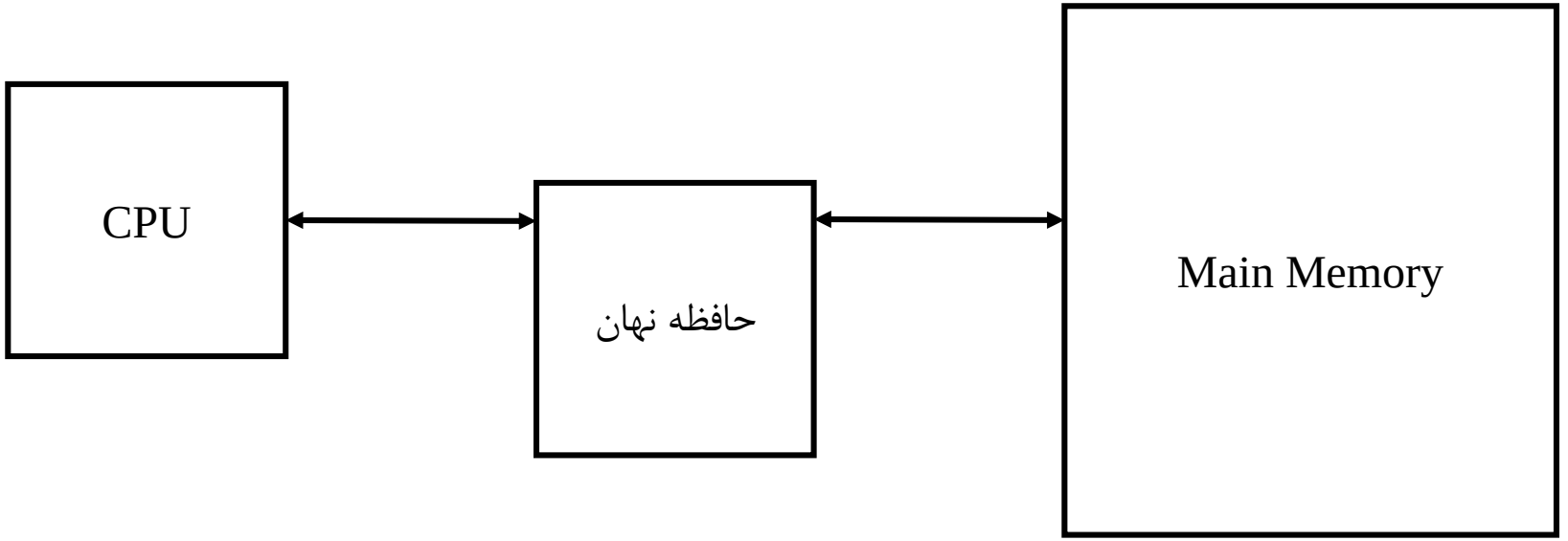
بنام خداوند جان و خرد

معماری کامپیوتر

جلسه نهم: حافظه نهان-۳

حافظه نهان (Cache)

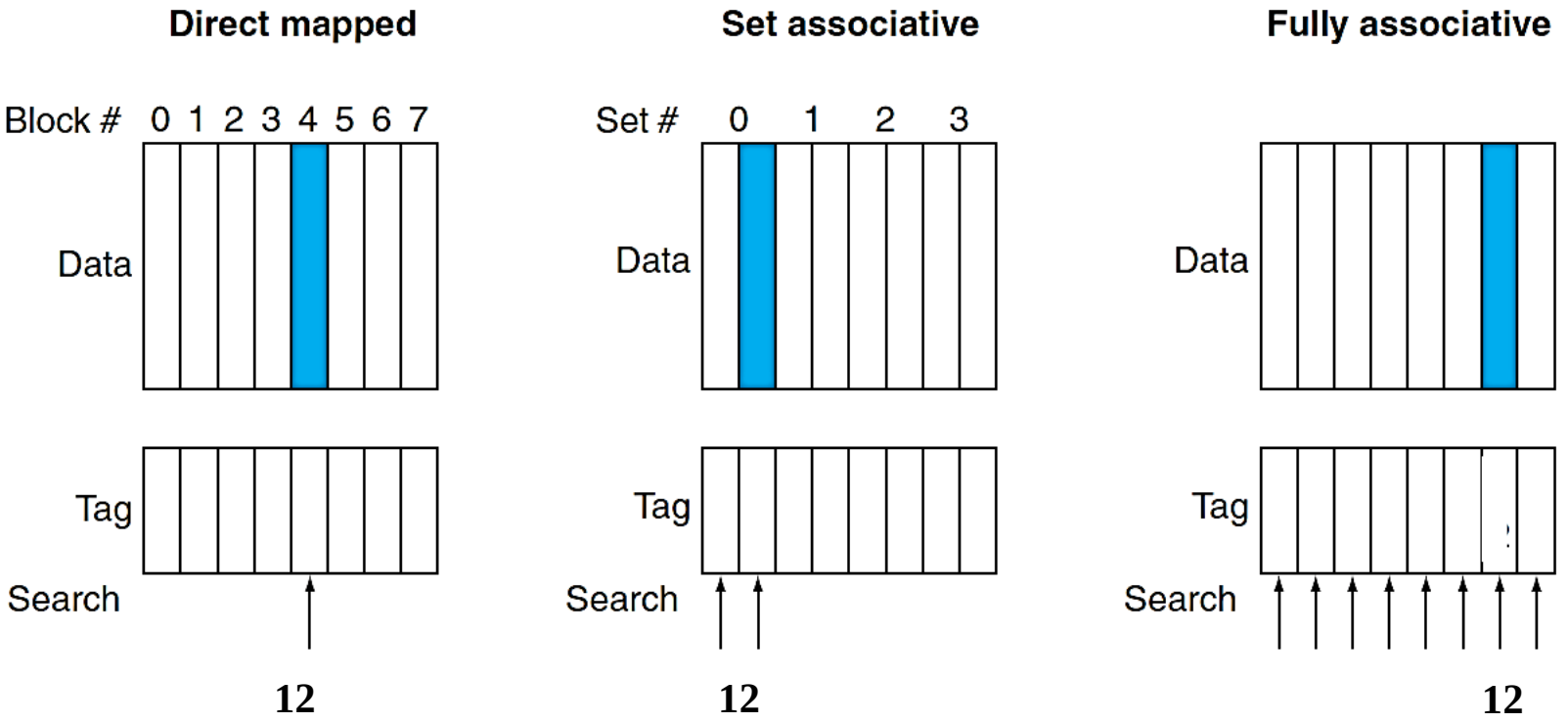
- درنظر گرفتن فضایی بین پردازنده و حافظه اصلی (مشابه جعبه کوچک کتابدار)
  - باهدف ذخیره‌سازی بخشی از حافظه اصلی جهت دسترسی سریع‌تر



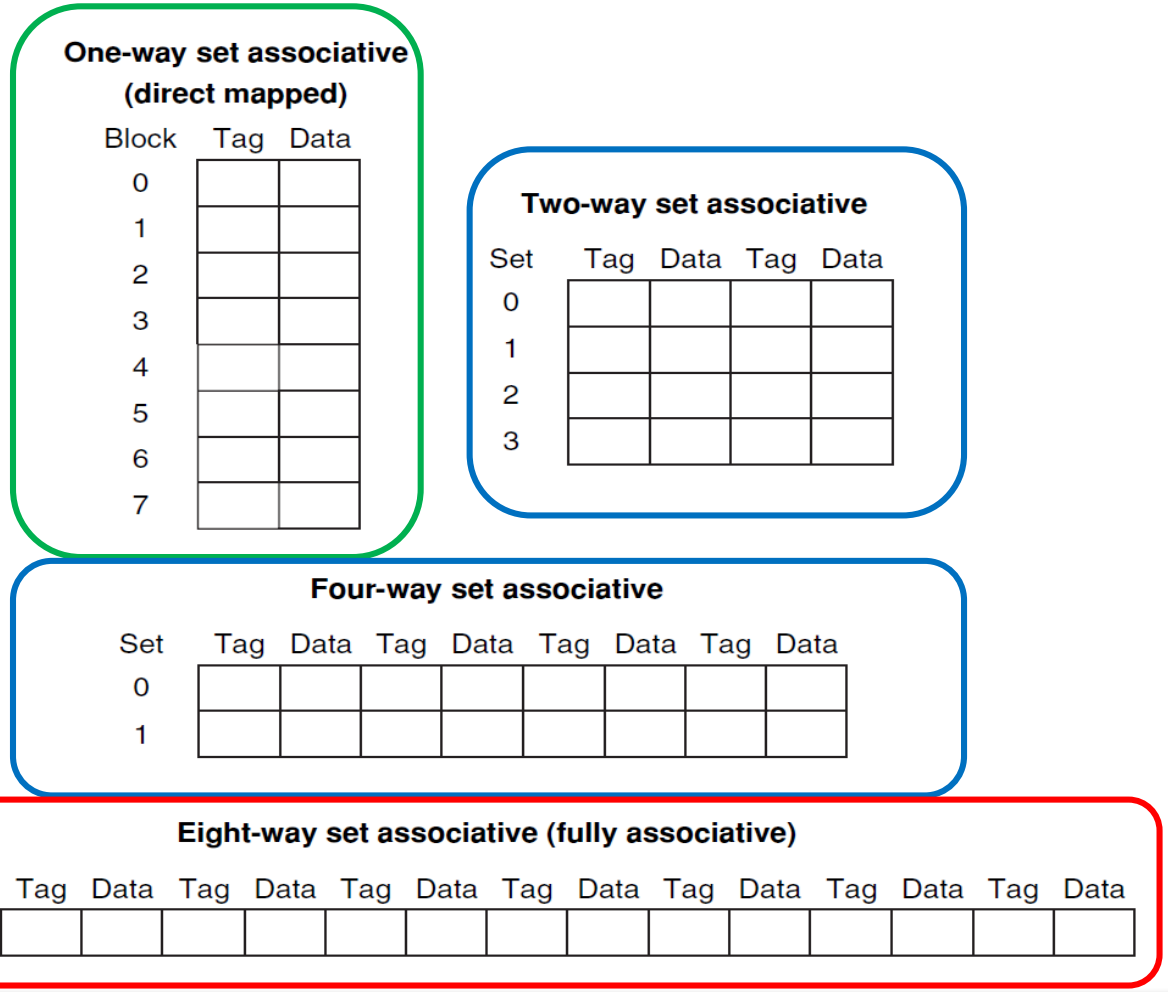
سیاست‌های جایدهی حافظه نهان

- داده‌ها را چگونه از حافظه اصلی در حافظه نهان ذخیره کنیم به‌نحوی که:
  - نرخ دسترسی بالا
  - جستجوی سریع و ساده داده‌های حافظه نهان
- سیاست‌های جایدهی:
  - Direct Map
  - Set Associative
  - Fully Associative

سیاست‌های جایدهی حافظه نهان



سیاست‌های جایدهی حافظه نهان



مثال

- حافظه نهان با سایز ۵۱۲ کیلوبایت و اندازه بلوک ۶۴ بایت داریم، همچنین سایز حافظه اصلی ۱۶ مگابایت است، اندازه و قالب آدرس برای سه حالت زیر چگونه است؟

a) DM b) 8WSA c) FA

- حل:

$M = 16\ MB = 2^{24}$, $C = 512\ KB = 2^{19}$, $B = 64B = 2^{6}$

a) اندازه آدرس: ۲۴ بیت، طول آفست: ۶ بیت، طول ایندکس: ۱۳=۶-۱۹، طول tag: ۵ بیت

b) اندازه آدرس: ۲۴ بیت، طول آفست: ۶ بیت، طول ایندکس: $\log\left(\frac{2^{13}}{2^{3}}\right) = 10$، طول tag: ۸ بیت

c) اندازه آدرس: ۲۴ بیت، طول آفست: ۶ بیت، طول ایندکس: صفر، طول tag: ۱۸ بیت

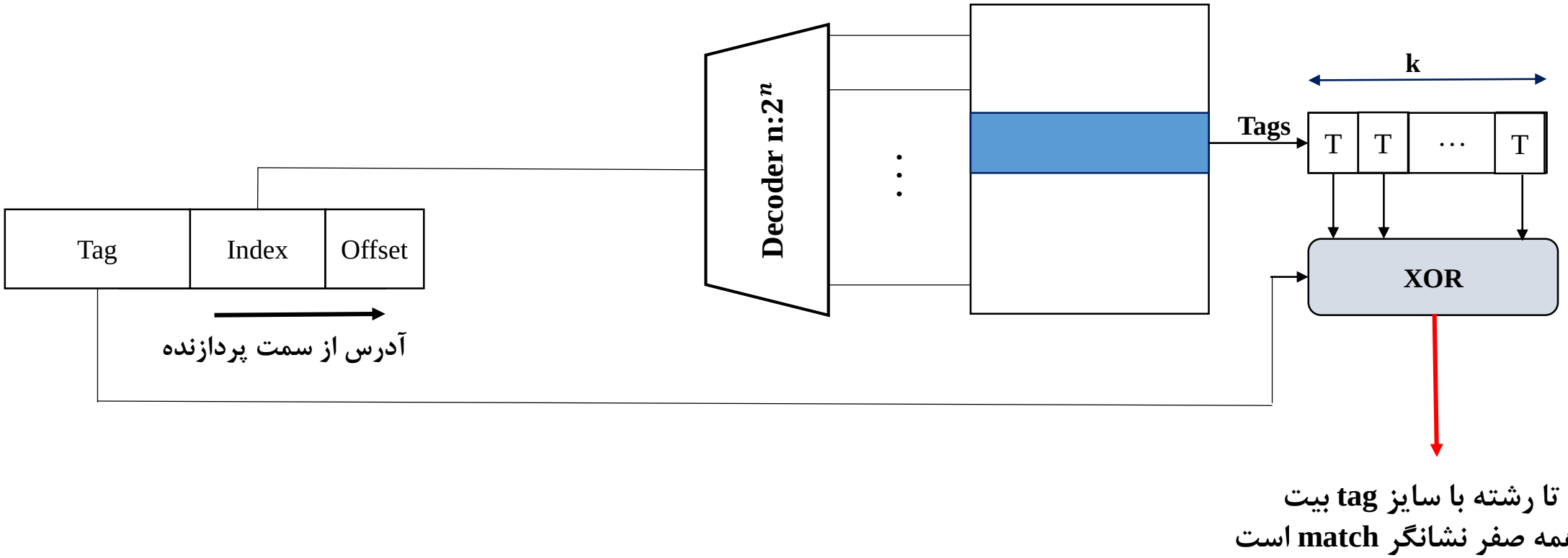
طراحی مداری حافظه نهان

- پیاده‌سازی مدار کنترلی انواع حافظه نهان براساس سیاست‌های نگاشت آدرس بررسی شده
- بررسی حالت کلی kWSA که همه حالت‌ها را پوشش می‌دهد
  - 1WSA: Direct map
  - 2/4/…WSA: set associative
  - cWSA: fully associative

طراحی مداری حافظه نهان

- با ورود آدرس از پردازنده به حافظه نهان
  - شماره set را استخراج می‌کنیم، بخش index از آدرس
  - این بخش به ماژول دیکدر وارد می‌شود
  - خروجی دیکدر به یک set اشاره دارد
  - برچسب آدرس ورودی با تمام k برچسب ذخیره شده در set مقایسه می‌شود
    - مقایسه توسط گیت xor
  - داده متناظر با برچسبی که خروجی xor آن صفر شده بازگردانده می‌شود (hit)

طراحی مداری حافظه نهان



طراحی مداری حافظه نهان

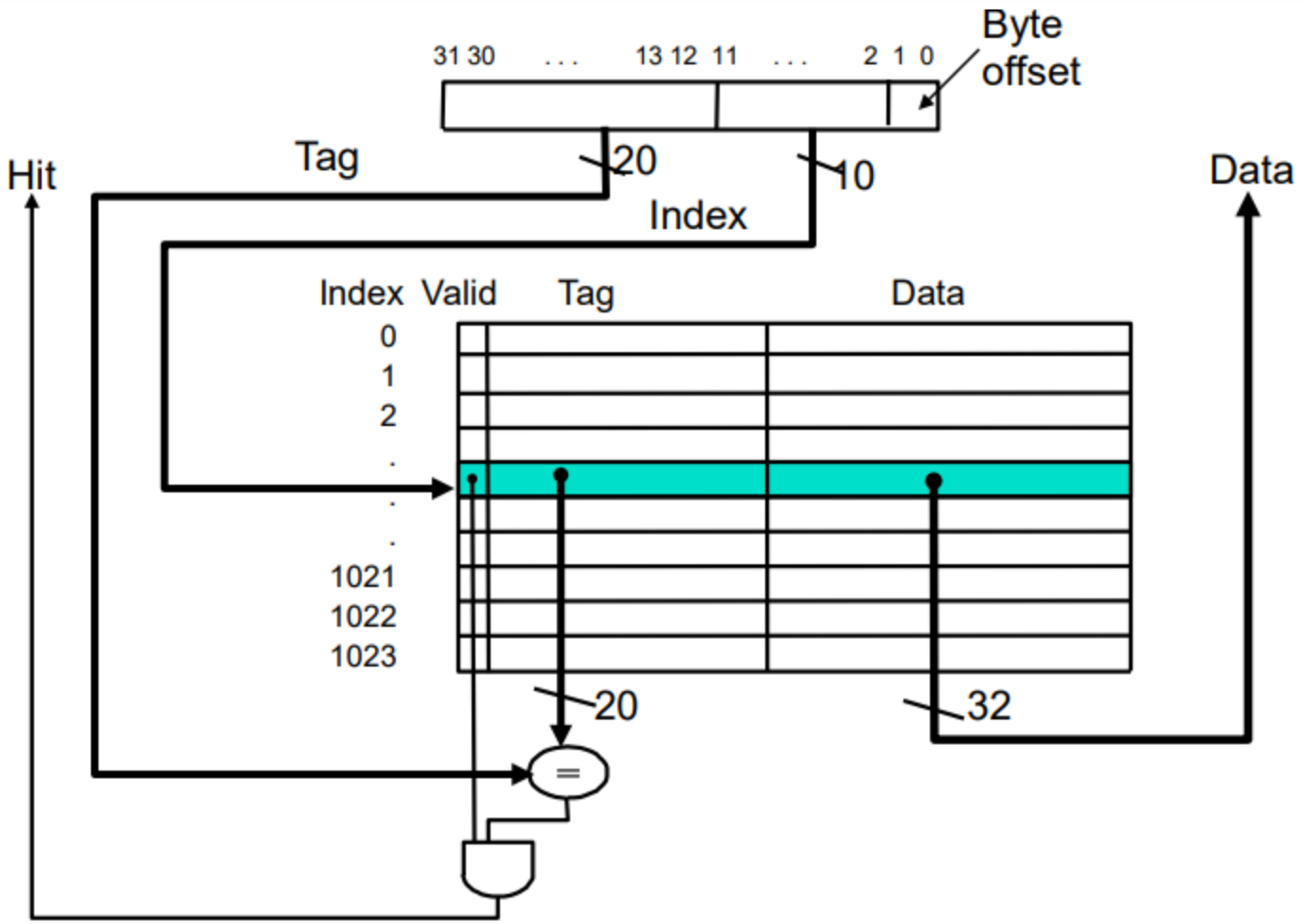
- اگر $k=1$ باشد یعنی طراحی نگاشت مستقیم، چه تغییراتی در مدار ایجاد می‌شود؟

- اگر $k=c$ باشد یعنی طراحی تمام انجمنی چطور؟

طراحی مداری حافظه نهان

- اگر $k=1$ باشد یعنی طراحی نگاشت مستقیم، چه تغییراتی در مدار ایجاد می‌شود؟
  - به بخش مقایسه‌گر نیازی نیست
  - دیکدر بزرگ‌تر می‌شود
- اگر $k=c$ باشد یعنی طراحی تمام انجمنی چطور؟
  - به دیکدر نیازی نیست
  - ماژول مقایسه بزرگ شده و توان مصرفی، مساحت و هزینه زیادی دارد

طراحی مداری حافظه نهان



مقایسه انواع طراحی حافظه نهان

	کارایی(HR)	توان مصرفی	مساحت	تاخیر
Direct Map	کم	کم	کم	کم
Set Associative	متوسط	متوسط	متوسط	متوسط
Fully Associative	زیاد	زیاد	زیاد	زیاد

حافظه نهان (Cache)

- **سوال‌های مهم:**
  - داده‌ها را چگونه از حافظه اصلی به حافظه نهان ببریم؟
  - داده‌های جدید را چگونه در حافظه نهان جایگزین کنیم؟
- **برای پاسخ به این دو سوال، دو بحث اساسی مطرح می‌شود:**
  - سیاست جای‌دهی (Placement Policy)
  - سیاست جای‌گزینی (Replacement Policy)

سیاست‌های جایگزینی

- **جایگزینی ایده‌ال:** بلوکی را حذف کند که در آینده استفاده نشود
- سیاست جایگزینی در حافظه نهان نگاشت مستقیم لازم نیست
  - یک جا داریم اگر عنصر جدید برای آن آدرس بیاید باید جایگزین شود
- در حافظه‌های نهان انجمنی این بحث مهم می‌شود
  - با افزایش k اهمیت بیشتر هم می‌شود

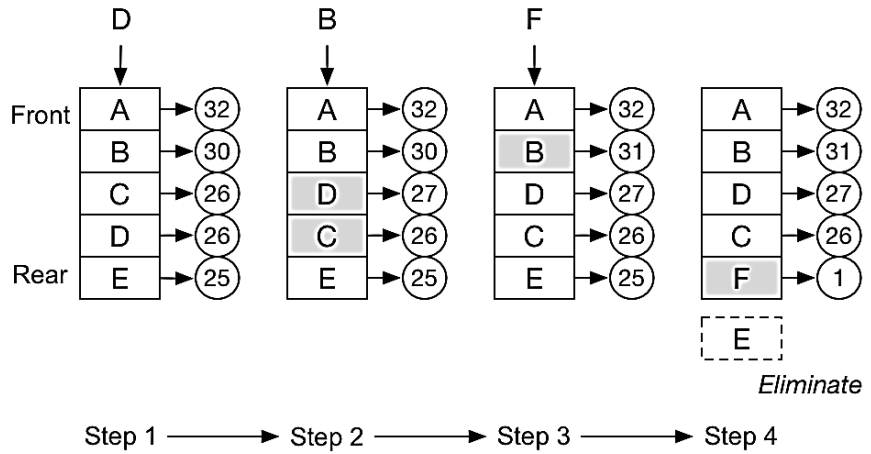
سیاست‌های جایگزینی

- **Random:** عنصری را به‌تصادف انتخاب کرده و جایگزین می‌کنیم
- **FIFO (First In First Out)**: عنصری که زودتر وارد شده را جایگزین کنیم
- **LIFO (Last In First Out)**: عنصری که دیرتر وارد شده را جایگزین کنیم
- **LRU (Least Recently Used)**: عنصری که اخیرا کمتر استفاده شده جایگزین شود

سیاست‌های جایگزینی

- **MRU (Most Recently Used)**: عنصری که اخیرا بیشتر استفاده شده جایگزین شود
- **LFU (Least Frequently Used)**: عنصری که تعداد دفعات کمتری استفاده شده جایگزین شود
- **MFU (Most Frequently Used)**: عنصری که تعداد دفعات بیشتری استفاده شده جایگزین شود

سیاست‌های جایگزینی



- **بهترین روش: LFU**
  - براساس تاریخچه تصمیم می‌گیرد
  - نیاز به شمارنده نامحدود دارد
  - پیاده‌سازی آن پیچیده و غیرممکن است

سیاست‌های جایگزینی

A B C D E D F

- **دومین بهترین روش: LRU**
  - نیاز به نگهداری Rank برای هر عنصر
    - نشانگر ترتیب ورود و همچنین دسترسی
  - با هر بار نوشتن جدید یا دسترسی، Rank افزایش می‌یابد
  - اشکال: نمیدانیم سقف اخیر بودن را چند بگذاریم
  - یک راه‌حل: درنظر گرفتن یک بیت برای هر داده
    - ریست کردن دوره ای این بیت
    - یک کردن زمان استفاده

سیاست‌های جایگزینی

- **سومین بهترین روش: Random**
  - پیاده‌سازی‌ها براساس LRU و Random انجام شده است
  - باقی روش‌ها کارایی مناسبی ندارند و در عمل استفاده نمی‌شوند