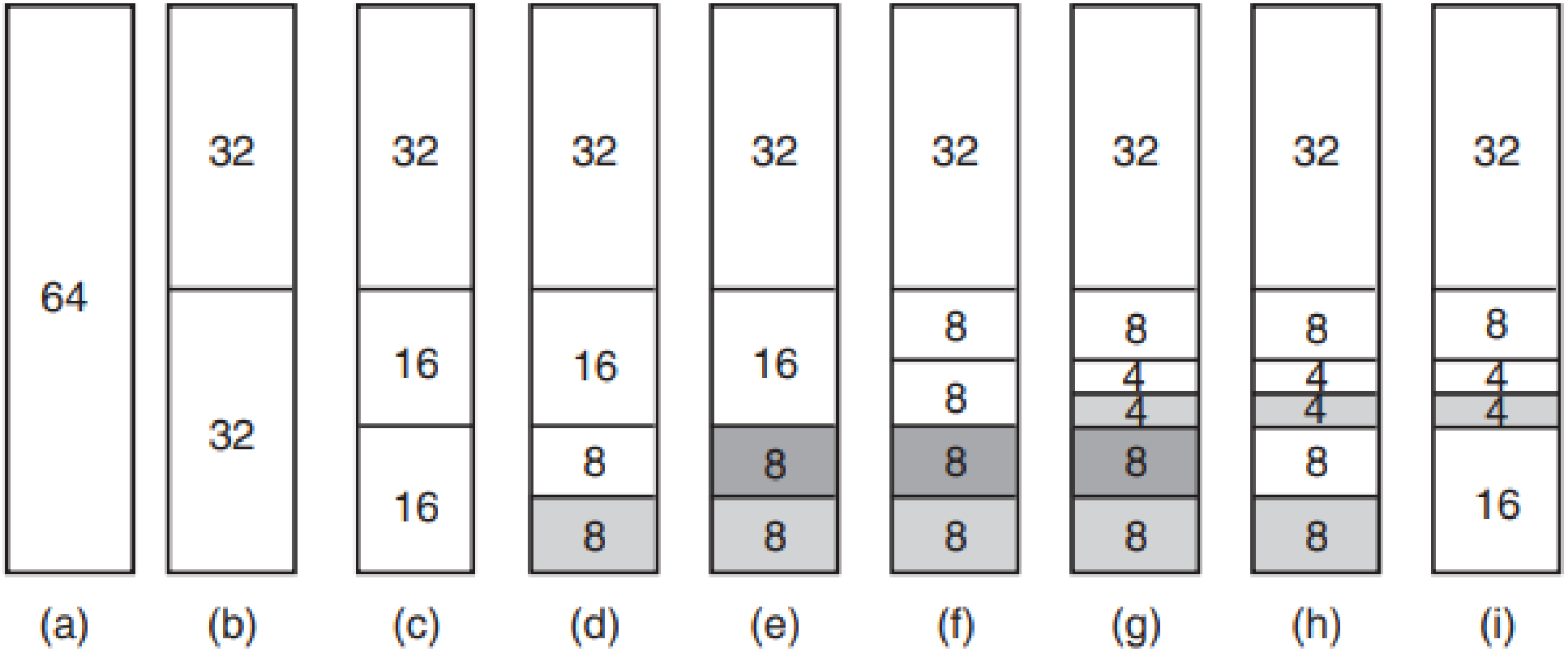
- برای 20K فضای آزاد مناسبی از لیست رایج وجود ندارد. از فضای عمومی حافظه باید استفاده کرد.

روش‌های تخصیص حافظه (سیستم رفاقتی)

- **سیستم رفاقتی (Buddy system)**
- اندازه بلاک‌های حافظه توانی از ۲ است.
  - اگر اندازه یک حفره برابر 2^K باشد و فرآیندی به اندازه S باید به داخل آن مبادله شود، اگر S از نصف اندازه حفره بزرگتر باشد، کل فضا به آن داده می‌شود، در غیر این صورت، کل بلوک نصف شده و دو بلوک رفاقتی ایجاد می‌شود.

- این شرایط به صورت بازگشتی اجرا می‌شود
- در صورت آزاد شدن یک حفره، امکان ترکیب رفقای مجاور وجود دارد.
- ایراد: اتلاف حافظه

الگوریتم رفاقتی



مثال: درخواست های وارد شده 8K، 8K، 4K

- درخواست A: 8K کل حافظه: 64K
  - فضای حافظه به ۲ قسمت مساوی تقسیم می‌شود. (32K, 32K) (تصویر b)
  - فضای 32K به دو قسمت تقسیم می شود (16K, 16K) (تصویر c)
  - فضای 16K به دو قسمت تقسیم می شود (8K, 8K).
    - درخواست در آن فضا قرار می‌گیرد. (تصویر d)
- درخواست B: 8K
  - در بخش ۸ تایی قرار داده می‌شود. (تصویر e)
- درخواست C: 4K
  - در تصویر f، فضای ١٦ به دو بخش تقسیم می‌شود. (8K, 8K)
  - در تصویر g، فضای ۸ به دو ٤تایی تقسیم شده و درخواست وارده را در آن جا می‌دهند.
- پایان درخواست B– آزادسازی (تصویر h)
- پایان درخواست A– آزادسازی و تجمیع دو فضا (رفاقت) (تصویر i)

بررسی روش‌های تخصیص همجوار

- در روش اولین برازش
  - در ابتدای لیست حافظه، تراکم فضای اشغال شده، زیاد است.
- در روش برازش بعدی،
  - توزیع برنامه ها در حافظه یکنواخت است.
- در بهترین برازش،
  - فضاهای بزرگتر برای درخواست های بزرگتر محفوظ هستند.
  - احتمال حفره های کوچک بدون استفاده زیاد می شود.
- در بدترین برازش،
  - ممکن است درخواست های بزرگ، برآورده نشود.
  - احتمال استفاده از حفره های آزاد بیشتر است.

بررسی روش‌های تخصیص همجوار

- از نظر سرعت و بهره وری حافظه، اولین برازش و بهترین برازش، بهتر از بقیه روش ها هستند.
- روش اولین برازش سریع تر از بهترین برازش است.
- کارایی برازش بعدی کمی کمتر از روش برازش اول است.

صفحه‌بندی (Paging)

- یکی از مشکلات تخصیص حافظه همجوار، تکه تکه شدن خارجی است. راه حل رویکرد صفحه‌بندی است.

- **صفحه‌بندی**
- یک روش تخصیص غیرهمجوار است.
- حافظه فیزیکی به اندازههای ثابتی به نام قاب (Frame) تقسیم می‌شود.
- حافظه منطقی برای فرآیندها به تعدادی صفحه به اندازه یکسان تقسیم می‌شود.
- تقسیم بندی حافظه مبتنی بر سخت‌افزار است.
- یک فرآیند شامل تعدادی صفحه است که در قاب‌های آزاد حافظه اصلی قرار می‌گیرد. بنابراین قاب‌های متعلق به یک فرآیند می‌توانند مجاور هم نباشند.

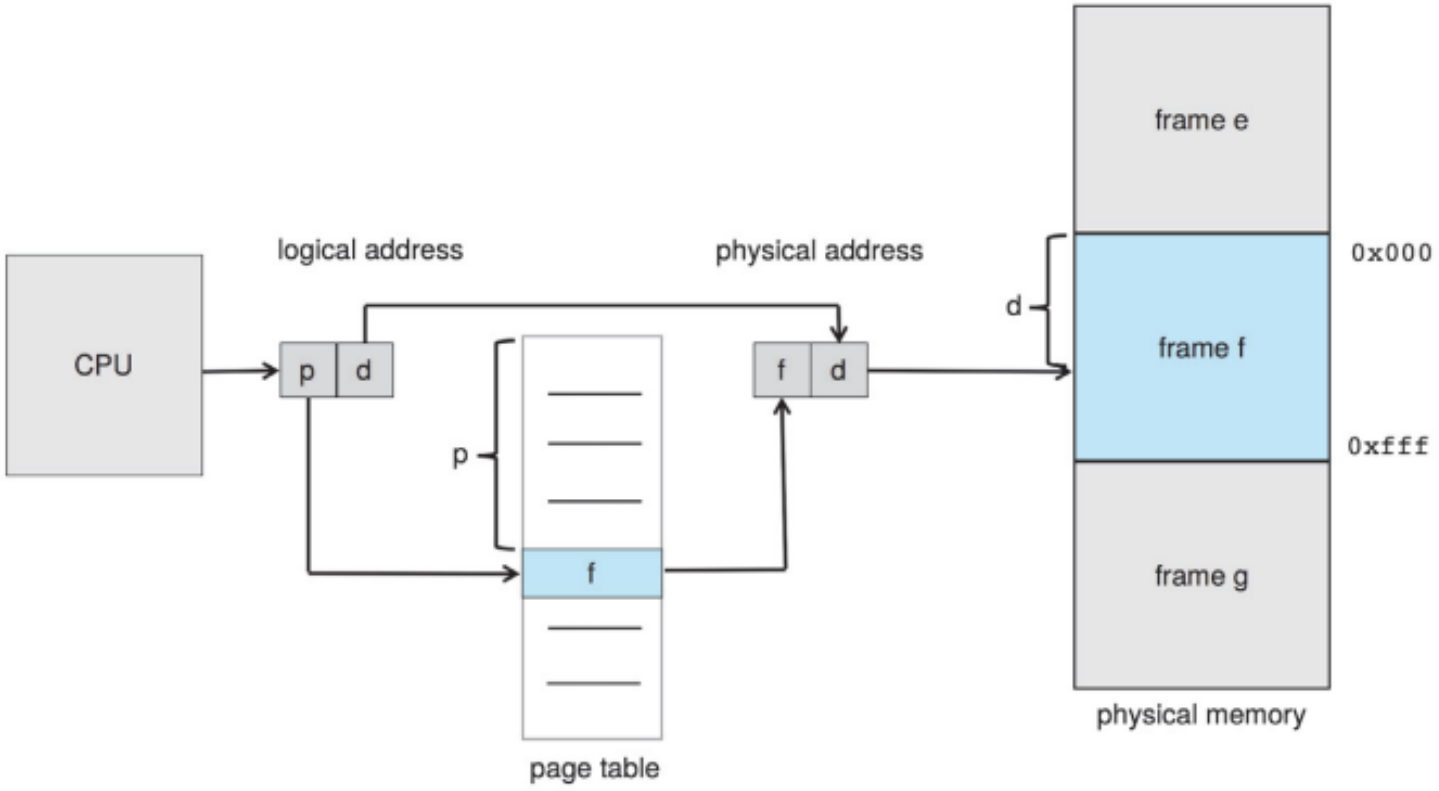
صفحه‌بندی (Paging)

- مولفه‌های آدرس برگرفته از CPU
  - شماره صفحه
  - آفست

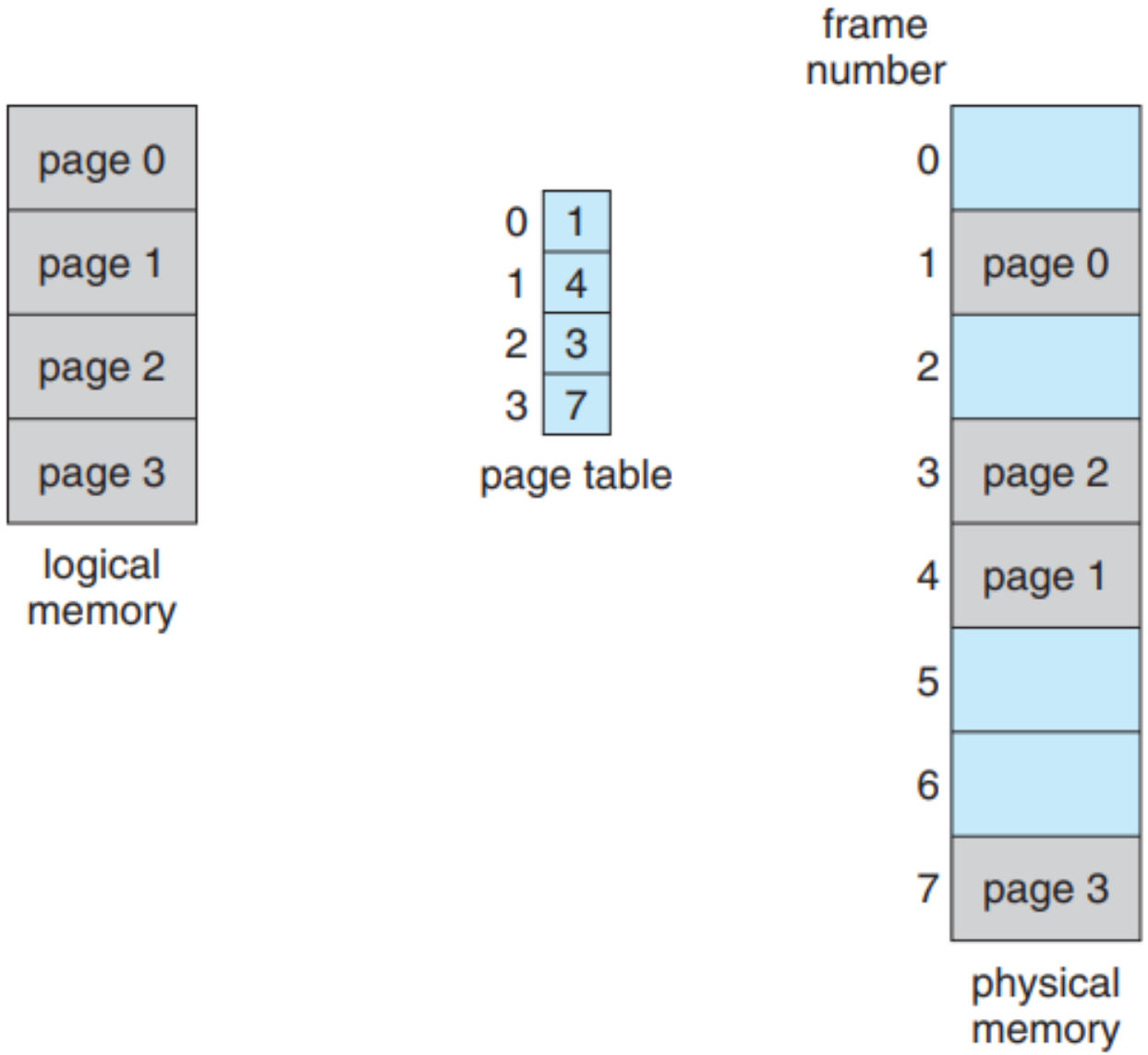
page number	page offset
p	d

- شماره صفحه، اشاره‌گر به **جدول صفحه** است.
- جدول صفحه شامل آدرس مبنای هر فریم در حافظه فیزیکی (RAM) است.
- مولفه های آدرس حافظه فیزیکی
  - شماره فریم (قاب)
  - آفست

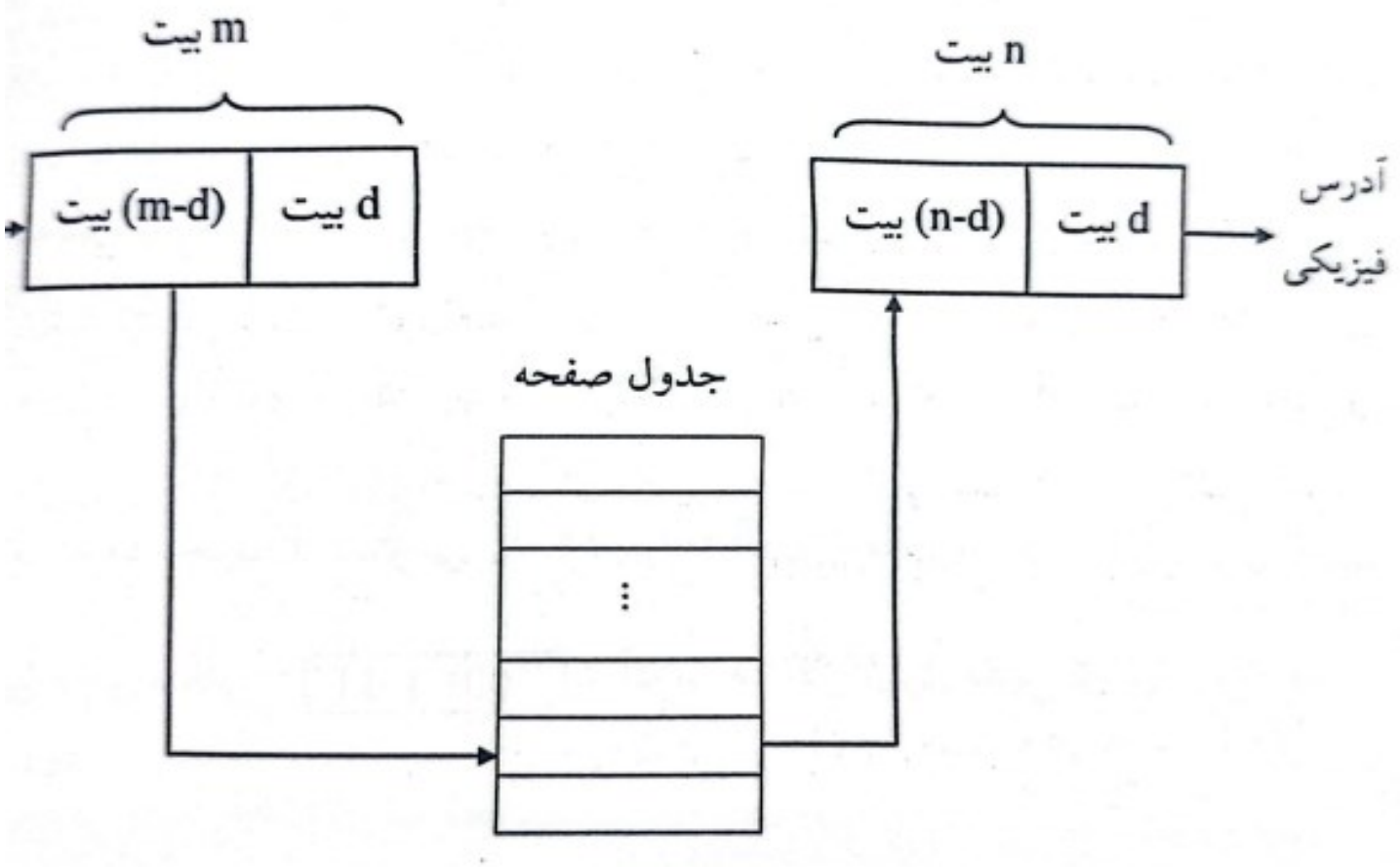
صفحه‌بندی (Paging)



صفحه‌بندی (Paging)

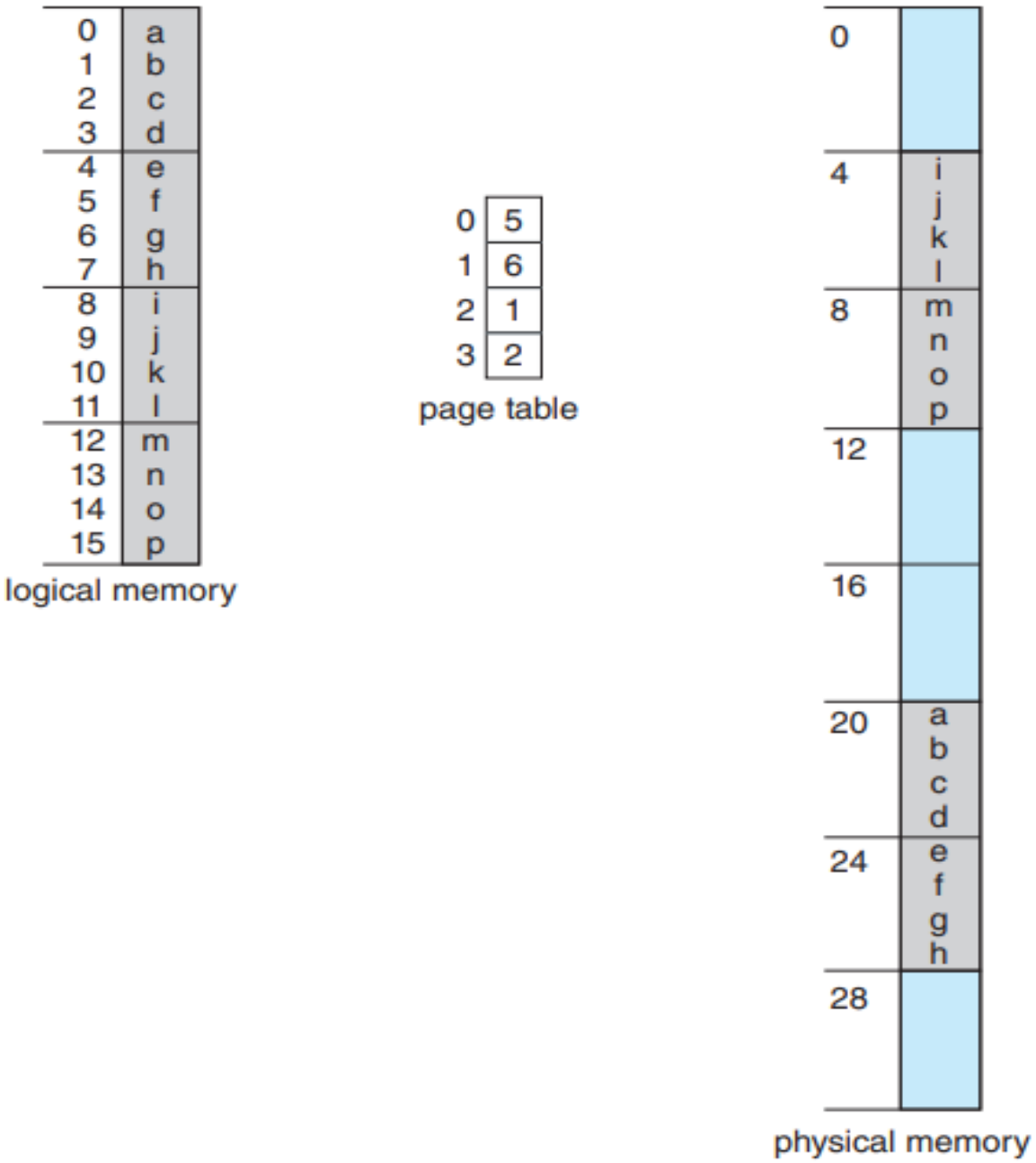


صفحه‌بندی (Paging)



- پارامترها
- اندازه حافظه فیزیکی: 2^n
- تعداد فریم ها: 2^{n-d}
- اندازه حافظه منطقی: 2^m
- اندازه قاب(صفحه): 2^d
- تعداد سطر جدول صفحه: 2^{m-d}

مثال



حافظه منطقی برنامه شامل ١٦ خانه حافظه است. ١٦ خانه به ٤ صفحه تقسیم شده است.پس هر صفحه شامل ٤ کلمه است.

پس جدول صفحه ٤ سطر دارد.

پس حافظه اصلی شامل قاب به اندازه ٤ کلمه است.

تعدادقاب ها ٨ تا است.

آدرس منطقی: 1011

شماره صفحه معادل ٢ است (١٠)

آفست معادل ١١ است.

آدرس فیزیکی معادل: 00111

مثال

- یک فضای آدرس منطقی صفحه‌بندی، شامل ۳۲ صفحه ۲ کیلوبایتی را که به یک فضای آدرس دو مگابایتی نگاشت شده است، در نظر گرفته، هر مدخل جدول صفحه باید چند بیتی باشد؟
- پاسخ

$32= 2^5$, $2KB=2\times2^{10}$, offset = 11 bit, page = 5 bit➔

Logical Address = 11+5 = 16

$2MB = 2\times2^{20}$, Physical Address = 21 bit→ frame+offset = 21➔ frame = 21-11=10

بیت‌های جدول صفحه

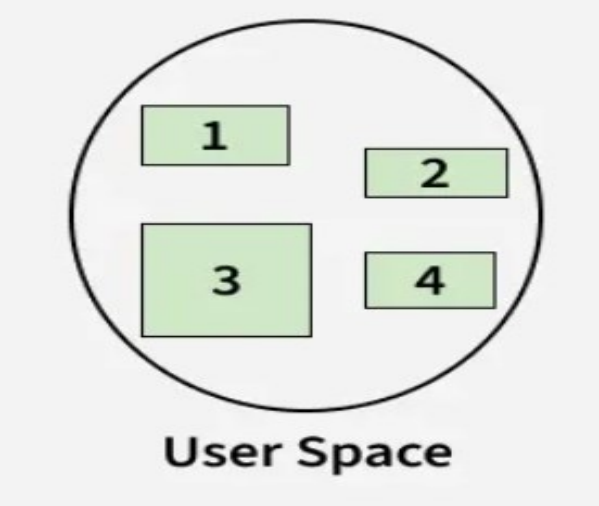
- در جدول صفحه، علاوه بر داده‌های پیش‌فرض، معمولا تعدادی بیت کنترلی اضافه وجود دارد.

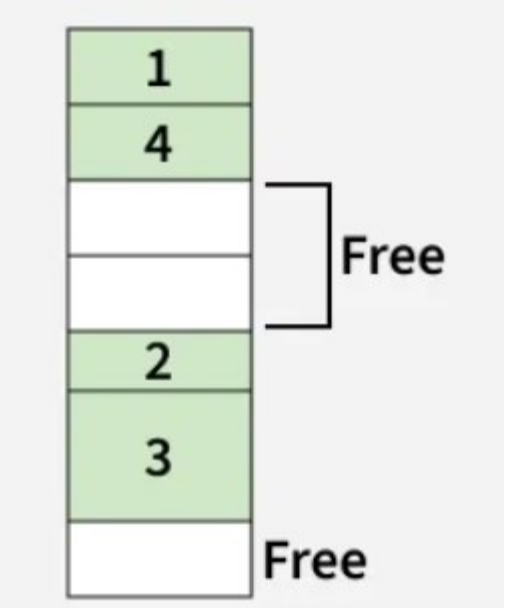
1. بیت خواندن، بیت خواندن/نوشتن،
2. بیت معتبر v و نامعتبر i ، در صورت معتبر بودن صفحه نظیر آن در فضای آدرس منطقی فرآیند قرار دارد.
3. Dirty bit: اگر پردازنده چیزی در صفحه نوشته باشد، یک است.
4. بیت Accessed: صفحه اخیرا خوانده یا نوشته شده است.

قطعه‌بندی (Segmentation)

- یکی دیگر از روش های مدیریت حافظه برای بهبود کارآمدی حافظه، قطعه‌بندی است.

- فضای آدرس برنامه کاربر به تعدای قطعه در اندازه‌های متغیر تقسیم می‌شود.
  - هر قطعه یک واحد منطقی از برنامه است. مانند:
  - قطعه کد، قطعه داده، قطعه پشته





فضای حافظه فیزیکی

قطعه‌بندی

- آدرس منطقی شامل دو مولفه است:

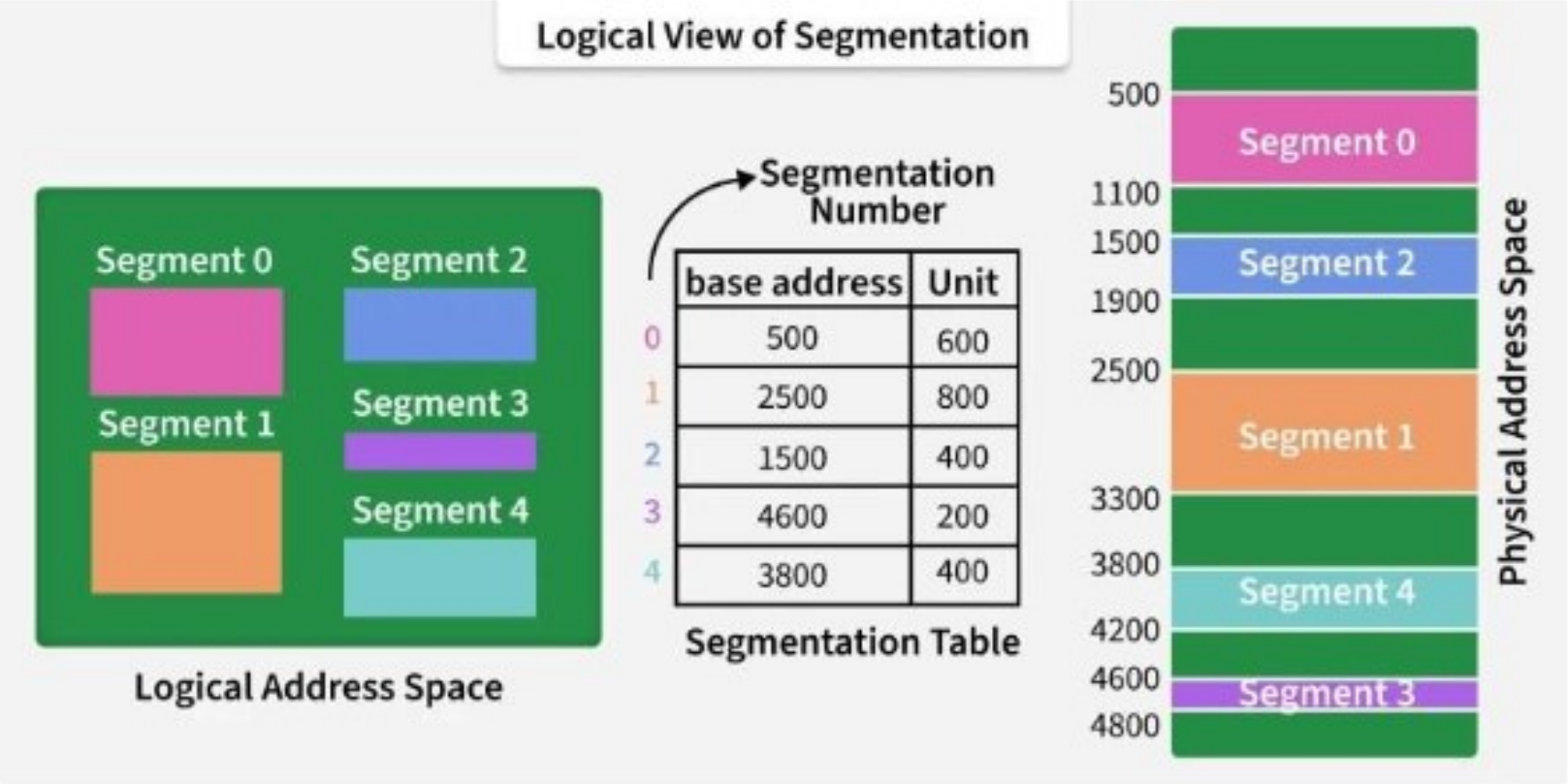
(۱). شماره قطعه (۲). آفست (فاصله)

- آدرس منطقی به آدرس فیزیکی تبدیل می شود. بدین منظور از جدول قطعه استفاده می شود.
- **جدول قطعه**

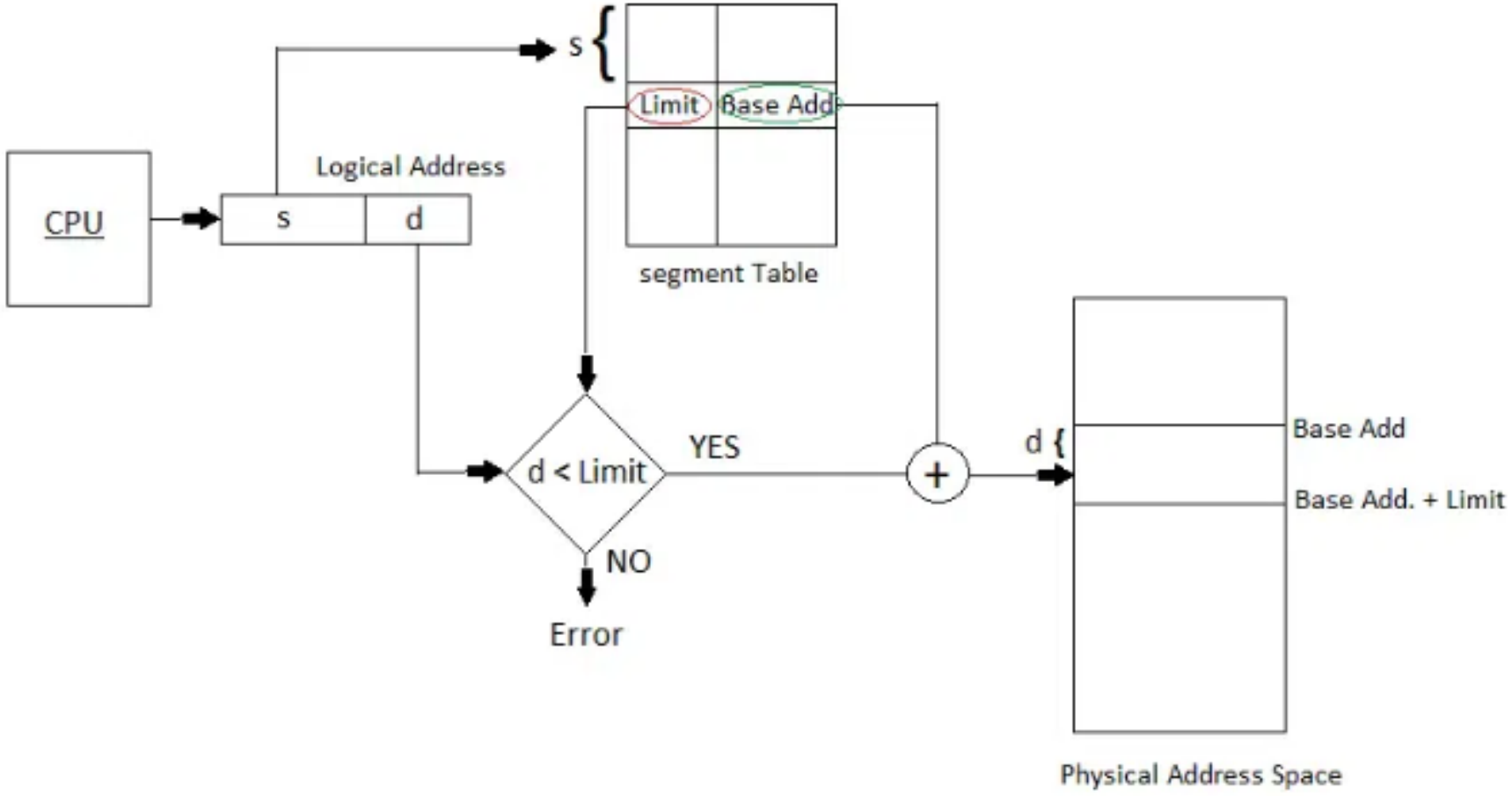
(۱) آدرس پایه قطعه (Base) (۲) طول قطعه (limit)

- آفست بین ۰ تا حد قطعه (limit) است.
- اگر آفست از طول قطعه (حد limit) بیشتر باشد، تله سخت افزاری رخ می دهد و سیستم عامل جلوی دستیابی غیرمجاز را می گیرد.
- **محاسبه آدرس فیزیکی: مجموع آفست و پایه**

قطعه‌بندی


Logical View of Segmentation
Segment 0
Segment 2
Segment 1
Segment 3
Segment 4
Logical Address Space
Segmentation Number
base address
Unit
0
500
600
1
2500
800
2
1500
400
3
4600
200
4
3800
400
Segmentation Table
500
1100
1500
1900
2500
3300
3800
4200
4600
4800
Segment 0
Segment 2
Segment 1
Segment 4
Segment 3
Physical Address Space

قطعه‌بندی


CPU
Logical Address
s
d
s
Limit
Base Add
segment Table
d < Limit
YES
NO
Error
+
d
Base Add
Base Add. + Limit
Physical Address Space

بیت های حفاظتی در قطعه

- در جدول قطعه و صفحه، علاوه بر داده‌های پیش فرض، معمولا تعدادی بیت کنترلی اضافه نیز وجود دارد. شامل:

- بیت خواندن، بیت خواندن/نوشتن،
- Accessed هرگاه پردازنده به سگمنت دسترسی پیدا کند این بیت یک می شود.
- بیت segment present: آیا قطعه در حافظه است یا در دیسک
- Privilege level: سطح اولویت اجرای برنامه را تعیین می کند.

مثال

- در یک سیستم حافظه صفحه بندی با یک جدول صفحه حاوی ٦٤ مدخل ١١ بیتی (شامل یک بیت اعتبار /عدم اعتبار) و صفحه های با اندازه هر یک ٥١٢ بایت، یک آدرس منطقی و یک آدرس فیزیکی چند بیت است؟
- **پاسخ**

Physical memory = $2^{10} \times 512 = 2^{19}$

Logical memory = $64 \times 512 = 2^{15}$

مثال

- در یک سیستم حافظه قطعه بندی ساده، جدول قطعه بدین صورت است:

Segment	Base	Length
0	1100	500
1	2500	1000
2	200	600
3	4000	1200

- کدامیک از آدرس های منطقی زیر، فاقد آدرس فیزیکی هستند؟

0, 300
2, 800
1, 600
3, 1100
1, 1111

d= 300, d<length, address=d+base=1400
d = 800, d<600, **trap**
d=600, 600<=1000, address=3100
d=1100, 1100<1200, address=5100
d= 1111, 1111<1000, **trap**

مقایسه صفحه‌بندی و قطعه‌بندی

- قطعه بندی
- امکان جداسازی منطقی اجزای برنامه از نظر عملکرد و استفاده اشتراکی و حفاظت بهتر
- تکه تکه شدن داخلی حذف شد.
- تکه تکه شدن خارجی دارد.

- صفحه بندی
- تکه تکه شدن داخلی دارد.
- تکه تکه شدن خارجی ندارد.
- داشتن فضای منطقی بزرگتر از فضای فیزیکی با توجه به دیدگاه حافظه مجازی

ذخیره‌سازی جدول صفحه

- سیستم عامل برای هر فرآیند یک جدول صفحه تخصیص می دهد.

- ذخیره کردن جدول صفحه مبتنی بر سخت افزار
  - روش بافر دم دستی (TLB) مبتنی بر حافظه سریع شرکت پذیر
  - ترکیبی از TLB و جدول صفحه

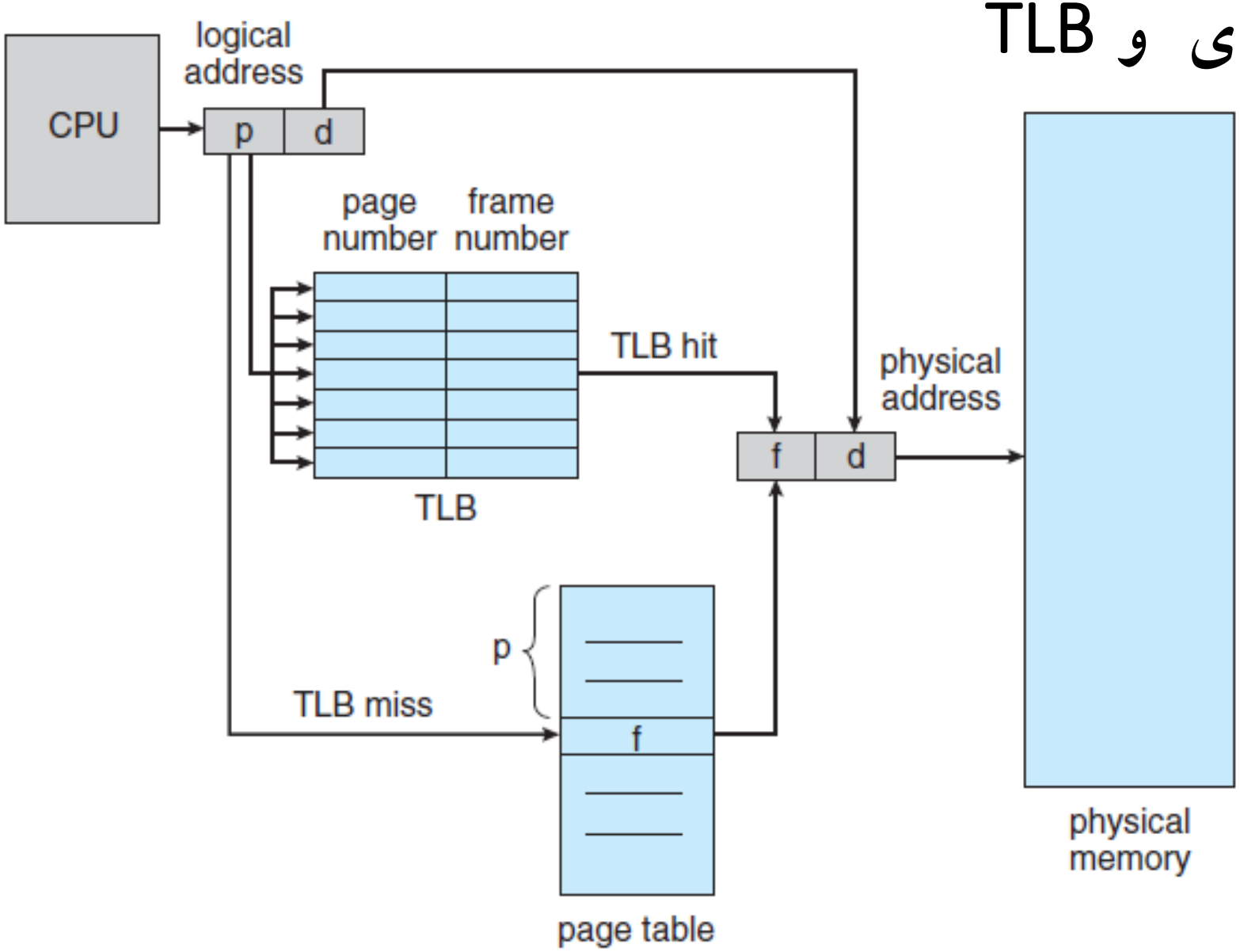
TLB =Translation Look-aside Buffer

روش بافر دم‌دستی ترجمه

- روش TLB یک حافظه سریع شرکت پذیر (انجمنی) associate است.
- از حافظه های با سرعت بالا ساخته شده است.
- هر سطر از دو بخش تشکیل شده است:
  - نشانه یا کلید (Tag) و داده یا مقدار (Data)
  - هنگام ورود اطلاعات به جدول، با همه TAG ها مقایسه می شود اگر برابر بود، داده یافت شده است و مقدار متناظر آن تعیین می گردد.
  - جستجو در این جدول بسیار سریع است.

- این نوع حافظه بسیار گران است. بنابراین از ترکیب آن با جدول حافظه معمولی نیز به عنوان کمکی استفاده می شود.

سخت‌افزار صفحه‌بندی و TLB



روش TLB

- جدول TLB، تعداد کمی از ورودی جدول صفحه را دارد.

- هنگامی که آدرس منطقی توسط CPU داده می‌شود، شماره صفحه در TLB جستجو می‌شود. اگر صفحه در TLB وجود داشته باشد، اصطلاحا دسترسی (برخورد) hit رخ داده است.

- اگر شماره صفحه در TLB نباشد، عدم‌دسترسی (عدم‌برخورد) miss رخ داده است.
  - در این شرایط به جدول صفحه مراجعه می شود و در آنجا جستجو انجام می شود.

روش TLB

- درصد تعداد دفعاتی که شماره صفحه در TLB یافت شود، را نسبت دسترسی hit rate گویند.

- درصد تعداد دفعاتی که شماره صفحه در TLB یافت نشود را نسبت عدم‌دسترسی miss rate گویند.
- زمان دستیابی به حافظه

$$hit \times (t_{tlb} + tm) + miss \times (t_{tlb} + 2 \times tm)$$

مثال

- در سیستم مدیریت حافظه، اگر زمان دسترسی به حافظه 400 ns و زمان دسترسی به جدول TLB برابر 50 ns و نسبت برخورد در جدول TLB برابر ۸۰٪ باشد، زمان دسترسی به حافظه را محاسبه نمایید.
- پاسخ

Hit rate= 80%, miss rate = 1- hit = 20%, tm = 400 ns, t_{tlb}=50ns

Time = 0.8×(50+400)+0.2 ×(50+2 ×400)= 530 ns

مثال

- در یک سیستم صفحه بندی، اگر زمان دستیابی به حافظه اصلی ٦٠ نانوثانه و زمان دستیابی به TLB برابر ٥ نانوثانیه باشد. با احتمال وجود ٧٥٪ شماره صفحه در TLB، نسبت بهبود تبدیل آدرس با وجود استفاده از TLB را تعیین نمایید.
- پاسخ

Access time (using TLB) = 0.75(5+60)+0.25(5+2× 60)= 48.75+31.25=80ns

Access time (without TLB) = 60× **2=120 ns**

نسبت بهبود 1.5 = 120/80

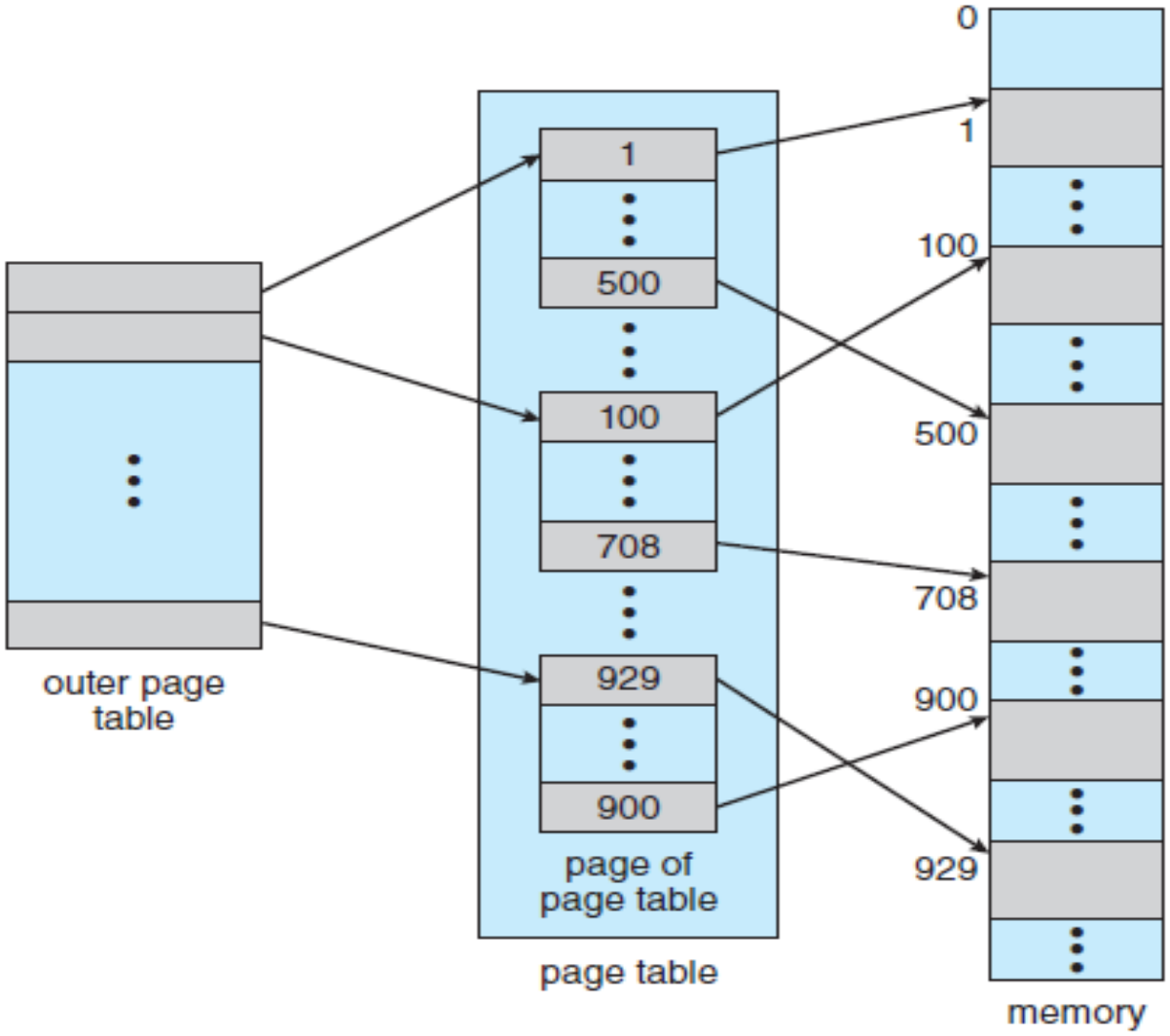
صفحه‌بندی چندسطحی

- در صورتی که فضای بزرگی از آدرس منطقی در دسترس باشد برای پیاده‌سازی جدول صفحه می‌توان از پیاده‌سازی چندسطحی استفاده کرد.
  - بنابراین با این روش، به ذخیره سازی دائمی تمامی جداول صفحه در حافظه نیازی نیست.

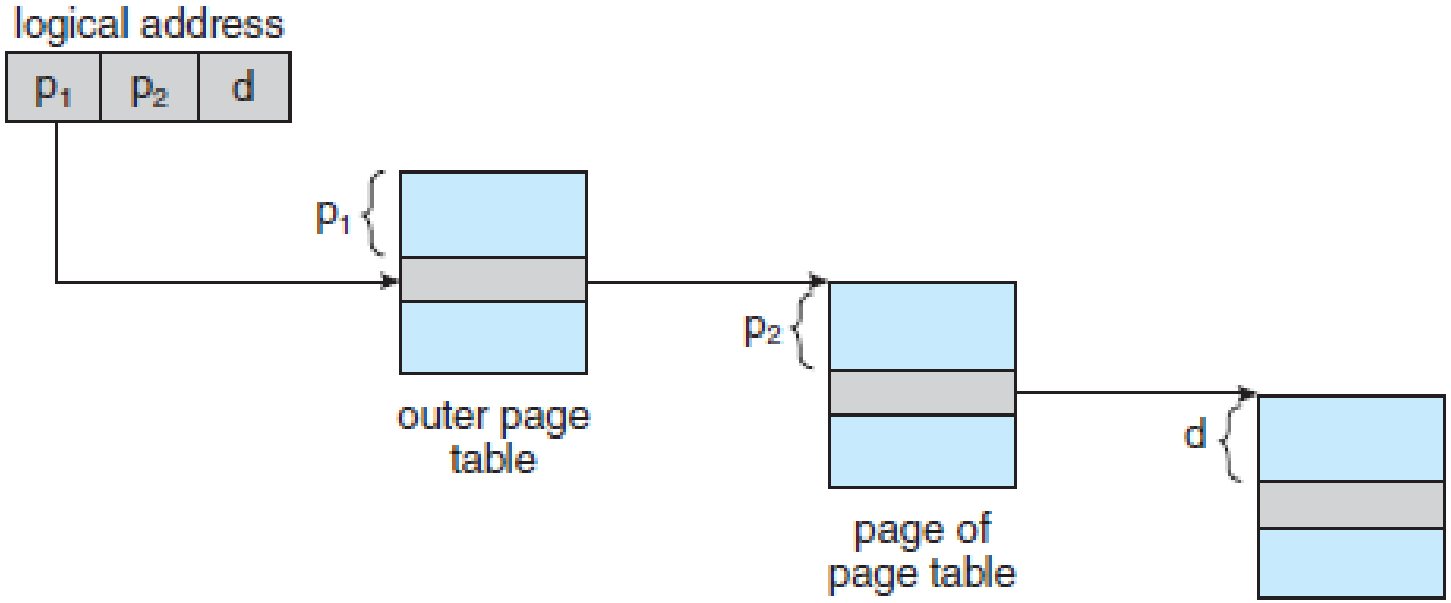
- جدول صفحه به چند تکه کوچکتر تقسیم می‌شود.

page number		page offset
p_1	p_2	d
10	10	12

صفحه‌بندی چندسطحی (دو سطحی)



صفحه‌بندی چندسطحی (دوسطحی)

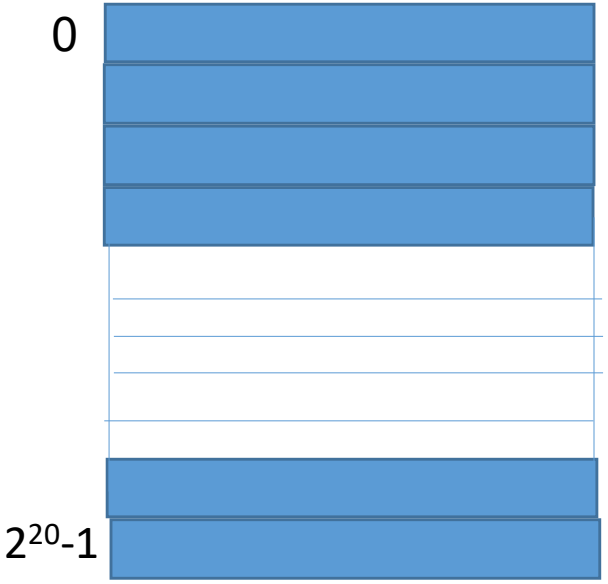


صفحه‌بندی چندسطحی (دوسطحی)

- **فرض کنید اندازه آدرس حافظه منطقی ۳۲ بیتی باشد. اندازه هر صفحه نیز 4KB است. روش ۲ سطحی و ۱ سطحی را مقایسه نمایید.**

روش ۱ سطحی:

- ۱۲ بیت برای آفست. ۲۰ بیت برای شماره صفحه.
- اندازه جدول صفحه $۲^{۲۰}$ است که برابر می شود با: ۱۰۴۸۵۷۶
- برای یک فرآیند به اندازه 12KB، به ۳ صفحه 4KB نیاز است.
- بنابراین در جدول صفحه به اندازه **۱۰۴۸۵۷۶** فقط ۳ تا صفحه نیاز است.

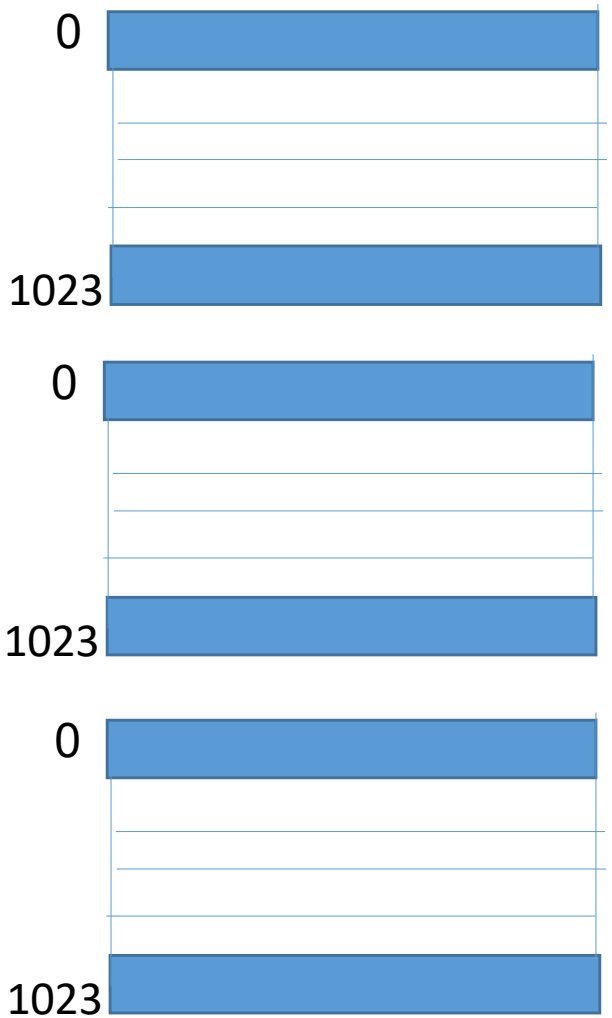


Page Table

Main Memory

روش یک سطحی

PT1



PT2

روش دو سطحی

Main Memory

صفحه‌بندی چندسطحی (دوسطحی)

- **فرض کنید اندازه آدرس حافظه منطقی ۳۲ بیتی باشد. اندازه هر صفحه نیز 4KB است. روش ۲ سطحی و ۱ سطحی را مقایسه نمایید.**

روش ۲ سطحی:

- ۱۲ بیت برای آفست. ۱۰ بیت برای شماره صفحه اول و ۱۰ بیت دیگر برای شماره صفحه دوم
- اندازه جدول صفحه سطح اول $۲^{۱۰}$ است که برابر می شود با: ۱۰۲٤
- اندازه جدول صفحه سطح ۲ نیز $۲^{۱۰}$ است که معادل ۱۰۲٤ است.
- برای یک فرآیند به اندازه 12KB، به ۳ صفحه 4KB نیاز است.
- بنابراین در جدول صفحه سطح اول ۳ تا صفحه استفاده می شود که هر کدام نیز به یک جدول صفحه در سطح ۲ اشاره خواهند کرد. در مجموع میزان حافظه مورد نیاز برابر با ٤ جدول ۱۰۲٤ تایی خواهد بود، یعنی: **٤۰۹٦**

حافظه مجازی

- در روش‌های مدیریتی پیشین، برای اجرا، کل برنامه (فرآیند) به حافظه اصلی منتقل می‌شد.
- **حافظه مجازی**
- تکنیکی است که اجازه می‌دهد فرآیندها بدون آنکه لازم باشند به طور کامل در حافظه اصلی قرار گیرند، اجرا شوند. بنابراین برنامه‌ها می‌توانند بزرگتر از حافظه اصلی باشند.
- **مزایا**
- چون هر برنامه می‌تواند حافظه فیزیکی کمتری اشغال کند، برنامه‌های بیشتری می‌توانند همزمان اجرا شوند.
- برنامه به اندازه حافظه فیزیکی محدود نمی‌شود. برنامه‌نویس می‌تواند برنامه‌هایی با فضای آدرس‌دهی مجازی فوق العاده بزرگ بنویسید.
- عملیات I/O کمتری جهت بارگذاری یا مبادله هر برنامه لازم است. بنابراین برنامه می‌تواند سریع‌تر باشد.

پیاده‌سازی حافظه مجازی

1. صفحه‌بندی نیازی (Demand paging)
2. قطعه‌بندی نیازی (Demand segmentation)
3. قطعه بندی با صفحه بندی

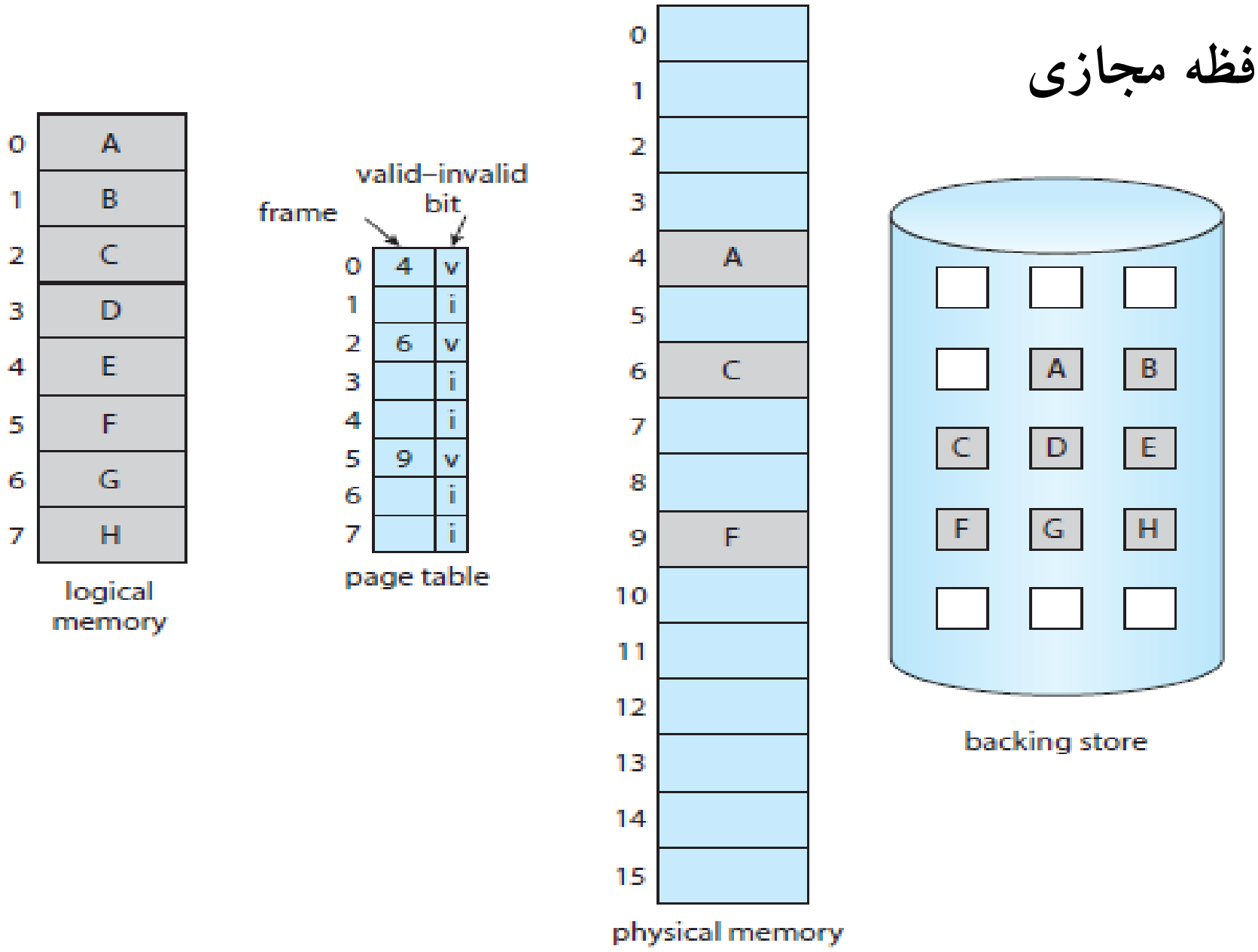
صفحه‌بندی نیازی

- فقط صفحاتی که نیاز فرآیند است از حافظه دیسک به حافظه اصلی آورده می‌شود.

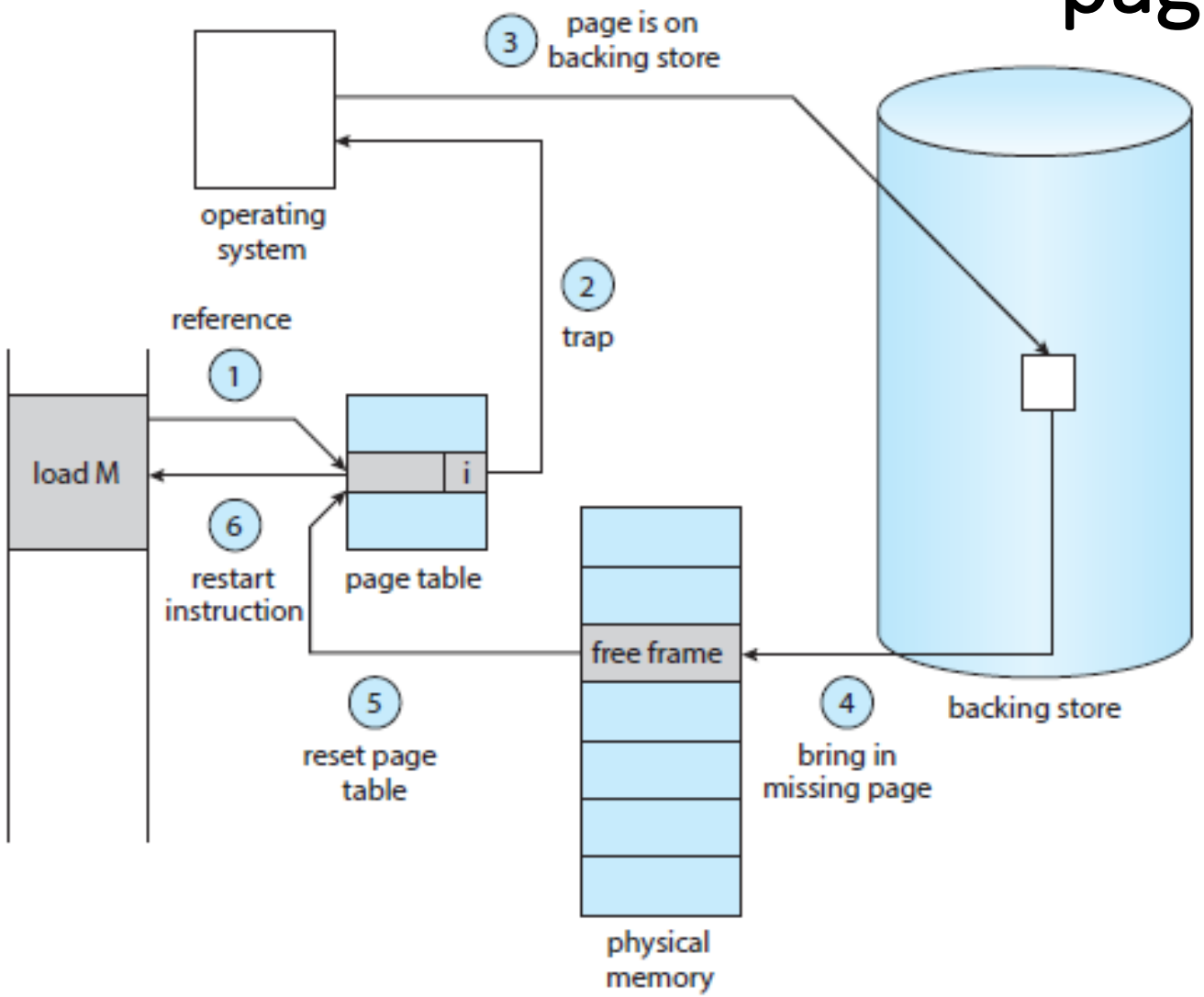
- اطلاعاتی مانند اینکه کدام صفحه در دیسک و کدام در حافظه است، نیاز می‌باشد.
  - با افزودن بیتی مانند valid و invalid این کار انجام می‌شود.

- دسترسی به صفحه‌ای که برچسب invalid دارد، اصطلاحا نقص صفحه **page fault** رخ داده است.

جدول صفحه در حافظه مجازی



رفتار سیستم در page fault



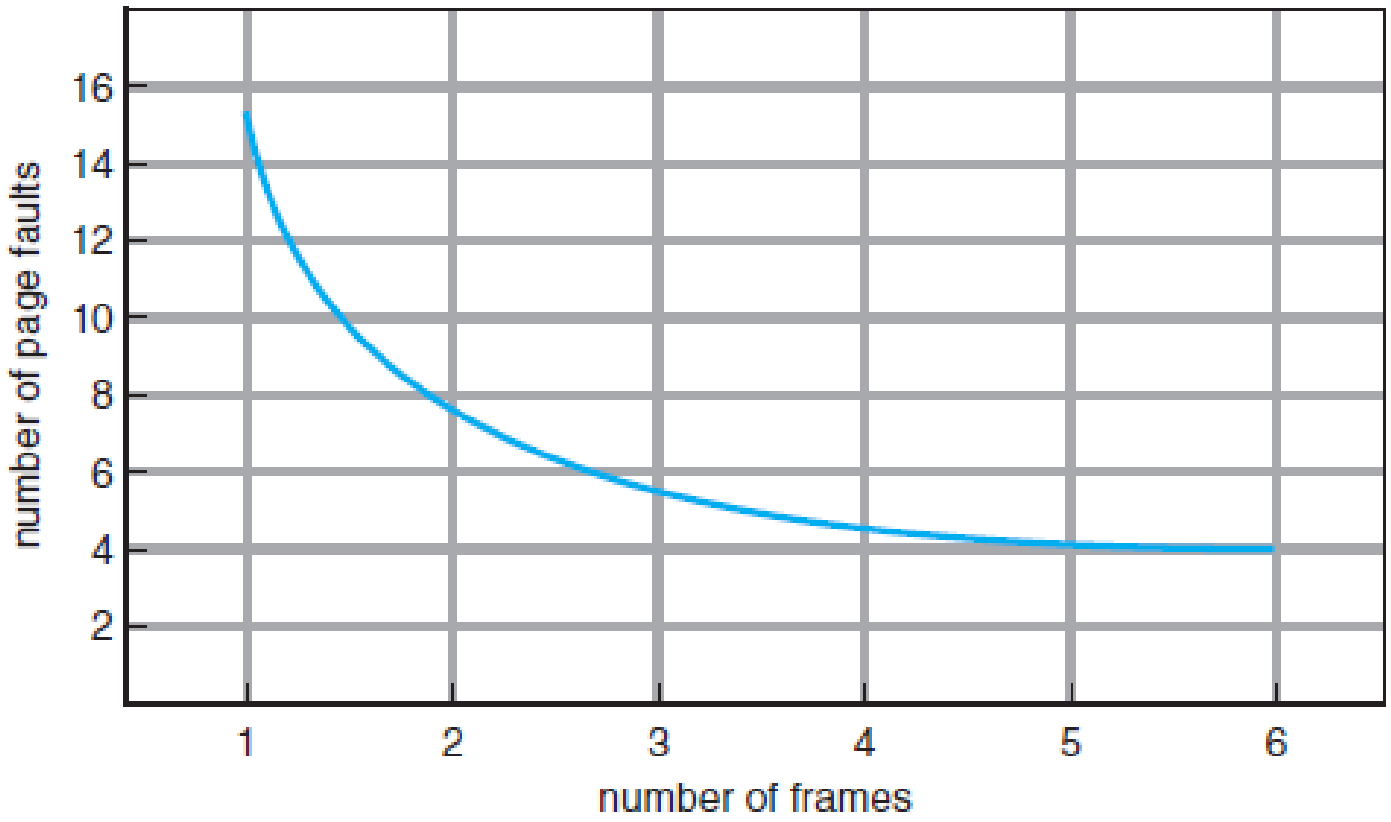
رفتار سیستم در page fault

1. آیا مراجعه به حافظه در محدوده آدرس منطقی فرآیند می باشد یا نه؟
2. اگر آدرس مربوطه خارج از محدوده برنامه است، پردازش خاتمه می‌یابد. اگر آدرس درست باشد، اما صفحه در حافظه نباشد، باید صفحه به حافظه اصلی آورده شود.

3. یک فریم (قاب) آزاد از لیست فریم های آزاد انتخاب می‌شود.
4. صفحه مورد نظر از دیسک به حافظه منتقل می شود.

1. جدول صفحه تغییر می یابد تا مشخص شود صفحه در حافظه قرار گرفته است.
2. اجرای دستور که در این شرایط وقفه خورده بود، مجدد از سر گرفته می‌شود.

ارتباط نقص صفحه با تعداد فریم (قاب)



اگر فریم (قاب) آزادی در حافظه اصلی وجود نداشته نباشد، باید یکی از صفحات در قاب حافظه اصلی به دیسک منتقل شده، تا فضای خالی ایجاد شود. به این تکنیک جایگزینی صفحه (Page replacement) گویند.

الگوریتم‌های جایگزینی صفحه

- الگوریتم FIFO
- الگوریتم بهینه (OPT)
- الگوریتم LRU
- الگوریتم سالمندی (Aging)
- الگوریتم LFU

الگوریتم FIFO

- ساده ترین الگوریتم است.
- زمانی که یک صفحه باید جایگزین شود، قدیمی‌ترین صفحه انتخاب می‌شود.

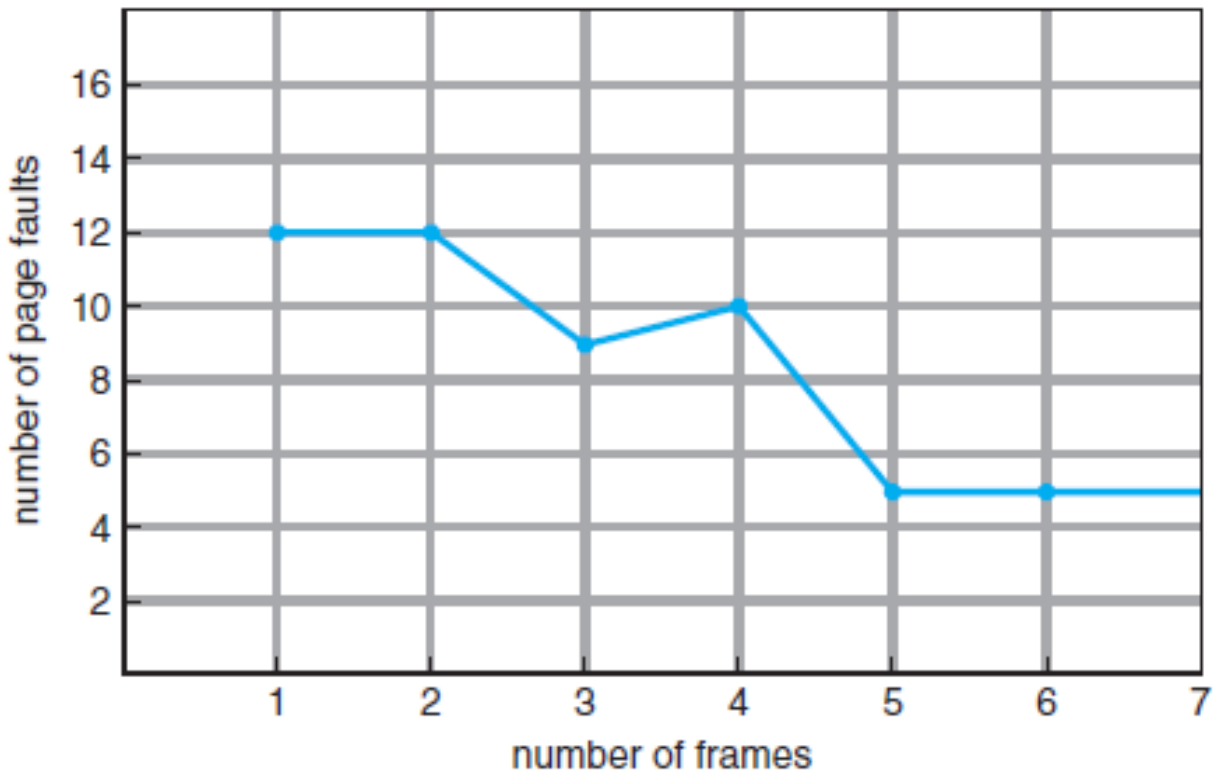
تعداد نقص صفحه ۱۵ است.

- با صف FIFO می توان آن را پیاده سازی کرد.
- مثال

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	0	1	2	3	0	4	2	3	0	1	2	7	0	1
	0	0	1	2	3	0	4	2	3	0	1	2	7	0	1	
		1	2	3	0	4	2	3	0	1	2	7	0	1		

الگوریتم FIFO



- در بررسی‌های انجام شده در الگوریتم FIFO، نتایج نشان داد که، با افزایش تعداد قاب‌ها در حافظه اصلی، تعداد نقص صفحه کاهش پیدا نکرد. این نتیجه غیرمنتظره را **ناهنجاری بلدی** (**Belady's Anomly**) گویند.

الگوریتم بهینه OPT

- این روش کم‌ترین خطای صفحه ممکن را برای یک تعداد معین قاب صفحه دارد.

- هنگام نقص صفحه، صفحه‌ای را جایگزین می‌کند که برای طولانی‌ترین دوره زمانی استفاده نخواهد شد.

- پیاده‌سازی این الگوریتم مشکل است. چون به اطلاعاتی از آینده نیاز دارد.

الگوریتم بهینه OPT

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

الگوریتم LRU

- مانند روش الگوریتم بهینه است و از گذشته اخیر به عنوان تقریبی برای آینده نزدیک استفاده می‌کند.

- صفحه‌ای جایگزین می‌شود که طولانی‌ترین دوره زمانی مورد استفاده قرار نگرفته است.

- به آن الگوریتم اخیرا کمترین استفاده شده «Least Recently Used» می‌گویند.

الگوریتم LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		

پیاده‌سازی LRU

- **مبتنی بر شمارنده**
- در هر سطر جدول صفحه یک فیلد برای زمان استفاده در نظر گرفته می شود.
- برای CPU نیز یک شمارنده اضافه می‌شود و به ازای هر مراجعه به حافظه، یک واحد افزایش می‌یابد.
- هنگام مراجعه به صفحه، محتویات این شمارنده در فیلد زمان صفحه کپی می‌شود.
- **مبتنی بر پشته**
- پشته ای از شماره صفحات وجود دارد.
- هنگام مراجعه به صفحه، شماره آن صفحه از پشته حذف شده و بالای پشته قرار می‌گیرد.
- خانه بالایی پشته، صفحه‌ای است که اخیرا بیشترین استفاده از آن شده است.
- پشته بهتر است با لیست پیوندی پیاده‌سازی شود.

پیاده‌سازی **LRU**، با توجه به بروزرسانی در هر مراجعه به حافظه، ارزان نیست.
بنابراین معمولا از روش‌های دیگری استفاده می‌شود که **LRU** را تقریب می‌زنند.

الگوریتم سالمندی – الگوریتم بیت‌های مراجعه اضافی

- با ثبت بیت‌های مراجعه در فاصله‌های زمانی، می‌توانیم اطلاعات بیشتری در رابطه با ترتیب مراجعات صفحه به دست آوریم.
- یک بایت برای هر صفحه جدول فرض می‌شود.
- در فواصل زمانی منظم، وقفه تایمر، کنترل سیستم را به سیستم‌عامل انتقال داده و سیستم‌عامل این بایت را به سمت راست شیفت می‌دهد.
- ثبات شیفت‌دهنده، سابقه‌ی استفاده هر یک از صفحات را در هشت دوره زمانی اخیر نشان می‌دهد.
- صفحه‌ای که مقدار عددی آن کوچکترین است، برای جایگزینی انتخاب می‌شود.
- **بیت مراجعه : Reference = R**

مثال

- با فرض ٤ قاب صفحه، در اولین پالس ساعت، بیت های ارجاع قاب های صفحه برابر ۰۱۱۱ است (بیت ارجتع قاب صفحه اول ۰ و بقیه صفحات ۱ است). ارزش بیتهای ارجاع در پالس های بعدی ساعت به ترتیب از چپ به راست برابر است با:

1010, 0010, 1010, 1011

با استفاده از الگوریتم Aging، هنگام نقص صفحه، کدام قاب صفحه کاندید جابجایی است؟

پاسخ

	R	شمارنده
قاب ۱	0	0000 0000
قاب ۲	1	1000 0000
قاب ۳	1	1000 0000
قاب ٤	1	1000 0000

قاب صفحه ۲ کمترین مقدار را دارد و باید جایگزین شود.

ادامه مثال

	R	شمارنده
قاب ۱	0	0100 0000
قاب ۲	0	00100 000
قاب ۳	1	11100 000
قاب ٤	0	00100 000

	R	شمارنده
قاب ۱	1	1000 0000
قاب ۲	0	0100 0000
قاب ۳	1	1100 0000
قاب ٤	0	0100 0000

	R	شمارنده
قاب ۱	1	1101 0000
قاب ۲	0	0000 1000
قاب ۳	1	1111 1000
قاب ٤	1	1000 1000

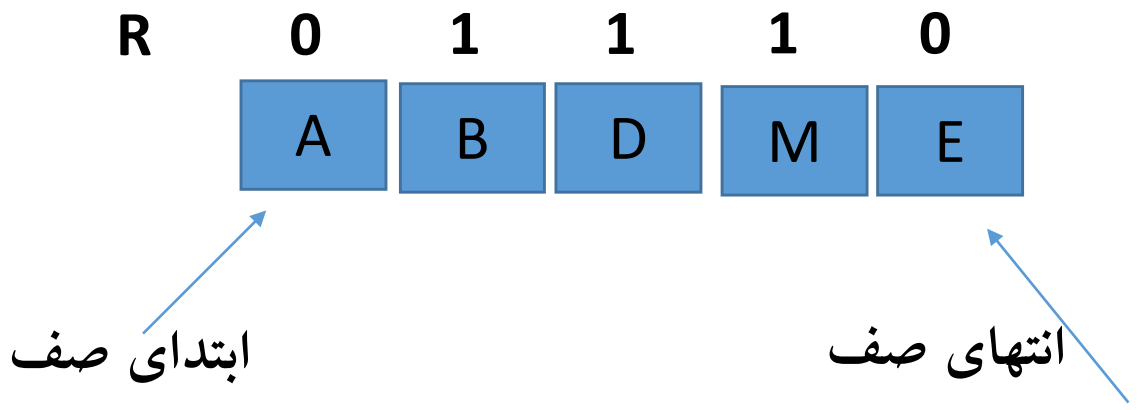
	R	شمارنده
قاب ۱	1	1010 0000
قاب ۲	0	0001 0000
قاب ۳	1	1111 0000
قاب ٤	0	0001 0000

الگوریتم دومین شانس

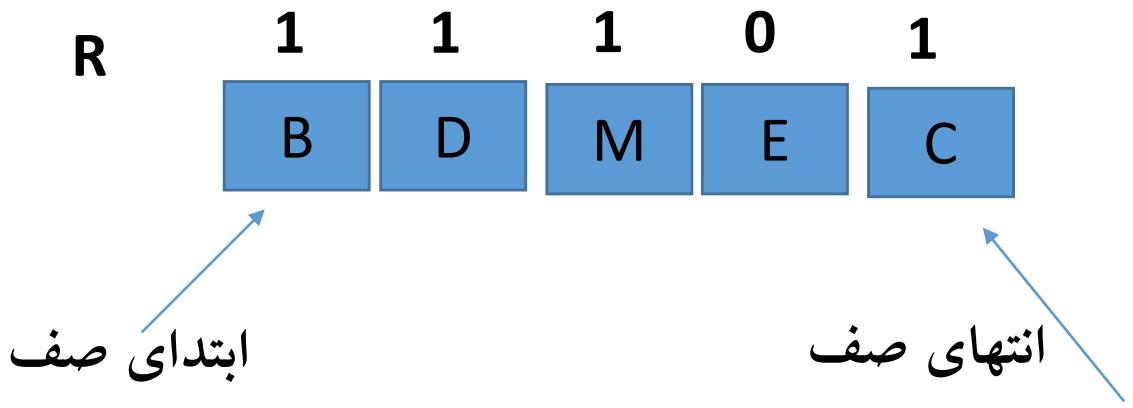
- مبتنی بر الگوریتم FIFO است.
- بر این ایده است که صفحاتی که زیاد استفاده شده اند، از حافظه خارج نشوند.
- از ابتدای صف که قدیمی ترین صفحات قرار دارند، بیت مراجعه R بررسی می‌شود. اگر صفحه استفاده نشده باشد، جایگزین خواهد شد. در غیر این صورت
- اگر بیت مراجعه صفحه برابر ۱ است، آن بیت صفر شده، صفحه به انتهای لیست منتقل می‌شود و زمان ورودش به زمان فعلی مقداردهی می‌شود. (مانند این است که صفحه تازه وارد حافظه شده است)

- در واقع، به صفحه یک شانس دومی داده می‌شود

الگوریتم دومین شانس

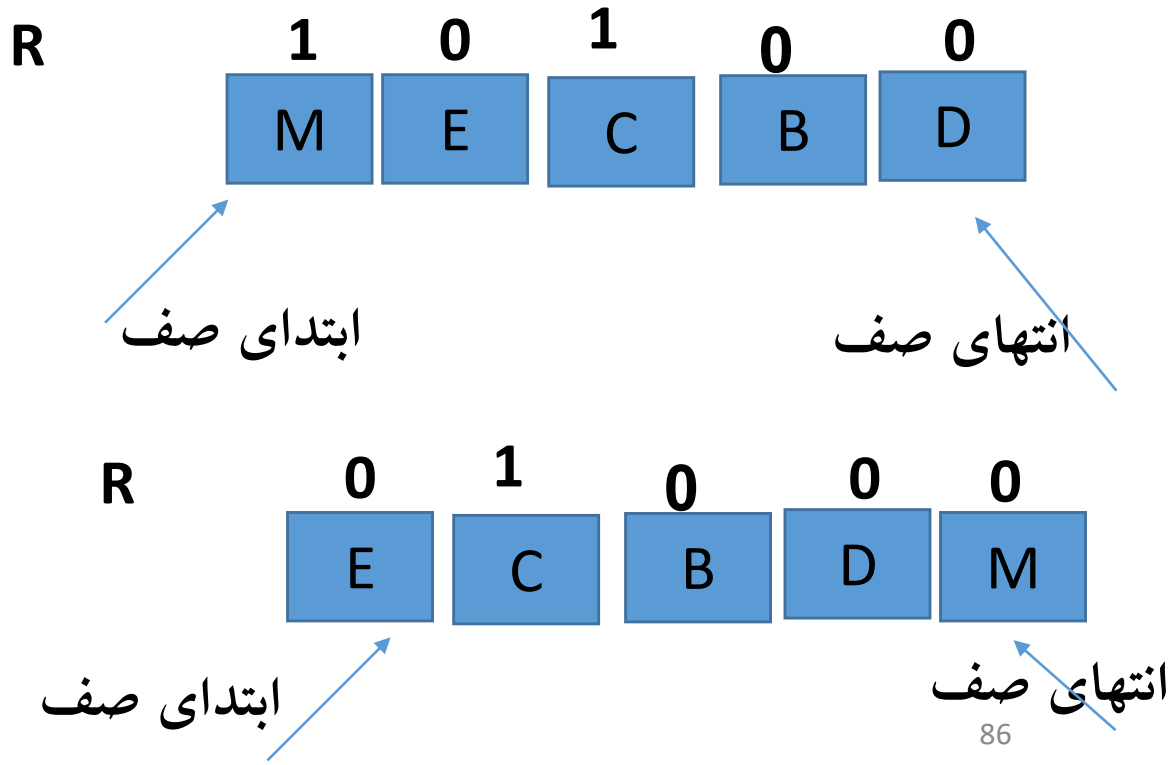


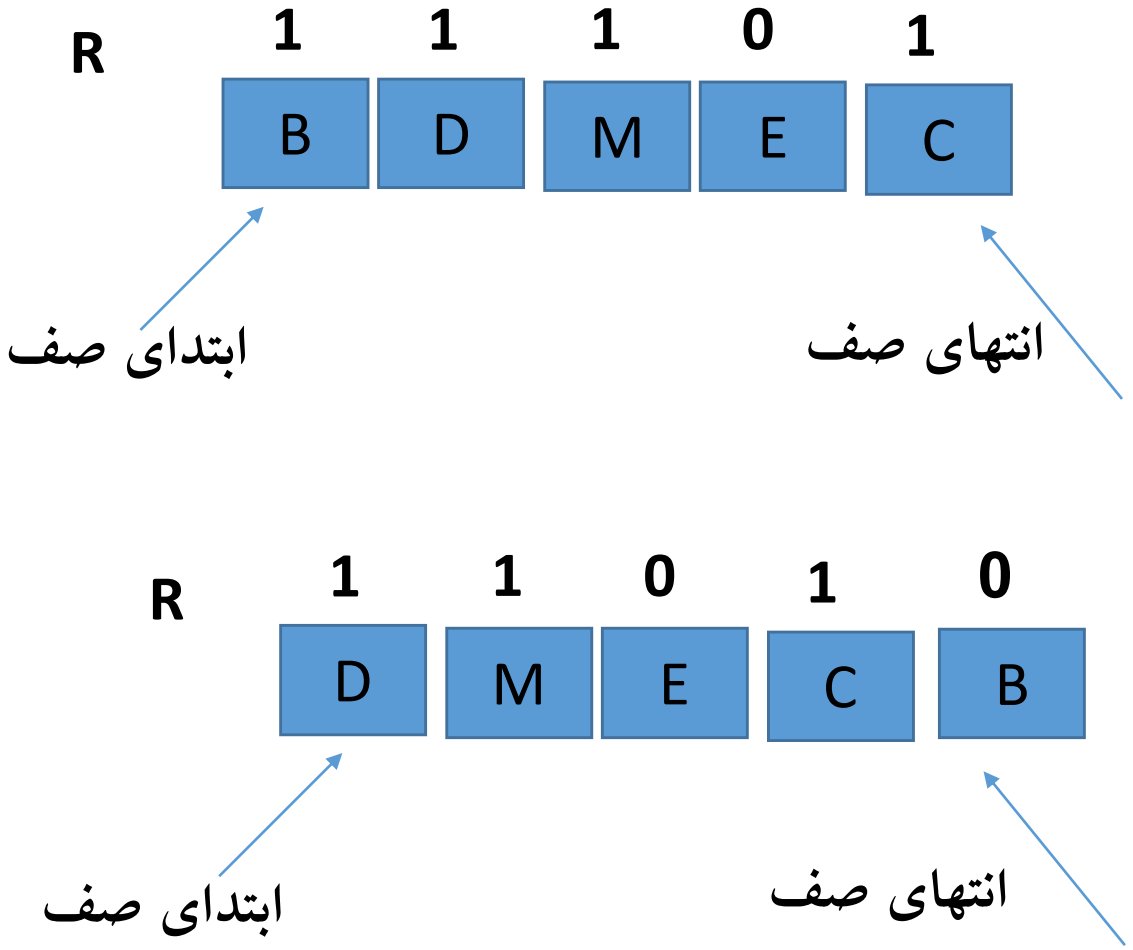
متقاضی صفحه C



الگوریتم دومین شانس

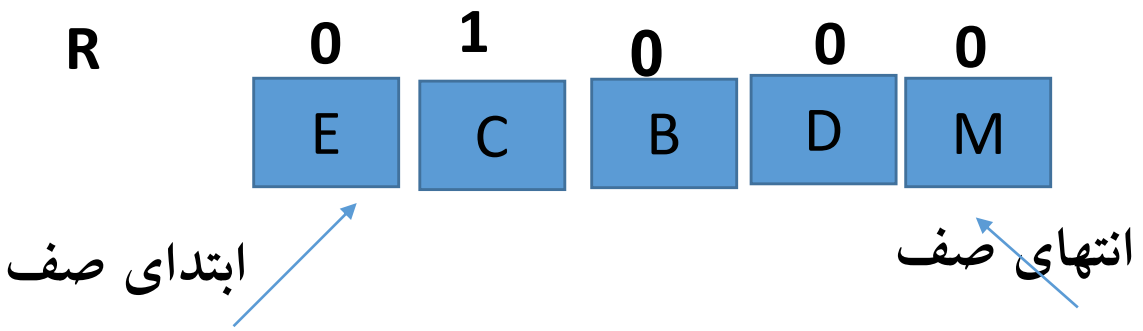
متقاضی صفحه N

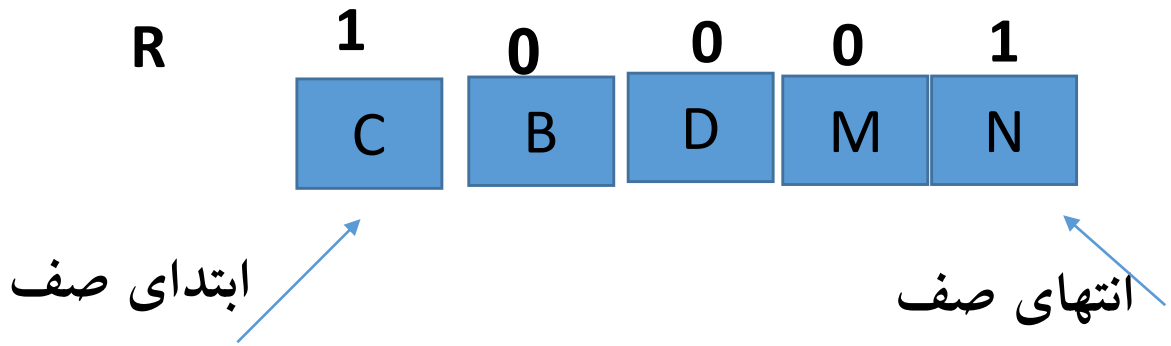




الگوریتم دومین شانس

متقاضی صفحه N





الگوریتم NRU (Not Recently Used)

- در این الگوریتم از ۲ بیت R و M استفاده می‌شود.
- در ابتدا برای تمام صفحات یک فرآیند این دو بیت برابر صفر هستند.
- در هر وقفه ساعت، سیستم عامل بیت R هر صفحه را صفر می‌کند.
- صفحات به چهار کلاس بر اساس مقادیر (R,M) تقسیم بندی می‌شوند.
  - کلاس ۰: R,M=0,0
  - کلاس ۱: R,M=0,1
  - کلاس ۲: R,M=1,0
  - کلاس ۳: R,M=1,1

- اولین صفحه ای که در پایین ترین کلاس غیرتهی قرار دارد، هنگام نقص صفحه برای جایگزینی انتخاب می‌شود.

الگوریتم NRU (Not Recently Used)

- این الگوریتم برای پیاده سازی مناسب است.

- مشابه روش FIFO امکان ناهنجاری بلدی را دارد.

- در بعضی کتب علمی این روش به نام دومین شانس بهبود یافته نامگذاری شده است.
  - **Enhanced Second Chance**

الگوریتم LFU (Least Frequently Used)

- یک شمارنده نرم‌افزاری به هر صفحه نسبت داده می‌شود.
- همه شمارنده‌ها در ابتدا صفر هستند.
- با هر بار دسترسی به هر صفحه مقدار شمارنده مربوط به آن افزایش می‌یابد.
- هر شمارنده نمایانگر آن است که صفحه مربوطه چند بار دستیابی شده است.

- هنگام نقص صفحه، صفحه‌ای که شمارنده آن کمترین مقدار را دارد برای جایگزینی انتخاب می‌شود.
  - در صورتی که صفحاتی با مقدار شمارنده یکسان موجود باشند، می‌توان بر اساس FIFO، صفحه‌ای را برای جایگزینی انتخاب کرد.

الگوریتم LFU (Least Frequently Used)

- **مزایا**
- کاهش حذف صفحات پرکاربرد.
- استفاده از تاریخچه واقعی سیستم.

- **معایب**
- سربار نگهداری شمارنده زیاد است.
- ممکن است صفحات قدیمی که مقدار شمارنده بالایی هم دارند، به مدت طولانی در حافظه باقی بمانند ولی دیگر استفاده نشوند.

الگوریتم MFU (Most Frequently Used)

- برعکس الگوریتم LFU است و صفحه ای که شمارنده آن بیشترین مقدار را نشان می‌دهد برای جایگزینی انتخاب می‌شود.

- مبتنی بر این ایده است که صفحه‌ای که مقدار شمارنده آن کم است، احتمالا جدید وارد حافظه شده است و هنوز در حال استفاده می‌باشد.

الگوریتم‌هایی مانند FIFO، NRU و دومین شانس مشکل ناهنجاری بلدی را دارند.

الگوریتم هایی مانند LRU، OPT، LFU مشکل ناهنجاری بلدی را ندارند.

تحلیل طراحی سیستم های صفحه بندی

- تخصیص قاب
- مجموعه مدل کاری
  - کوبیدگی
- تخصیص سراسری و محلی
- بافر کردن صفحه
- فرکانس نقص صفحه

تخصیص قاب

- میزان تخصیص حافظه به فرآیندهای مختلف مساله مهمی است.

1. تخصیص مساوی
   - تمام قاب های موجود (m) به n تا فرآیند بطور مساوی تقسیم می شود.
   - مثلا ۱۰ فرآیند و ۸۰ قاب حافظه داشته باشیم به هر فرآیند ۸ قاب تخصیص داده می شود.
2. تخصیص متناسب
   - حافظه آزاد به نسبت اندازه فرآیندها تخصیص داده می شود.
   - مثال: دو فرآیند P1 و P2 هر کدام به ترتیب ۱۰ و ۵۰ صفحه دارند. تعداد قاب های آزاد حافظه اصلی ٤۰ می باشد.

- P1: 10/(10+50)×40=40/6=7
- P2: 50/(10+50) ×40=5/6×40=33

مجموعه مدل کاری

- استراتژی صفحه بندی درخواستی
- در ابتدای اجرای یک فرآیند، معمولا صفحات آن در حافظه وجود ندارد. بنابراین در ابتدا نقص صفحه رخ می دهد. اما معمولا بعد از گذشت مدت زمانی، اکثر صفحات مورد نیاز فرآیند در حافظه قرار دارند و برنامه در سرعت مناسبی اجرا خواهد شد. این استراتژی که صفحات فقط موقعی که نیاز هستند، بارگذاری می‌شوند و از قبل در حافظه وجود ندارند، صفحه‌بندی درخواستی گویند.

- مجموعه صفحاتی که فرآیند در حال حاضر از آنها استفاده می کند، **مجموعه کاری** نام دارد.

کوبیدگی (Thrashing)

- هنگامی که حافظه کوچک است و مجموعه کاری نتواند در حافظه باشد، تعداد نقص صفحه افزایش می یابد. بنابراین فرآیند به کندی اجرا می‌شود. چون هر بار نقص صفحه، زمان دسترسی به دیسک به سیستم تحمیل می‌شود.

- به برنامه ای که هر چند دستورالعمل یکبار نقص صفحه به وجود بیاورد **کوبیدگی** می‌گویند.

- در طراحی سیستم صفحه‌بندی سعی می شود که قبل از دادن CPU به یک فرآیند، مجموعه کاری اش در حافظه وجود داشته باشد، این طرح را **مدل مجموعه کاری** می‌گویند. بنابراین نقص صفحه کاهش می یابد.
  - روش هایی مانند الگوریتم سالمندی، که مراجعات را در یک دوره زمانی مناسبی در قالب یک بایت نشان می دهد، توصیف خوبی از فرآیند را دارد و می توان بر این اساس، مجموعه کاری را شناسایی کرد.

تخصیص سراسری و محلی

- الگوریتم های جایگزینی صفحه می‌توانند در قالب سراسری یا محلی تقسیم‌بندی شوند.
- سراسری
- هنگامی که فرآیند به یک قاب نیاز دارد، می تواند از بین مجموعه کل فریم‌ها انتخاب نماید. یعنی می‌تواند فریمی از مجموعه یک فرآیند دیگر را انتخاب نماید.

- محلی
- هر فرآیند فقط مجاز است که از مجموعه قاب های مخصوص خود استفاده کند.

می‌توان برای اجرای فرآیندهای با اولویت بالا، سیستم‌عامل از قاب‌های فرآیندهای با اولویت پایین‌تر نیز استفاه نماید که مبتنی بر روش سراسری خواهد بود.

بافر کردن صفحه

- **ایده اصلی:**

1. تعدادی فریم از لیست قاب های آزاد حافظه رزرو می شود.
2. هنگام خطای نقص صفحه، یک قاب در حافظه اصلی برای جایگزینی انتخاب می‌شود.
3. صفحه مذکور به جای اینکه در این قاب قرار گیرد در یکی از قاب های رزروی قرار می‌گیرد.
4. فرآیند اجرای خود را سریعا آغاز می کند و منتظر نوشتن صفحه جایگزین برروی دیسک نمی‌شود.
5. بعدا در فرصتی مناسب که در حالی که برنامه در حال اجرا است، قاب انتخاب شده برروی دیسک نوشته شده و به عنوان یک قاب آزاد رزرو شده نگهداری خواهد شد.

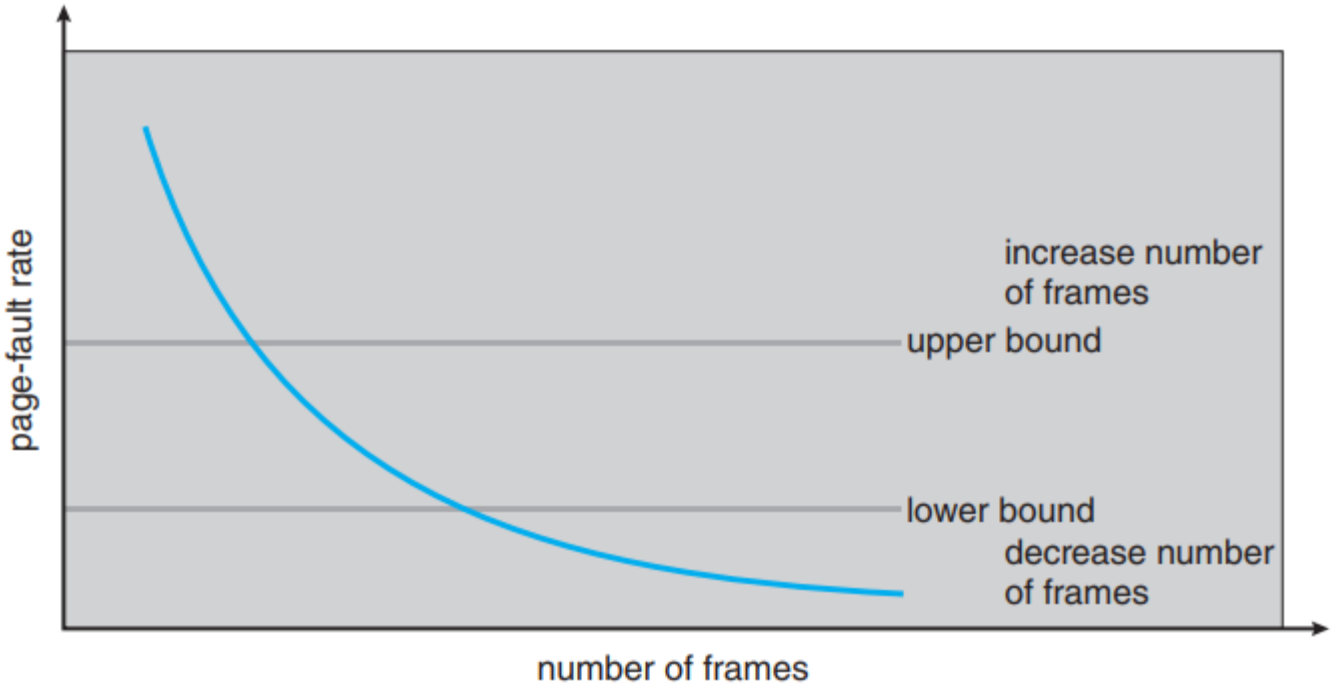
- **مزایا**
  - افزایش کارایی سیستم

فرکانس نقص صفحه (Page Fault Frequency-PFF)

- با کنترل تعداد خطاهای صفحه می‌توان کوبیدگی را کنترل کرد.
- اگر تعداد فرآیندهای موجود در حافظه آنقدر زیاد باشد که نتوان نرخ خطای صفحه را پایین تر از کران بالا گرفت، تعدادی از فرآیندها را معلق کرده و قابهای آن ها را آزاد می کنند.
- اگر نرخ خطای صفحه از حد کران بالا بیشتر شود به آن فرآیند قاب آزاد دیگری می‌دهیم

- اگر نرخ خطای صفحه از کران پایین کمتر شود، قابی از آن فرآیند را آزاد می‌کنیم زیرا این فرآیند قاب‌های بیش از حد نیاز در اختیار دارد.

فرکانس نقص صفحه (Page Fault Frequency-PFF)



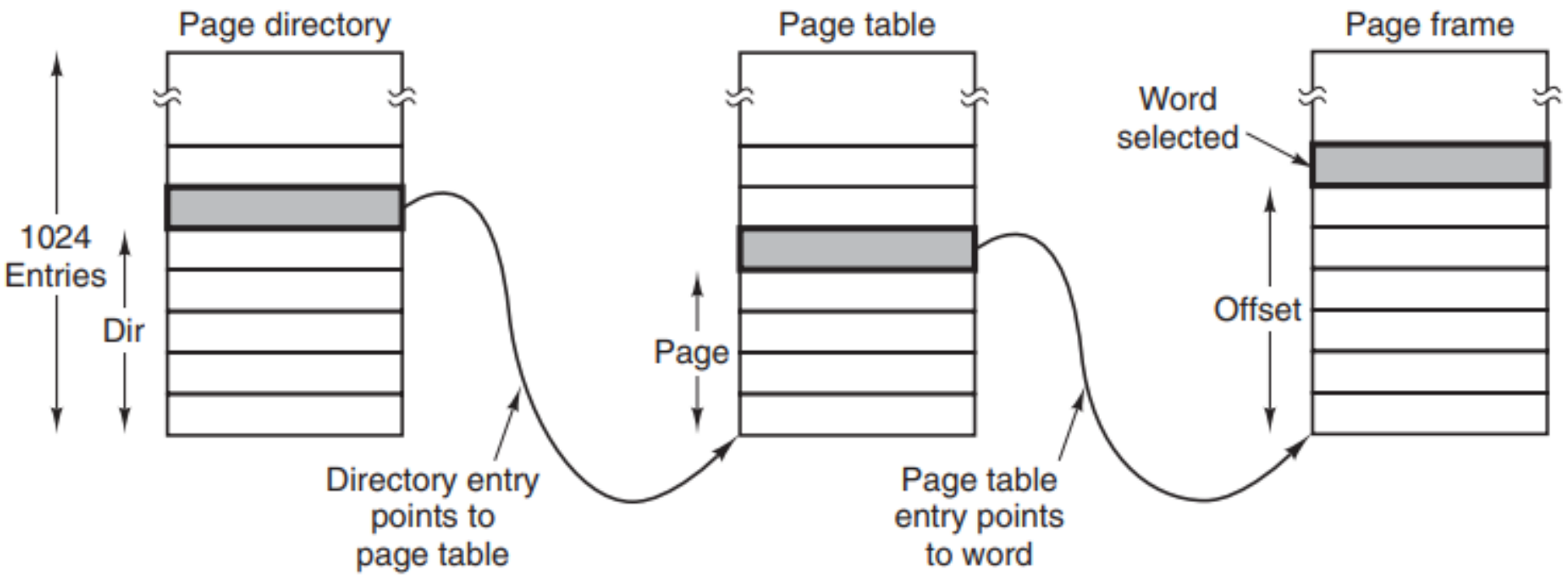
ترکیب قطعه‌بندی با صفحه‌بندی

- در صورت بزرگی اندازه قطعات، نگهداری دائمی آنها در حافظه مناسب نیست. بنابراین از صفحه بندی قطعات استفاده می شود که فقط تعدادی از صفحات مورد نیاز در حافظه باقی بمانند.

- در معماری‌های اخیر پردازنده‌ها، از ترکیب قطعه‌بندی و صفحه‌بندی استفاده می‌شود. بنابراین سیستم‌عامل بر روی این معماری‌ها، در تبدیلات آدرس از این تفکر استفاده می‌کنند.

- برنامه به تعدادی قطعه تقسیم شده و هر قطعه نیز به تعدادی صفحه تقسیم خواهد شد.

ترکیب قطعه‌بندی با صفحه‌بندی

- آدرس منطقی به ۳ قسمت تقسیم می شود:
  - شماره قطعه
  - شماره صفحه
  - آفست

ترکیب قطعه‌بندی با صفحه‌بندی