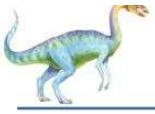
به نام خدا

تمرین پنجم تئوری سیستم عامل

فرزان رحمانی – ۹۹۵۲۱۲۷۱

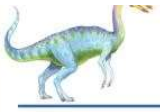
۱.

الف)



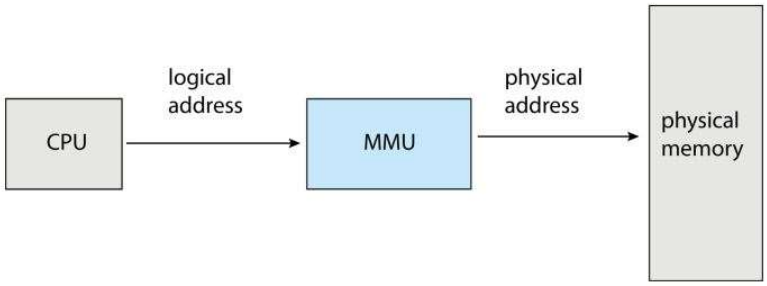
Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as **virtual address**
  - **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme
- **Logical address space** is the set of all logical addresses generated by a program
- **Physical address space** is the set of all physical addresses generated by a program



Memory-Management Unit (MMU)

- Hardware device that at run time maps virtual to physical address



- Many methods possible, covered in the rest of this chapter

What is a Logical Address?

An address induced by CUP when the program is executing is known as a logical address. It is assumed as a virtual address because it doesn't have a physical presence and that is why it is also known as a virtual address. A logical address may be distinct from a physical address because of the operation and process of an address translator

What is a Physical Address?

An address that helps in identifying the physical location of needed data is known as a physical address. We cannot use the physical address directly, to use it we have to access it by its corresponding logical address.

We can use the physical address space term for all the physical addresses corresponding to the logical addresses.

Difference between Logical Address and Physical Address in Operating System

S.No	Logical Address	Physical Address
1	Logical address is rendered by CPU.	Physical address is like a location that is present in the main memory.
2	It is a collection of all logical addresses rendered by the CPU.	It is a collection of all physical addresses mapped to the connected logical addresses.
3	Logical address of the program is visible to the users.	We cannot view the physical address of the program.
4	Logical address is generated by the CPU.	Physical address is computed by MMU.
5	We can easily utilise the logical address to access the physical address.	We can use the physical address indirectly.

ب) اگر page ها بزرگ باشند، باعث افزایش integral fragmentation می شود. ولی اندازه page table و تعداد page ها کوچک تر می شود و مدیریت آن و رزرو حافظه آن کمتر می شود و overhead محاسباتی کمتری دارد.
اگر pageها کوچک شوند، integral fragmentation کمتر می شود و باعث می شود از حافظه بهینه تر استفاده کنیم اما تعداد page ها و اندازه page table بیشتر می شود و overheadمحاسباتی زیادی دارد.

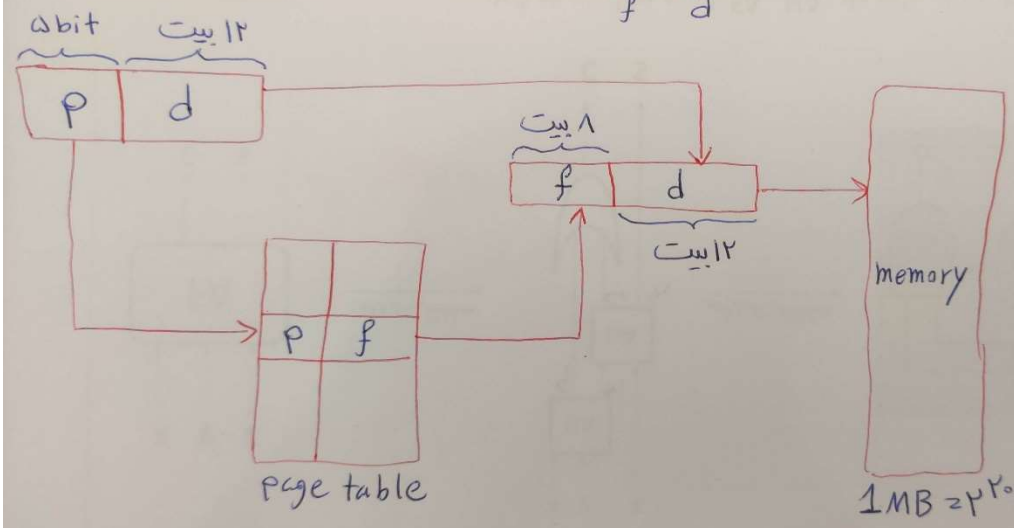
large pages:
pro - smaller page table, less page faults, less overhead in reading/writing of pages
con - more internal fragmentation, worse locality of reference
smaller pages:
pro - reduces internal fragmentation, better with locality of reference
con - bigger page table, more page faults, overhead in reading/writing of pages

۲) $4KByte = 2^2 \times 2^{10} Byte = 2^{12} Byte$ ← $d = 12$

2^{32} صفحه $= 2^5$ صفحه $\rightarrow p = 5$

فضای آدرس دهی $= 1MByte = 10^{20} Byte = 10^8 \times 10^{12} \rightarrow f = 8$

(f: 10^8، d: 10^{12})



اندازه هر درایه جدول 8bit است.

۳. با توجه به تعاریف الگوریتم ها به حل می پردازیم.

Dynamic Storage-Allocation Problem

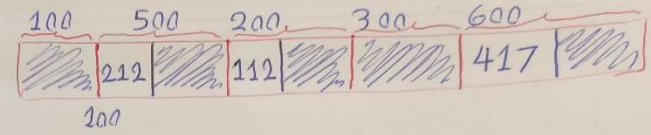
How to satisfy a request of size ***n*** from a list of free holes?

- **First-fit**: Allocate the ***first*** hole that is big enough
- **Best-fit**: Allocate the ***smallest*** hole that is big enough; must search entire list, unless ordered by size
  - Produces the smallest leftover hole
- **Worst-fit**: Allocate the ***largest*** hole; must also search entire list
  - Produces the largest leftover hole

First-fit and best-fit better than worst-fit in terms of speed and storage utilization

First-Fit:

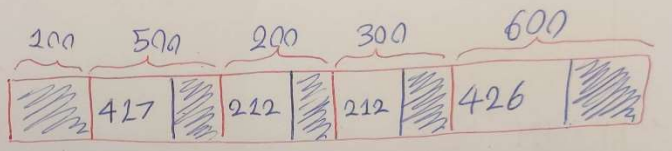
212K is put in 500k partition.
417K is put in 600k partition.
112K is put in 288k partition.
426K must wait.



(new partition 288K = 500K - 212K)

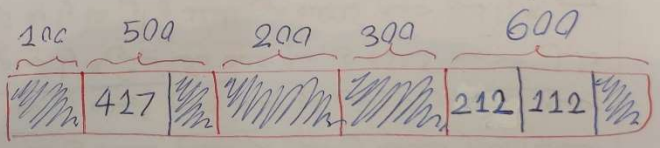
Best-Fit:

212K is put in 300K partition.
417K is put in 500k partition.
112K is put in 200K partition.
426K is put in 600K partition.



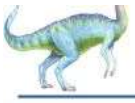
Worst-Fit:

212K is put in 600K partition.
417K is put in 500K partition.
112K is put in 388 partition.
426K must wait.



(new partition 388 = 600 - 212)

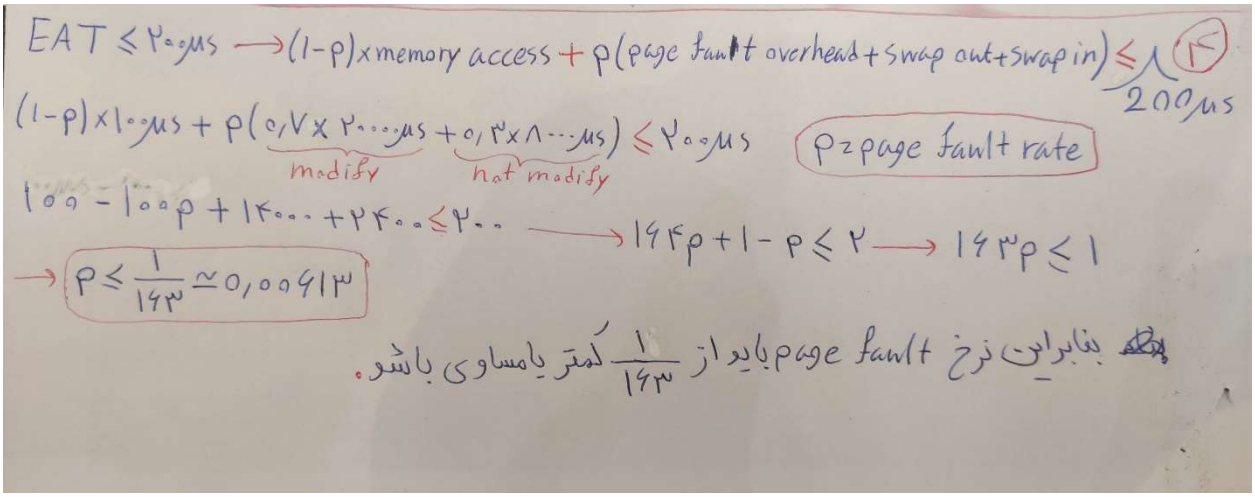
در این مثال الگوریتم Best-Fit منطقی تر و بهتر است ولی در عمل دارد overhead چرا که باید کل partition ها بررسی و سرچ شوند.



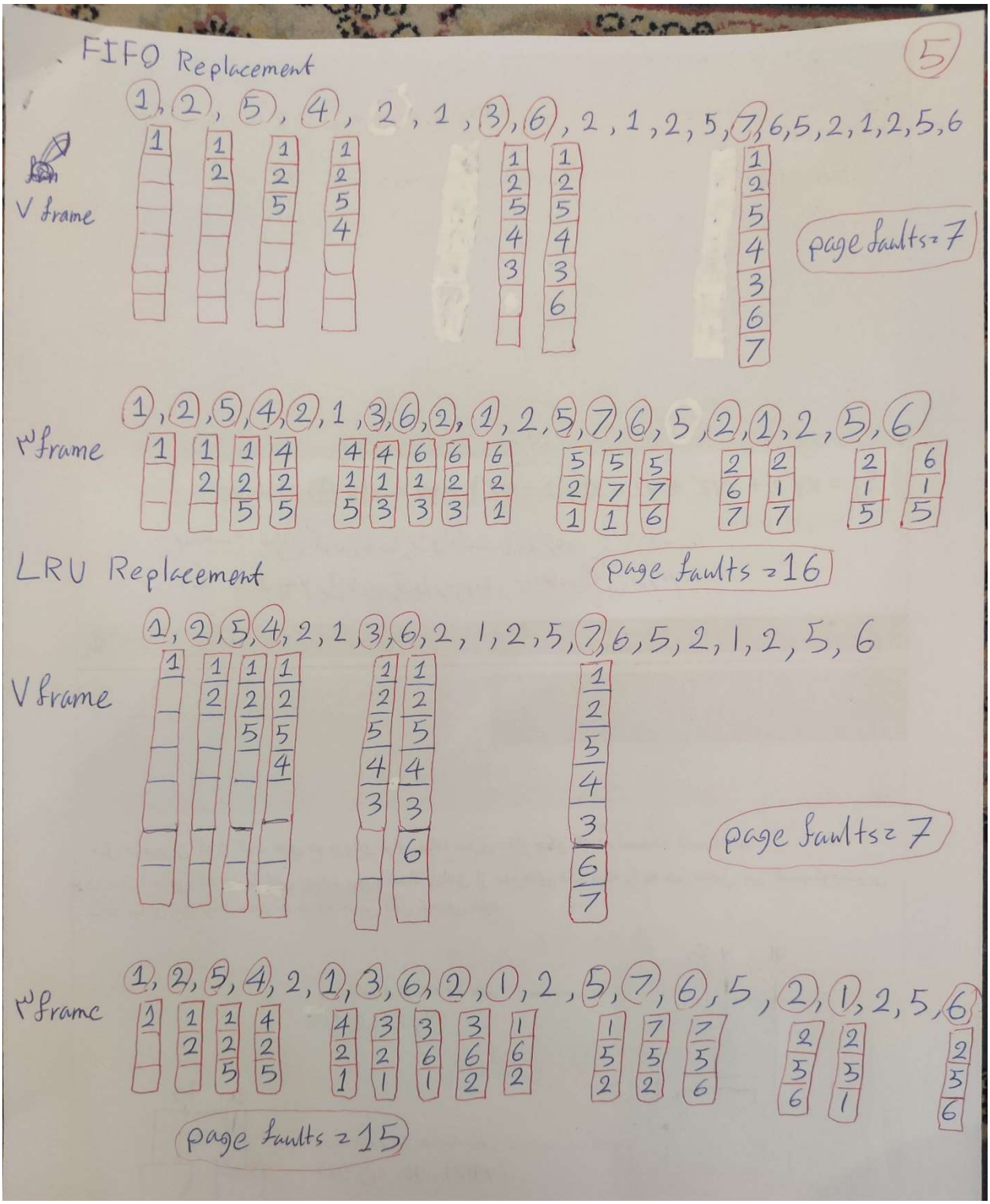
Performance of Demand Paging

- Three major activities
  - Service the interrupt – careful coding means just several hundred instructions needed
  - Read the page – lots of time
  - Restart the process – again just a small amount of time
- Page Fault Probability $0 \le p \le 1$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault
- Effective Access Time (EAT)

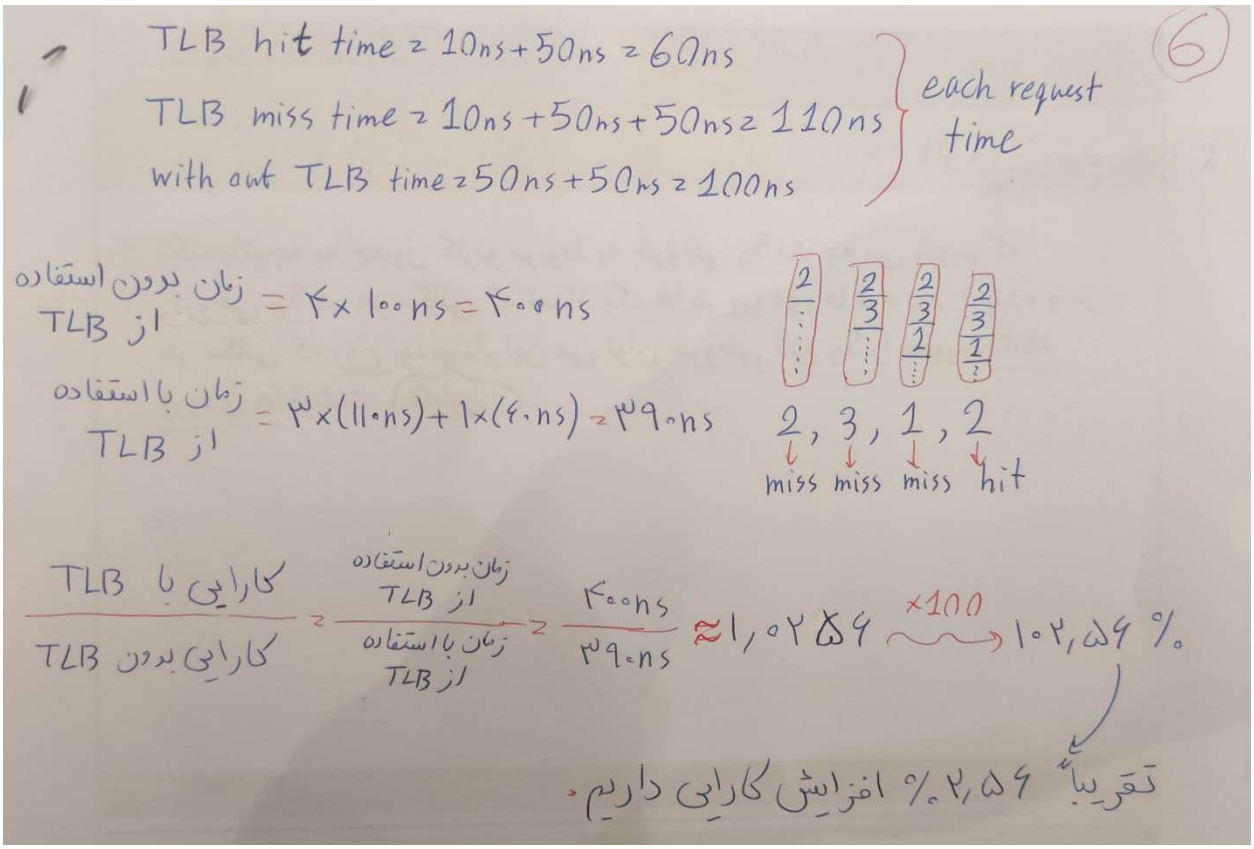
  EAT = (1 – p) x memory access
  + p (page fault overhead
  + swap page out
  + swap page in)



FIFO: Replace page that will not be used for longest period of time.
LRU: Replace page that has not been used in the most amount of time.



۶.


(6)
TLB hit time = 10ns + 50ns = 60ns
TLB miss time = 10ns + 50ns + 50ns = 110ns
with out TLB time = 50ns + 50ns = 100ns
each request time
زمان بدون استفاده از TLB = ۴ × ۱۰۰ns = ۴۰۰ns
زمان با استفاده از TLB = ۳ × (۱۱۰ns) + ۱ × (۶۰ns) = ۳۹۰ns
2, 3, 1, 2
miss miss miss hit
کارایی با TLB / کارایی بدون TLB = زمان بدون استفاده از TLB / زمان با استفاده از TLB = ۴۰۰ns / ۳۹۰ns ≈ ۱,۰۲۵۶ ×100 ۱۰۲,۵۶ %
تقریباً ۲,۵۶ % افزایش کارایی داریم.

پایان