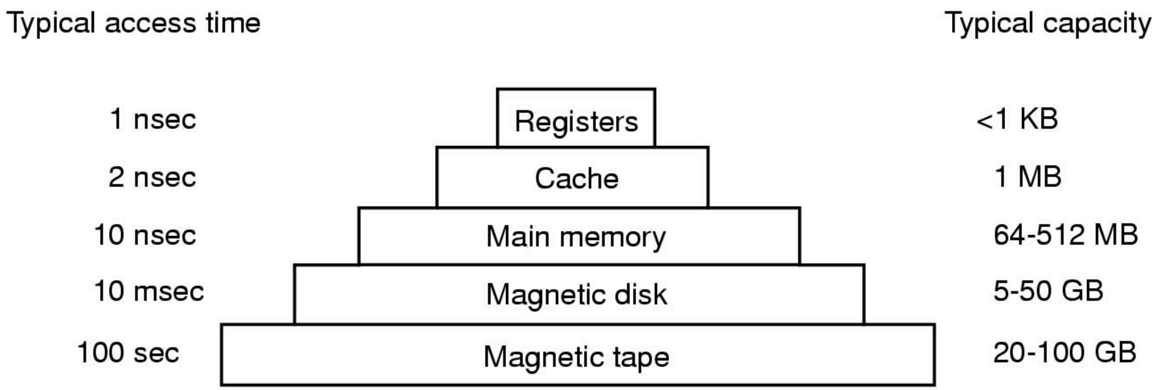
Operating Systems

Jinghui Zhong （钟竟辉）
Office：B3-515
Email ： jinghuizhong@scut.edu.cn

Memory Management

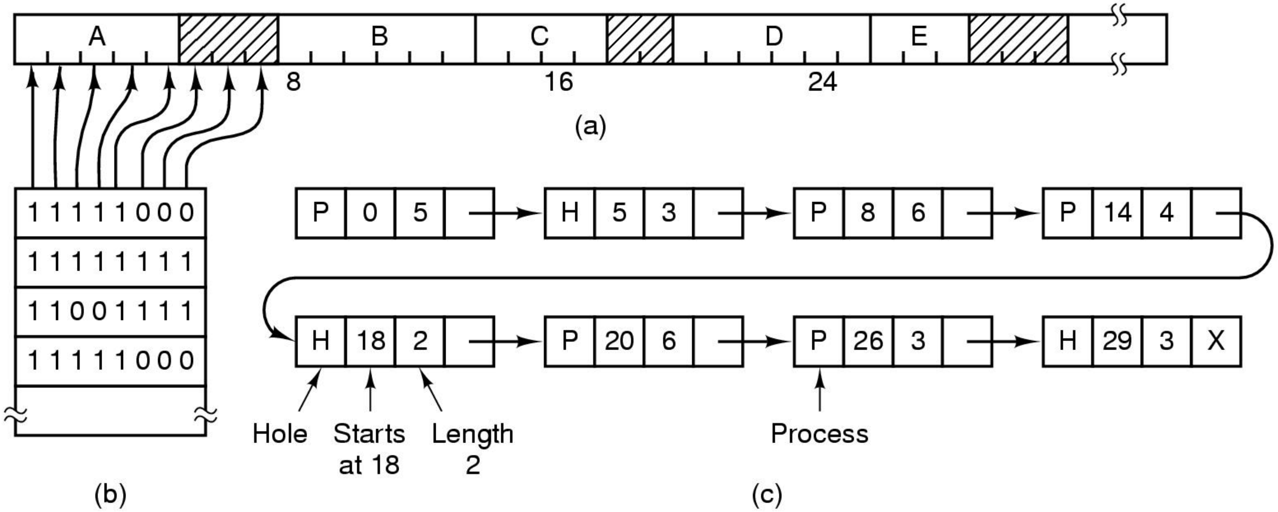
- Memory hierarchy
  - ✓small amount of fast, expensive memory – cache
  - ✓some medium-speed, medium price main memory
  - ✓gigabytes of slow, cheap disk storage



- Memory manager handles the memory hierarchy

Memory Management with Bit Maps & List

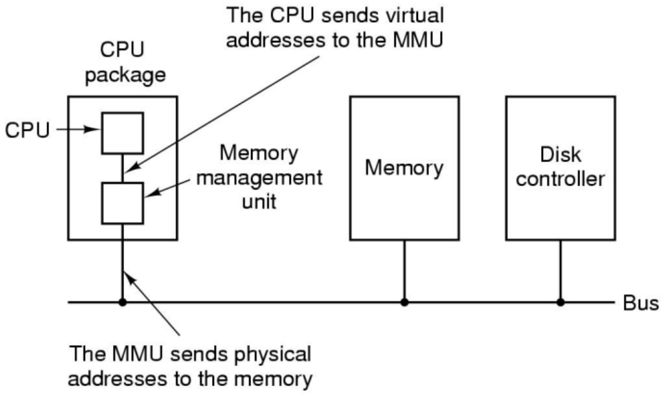
- Bit Maps & List method



- Drawback of bitmaps: to find consecutive 0 bits in the map is time-consuming

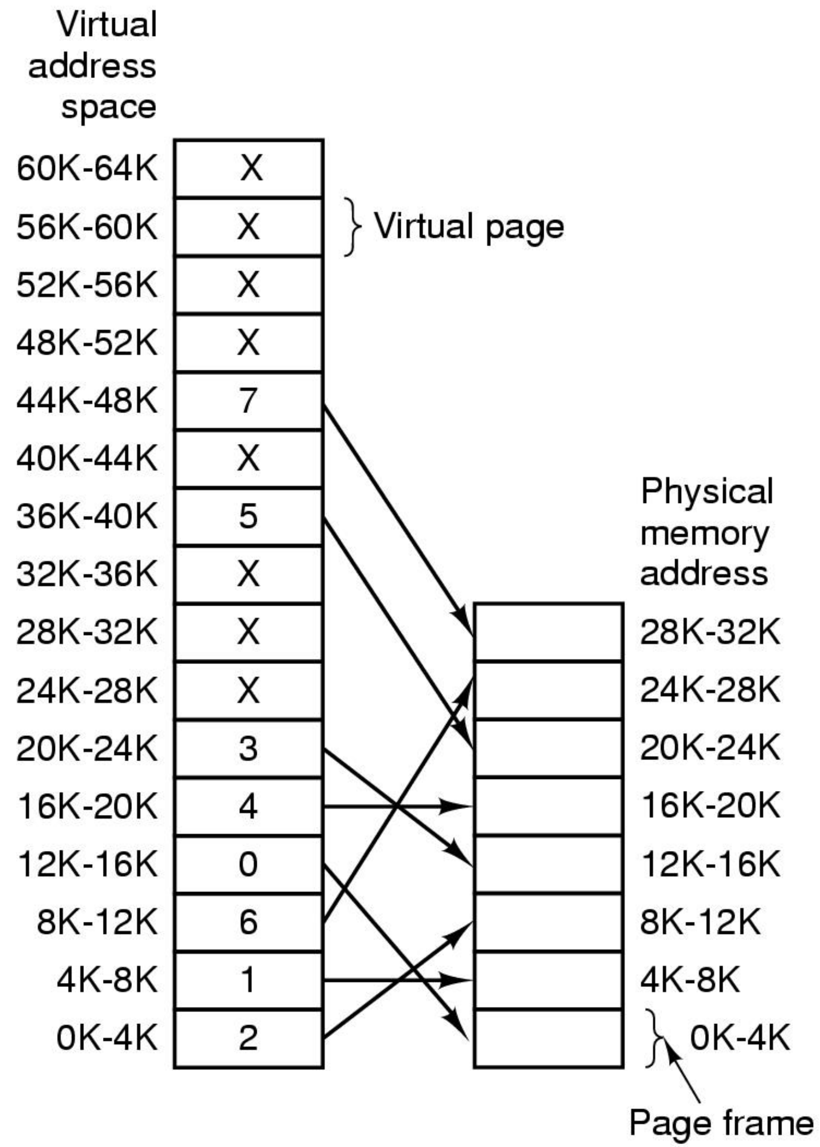
Virtual Memory

- Problem: Program too large to fit in memory
- **Virtual memory** - OS keeps the part of the program currently in use in memory
- **Paging** is a technique used to implement virtual memory.
- **Virtual Address** is a program generated address.
- The **MMU** (memory management unit) translates a virtual address into a physical address.

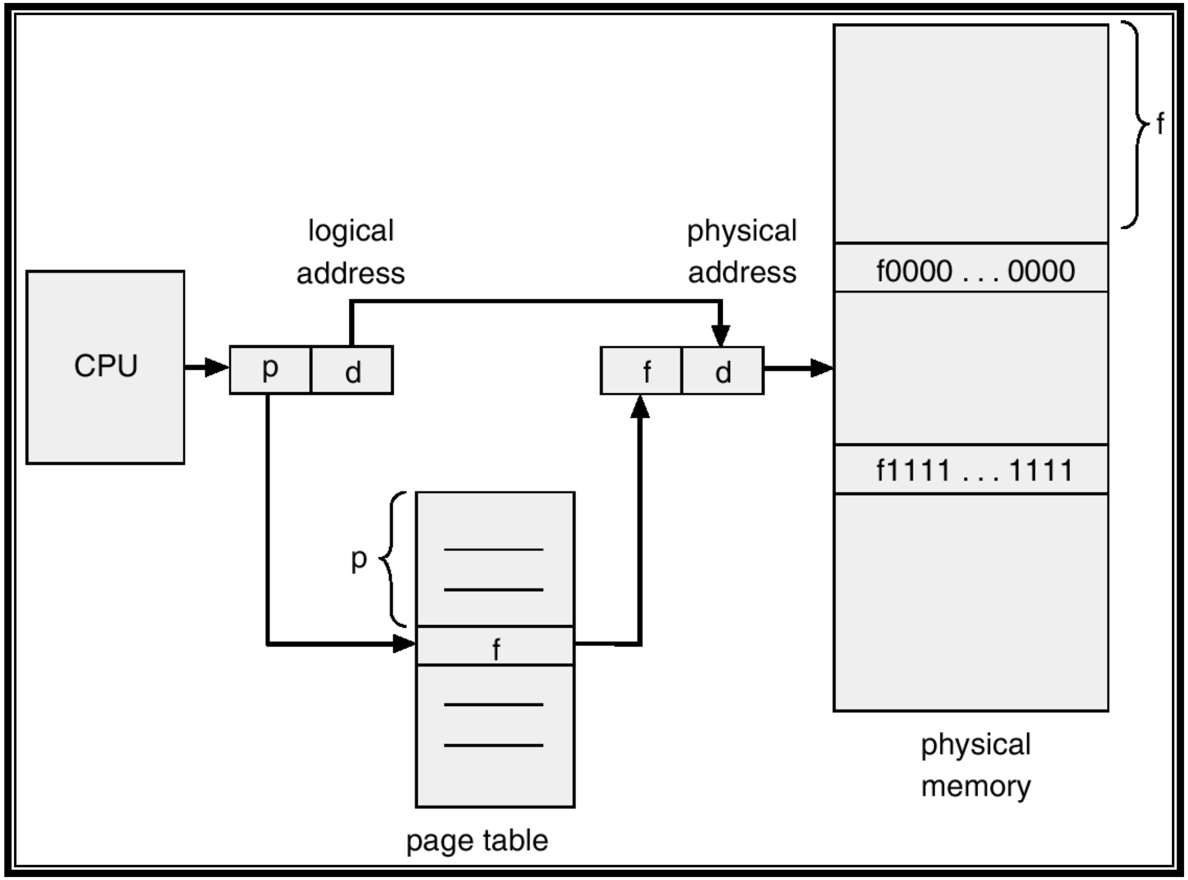


Page Table

The relation between virtual addresses and physical memory addresses given by page table

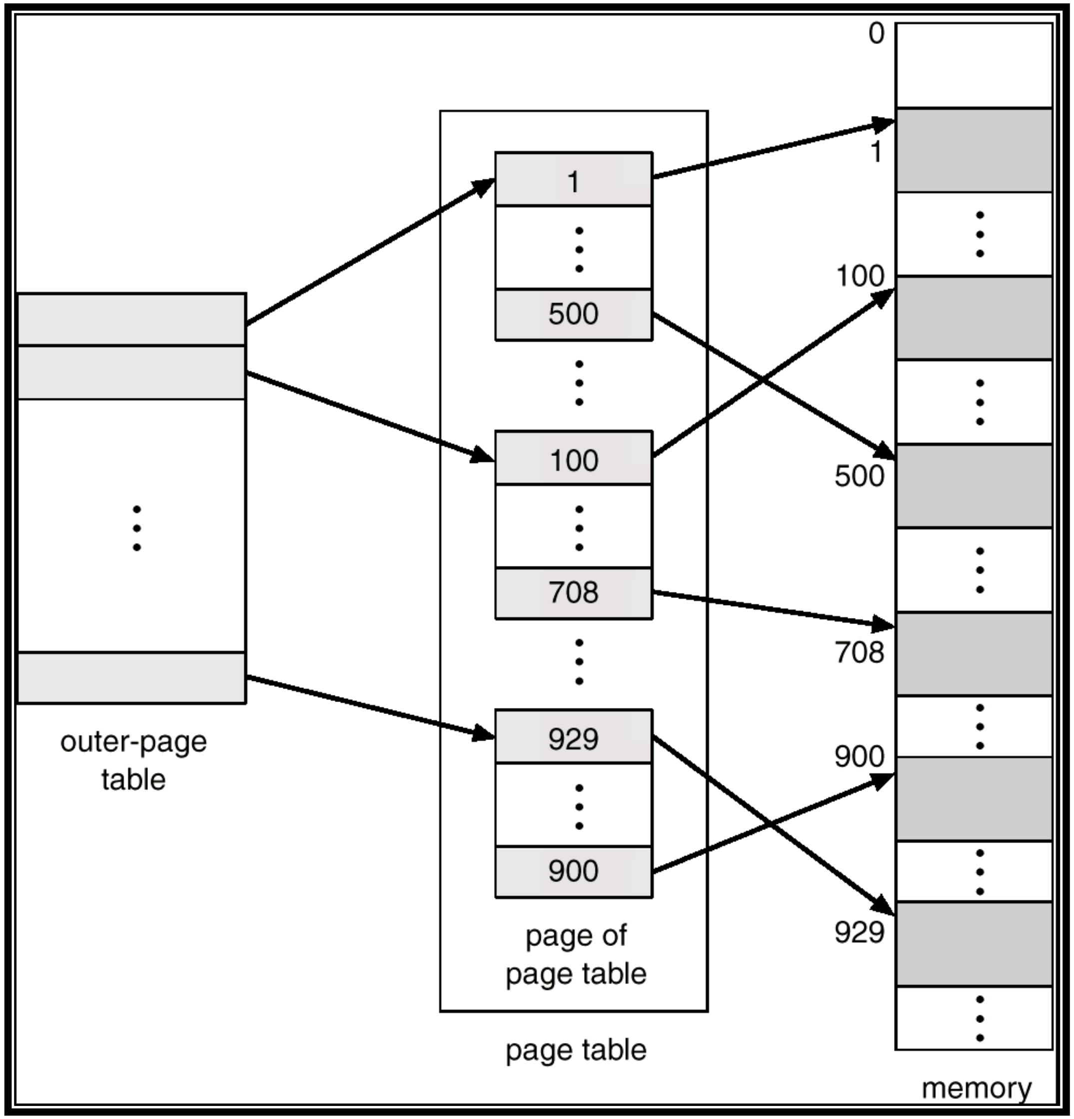


Pure paging



CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
f
p
f
page table
physical memory

Two-Level Page-Table Scheme

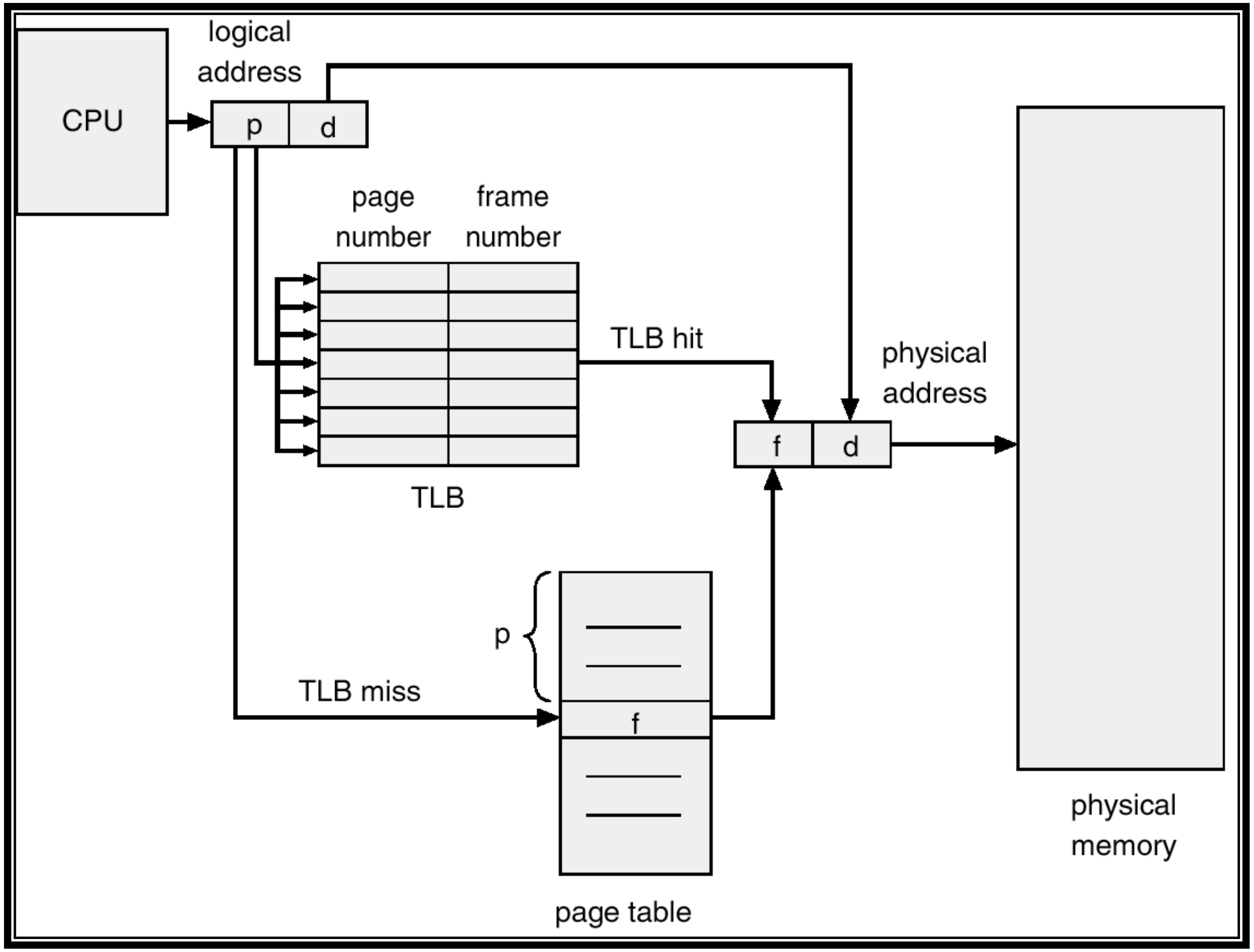


TLB

- **Observation:** Most programs make a large number of references to **a small number of pages**.

- **Solution:** Equip computers with a small hardware device, called **Translation Look-aside Buffer (TLB)** or **associative memory**, to map virtual addresses to physical addresses without using the page table.

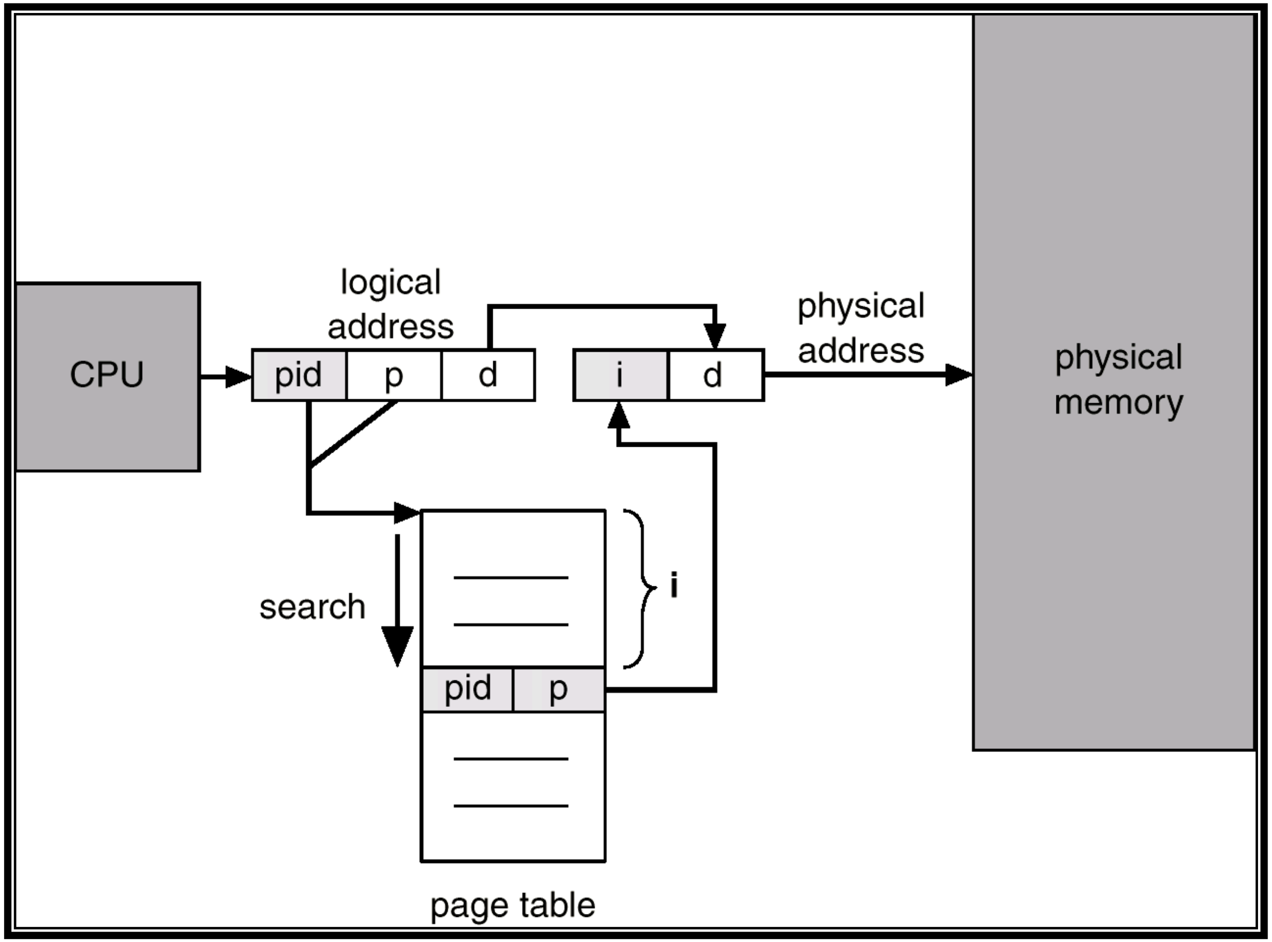
Paging Hardware With TLB



Inverted Page Table

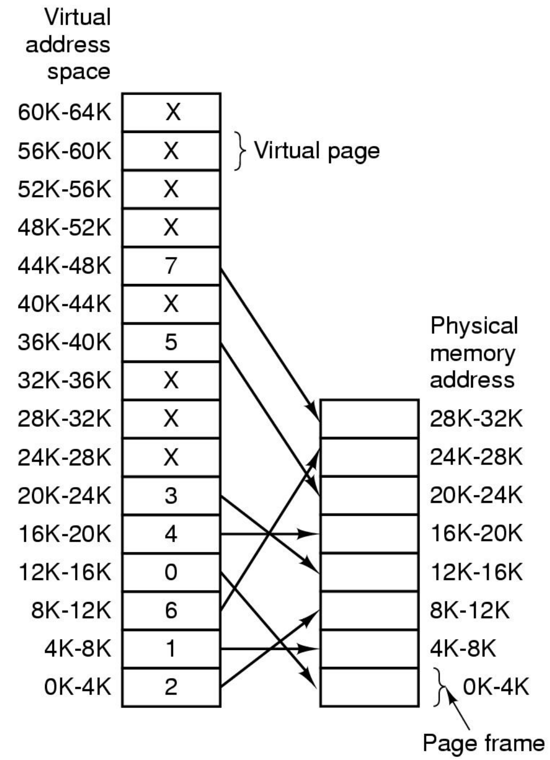
- Usually, each process has a page table associated with it. One of drawbacks of this method is that each page table may consist of millions of entries.
- To solve this problem, an **inverted page table** can be used. There is one entry for each real page (frame) of memory.
- Each entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page.

Inverted Page Table Architecture



Page Replacement Algorithms

- Page fault forces choice
  - ✓which page must be removed
  - ✓make room for incoming page
- Modified page must first be saved
  - ✓unmodified just overwritten
- Better not to choose an often used page
  - ✓will probably need to be brought back in soon



Not Recently Used Page Replacement Algorithm

- **Each page has Reference bit (R) and Modified bit (M).**
  - ✓ bits are set when page is referenced (read or written recently), modified (written to)
  - ✓ when a process starts, both bits R and M are set to 0 for all pages.
  - ✓ periodically, (on each clock interval (20msec)), the R bit is cleared. (i.e. R=0).
- **Pages are classified**
  - ✓ Class 0: not referenced, not modified
  - ✓ Class 1: not referenced, modified
  - ✓ Class 2: referenced, not modified
  - ✓ Class 3: referenced, modified
- **NRU removes page at random**
  - ✓ from lowest numbered non-empty class

Least Recently Used (LRU)

- **Assume pages used recently will used again soon**
  - ✓throw out page that has been unused for longest time
- **Software Solution?**

Must keep a linked list of pages: most recently used at front, least at rear; update this list every memory reference Too expensive!!

- **Hardware solution?**

Equip hardware with a 64 bit counter.

Least Recently Used (LRU)

- **Hardware solution:** Equip hardware with a 64 bit counter.
  - That is incrementing after each instruction.
  - The counter value is stored in the page table entry of the page that was just referenced.
  - choose page with lowest value counter
  - periodically zero the counter
  - Problem?

  page table is very large, become even larger.

- **Maintain a matrix of *n* x *n* bits for a machine with *n* page frames.**
  - ✓ When page frame *K* is referenced:
    - (i) Set row *K* to all 1s.
    - (ii) Set column *K* to all 0s.
  - ✓ The row whose binary value is smallest is the LRU page.

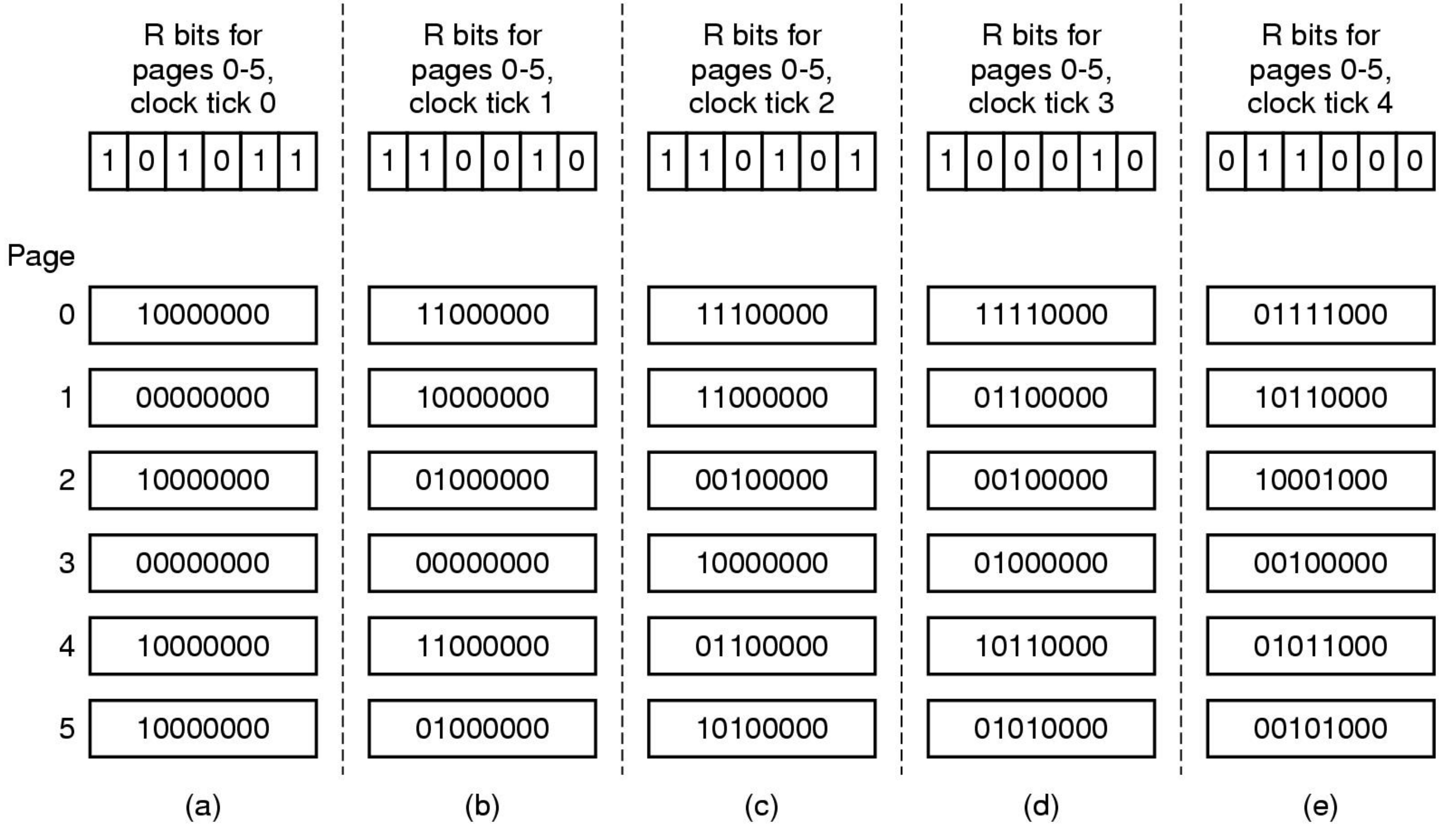
Simulating LRU in Software

- LRU hardware is not usually available. NFU (Not Frequently Used) is implemented in software.
  - ✓At each clock interrupt, the R bit is added to the counter associated with each page. When a page fault occurs, the page with the lowest counter is replaced.
  - ✓Difference? Problem?

  NFU never forgets, so a page referenced frequency long ago may have the highest counter.

- Modified NFU = NFU with Aging - at each clock interrupt:
  - ✓ The counters are shifted right one bit, and
  - ✓ The R bits are added to the leftmost bit.
  - ✓ In this way, we can give higher priority to recent R values.

Simulating LRU in Software



- The aging algorithm simulates LRU in software
- Note 6 pages for 5 clock ticks, (a) – (e)

Working-Set Model

- Pages are loaded only on demand. This strategy is called **demand paging**.
- During the phase of execution the process references relatively small fraction of its pages. This is called a **locality of reference**.
- The set of pages that a process is using currently is called its **working set**.
- A program causing page faults every few instructions is said to be **thrashing**.
- Paging systems keep each process's working set in memory before letting the process run. This approach is called the **working set model**.

Working-Set Model

- The idea is to examine the most recent page references. Evict a page that is not in the working set.
- The working set of a process is the set of pages it has referenced during the past τ seconds of **virtual time** (the amount of CPU time a process has actually used).
- Scan the entire page table and evict the page:
  - ✓R= 0, its age is greater than τ.
  - ✓R = 0, its age is not greater than τ and its age is largest.
  - ✓R = 1, randomly choose a page.
- The basic working set algorithm is expensive. Instead, WSCLock is used in practice.

Modeling Page Replacement Algorithms

- **Belady’s anomaly:**

More page frames might not always have fewer page faults.

All pages frames initially empty

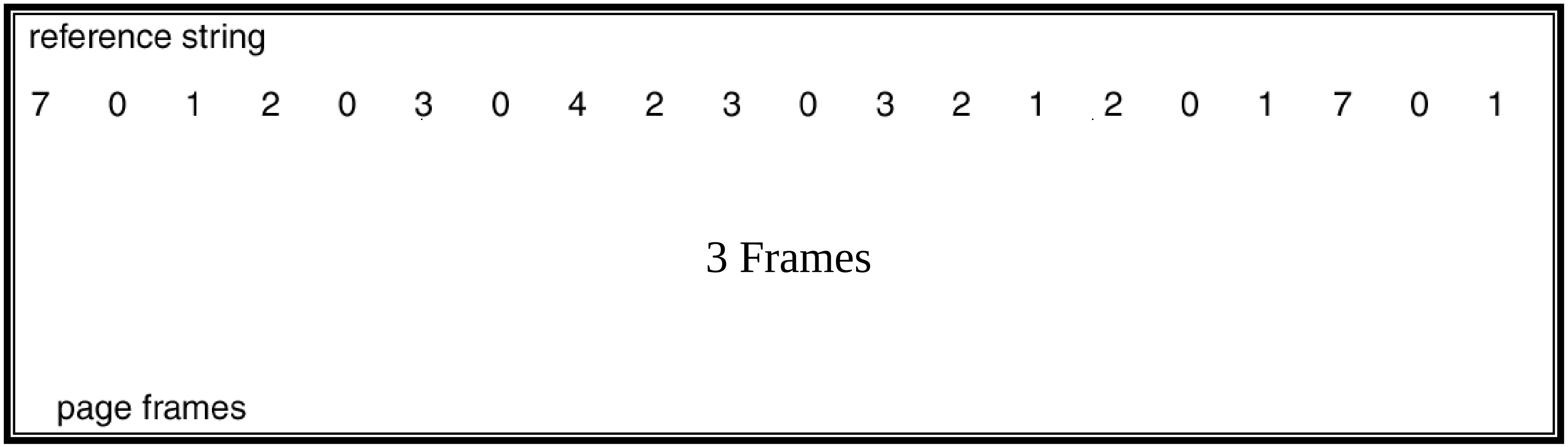
		0	1	2	3	0	1	4	0	1	2	3	4	
Youngest page		0	1	2	3	0	1	4	4	4	2	3	3	
			0	1	2	3	0	1	1	1	4	2	2	
Oldest page				0	1	2	3	0	0	0	1	4	4	
		P	P	P	P	P	P	P			P	P		9 Page faults

(a)

		0	1	2	3	0	1	4	0	1	2	3	4	
Youngest page		0	1	2	3	3	3	4	0	1	2	3	4	
			0	1	2	2	2	3	4	0	1	2	3	
Oldest page				0	1	1	1	2	3	4	0	1	2	
					0	0	0	1	2	3	4	0	1	
		P	P	P	P			P	P	P	P	P	P	10 Page faults

(b)

Page Replacement



- FIFO： 15 page faults
- LRU： 12 page faults

Page Size

- Small page size

✓**Advantages:**

less internal fragmentation

✓**Disadvantages:**

programs need many pages → larger page tables

Page Size

- Overhead due to page table and internal fragmentation

  s = average process size in bytes,

  p = page size in bytes,

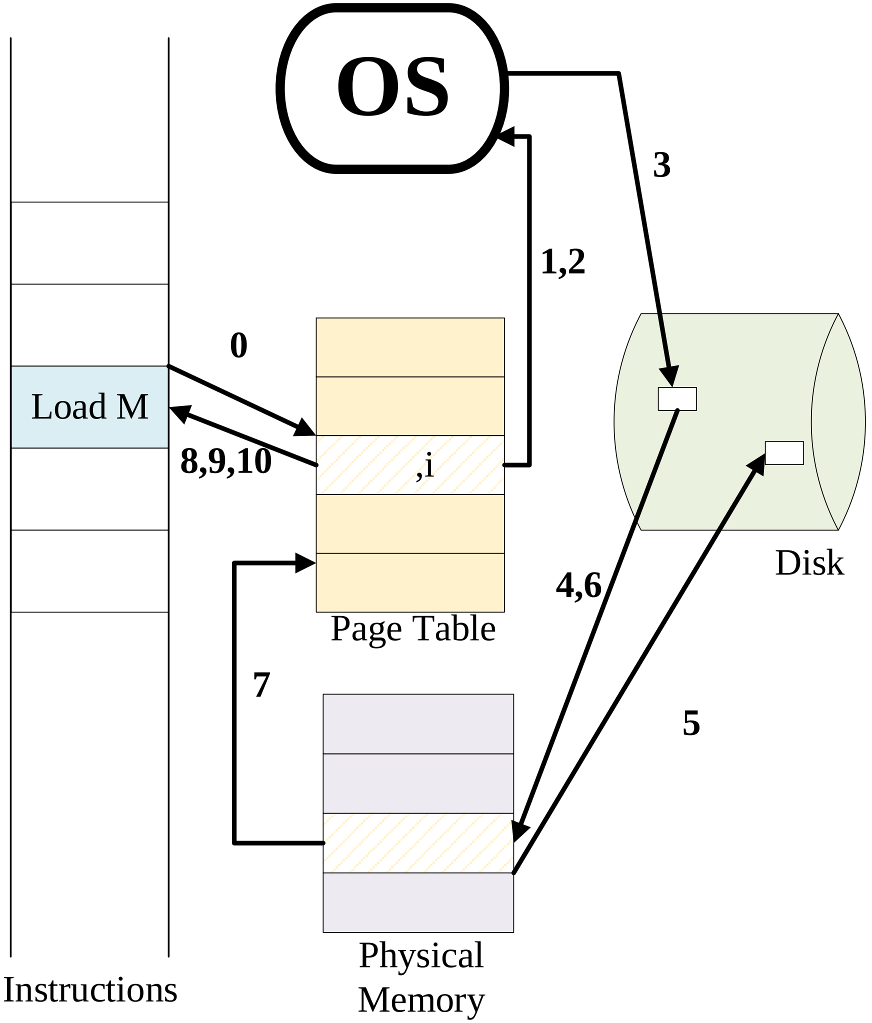
  e = page entry

$$overhead = \underbrace{\frac{s \cdot e}{p}}_{\text{page table space}} + \underbrace{\frac{p}{2}}_{\text{internal fragmentation}}$$

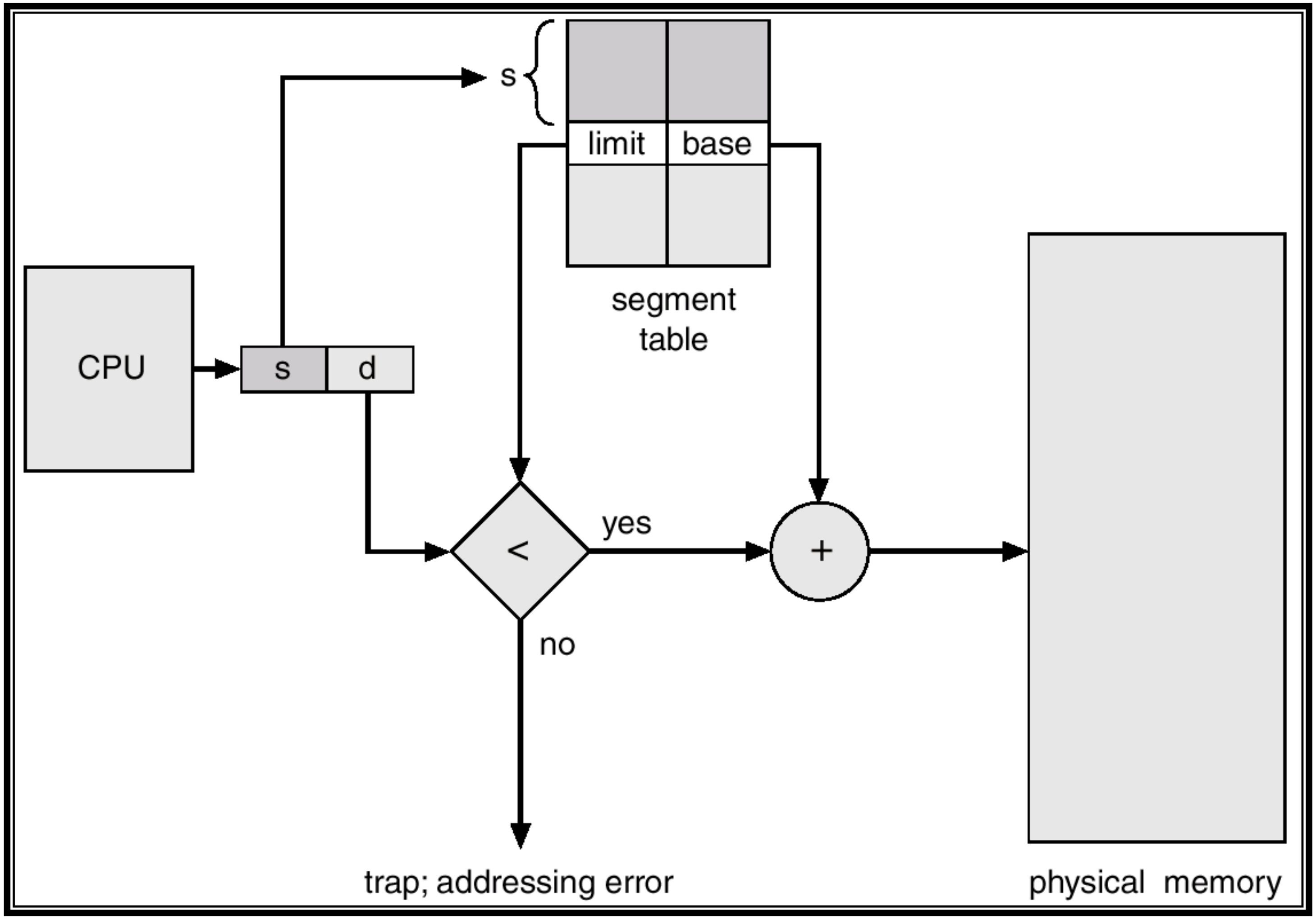
Optimized when $p = \sqrt{2se}$

Page Fault Handling

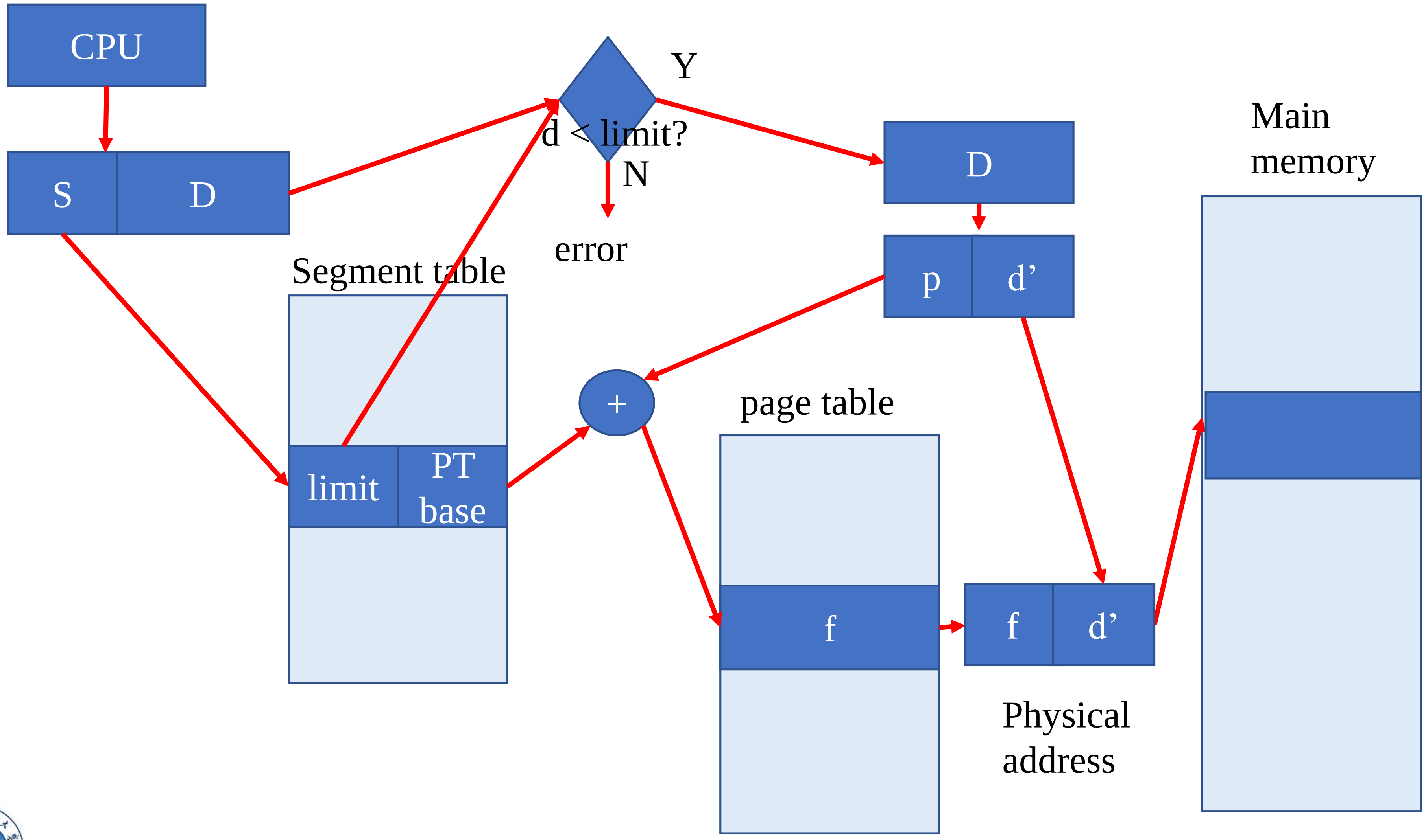
1. Hardware traps to kernel
2. Save general registers
3. Determines which virtual page needed
4. Seeks page frame
5. If the selected frame is dirty, write it to disk
6. Brings new page in from disk
7. Page tables updated
8. Instruction backed up to when it began
9. Faulting process scheduled
10. Registers restored & Program continues



Segmentation



Segmentation with Paging (MULTICS)



File Access

- **Sequential access**
  - ✓read all bytes/records from the beginning
  - ✓cannot jump around
  - ✓convenient when medium was magnetic tape
- **Random access**
  - ✓bytes/records read in any order
  - ✓essential for database systems

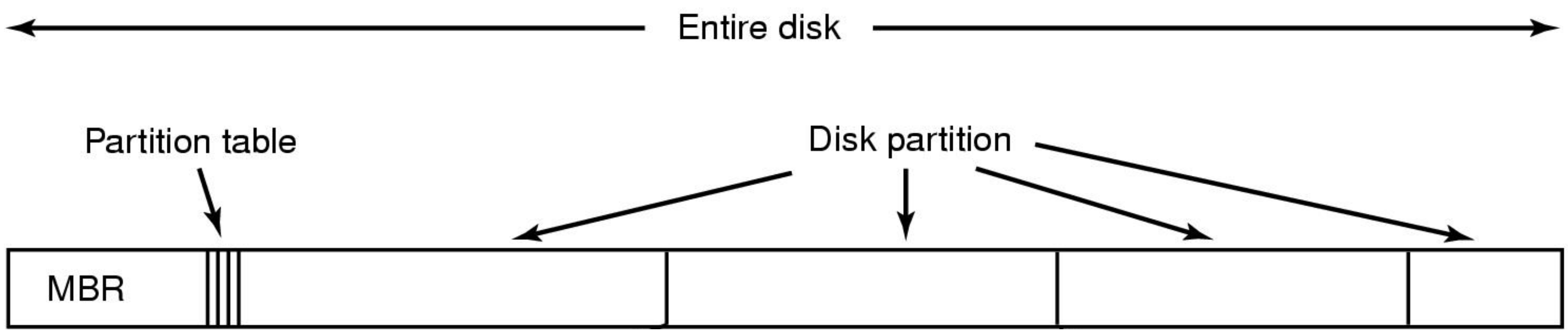
Directories

- **File systems have directories or folders to keep track of files.**
- **Two different methods are used to specify file names in a directory tree:**
  ① **Absolute path name** consists of the path from the root directory to the file. e.g., cp /usr/ast/mailbox /usr/ast/mailbox.bak

  ② **Relative path name** consists of the path from the current directory (**working directory**). e.g, cp ../lib/dictionary ➔ cp /usr/lib/dictionary

File System Layout

- **File system layout:**
  - MBR (Master Boot Record) is used to boot the computer.
  - The partition table gives the starting and ending addresses of each partition.



File System Layout

① **Boot block:** read in by the MBR program when the system is booted.

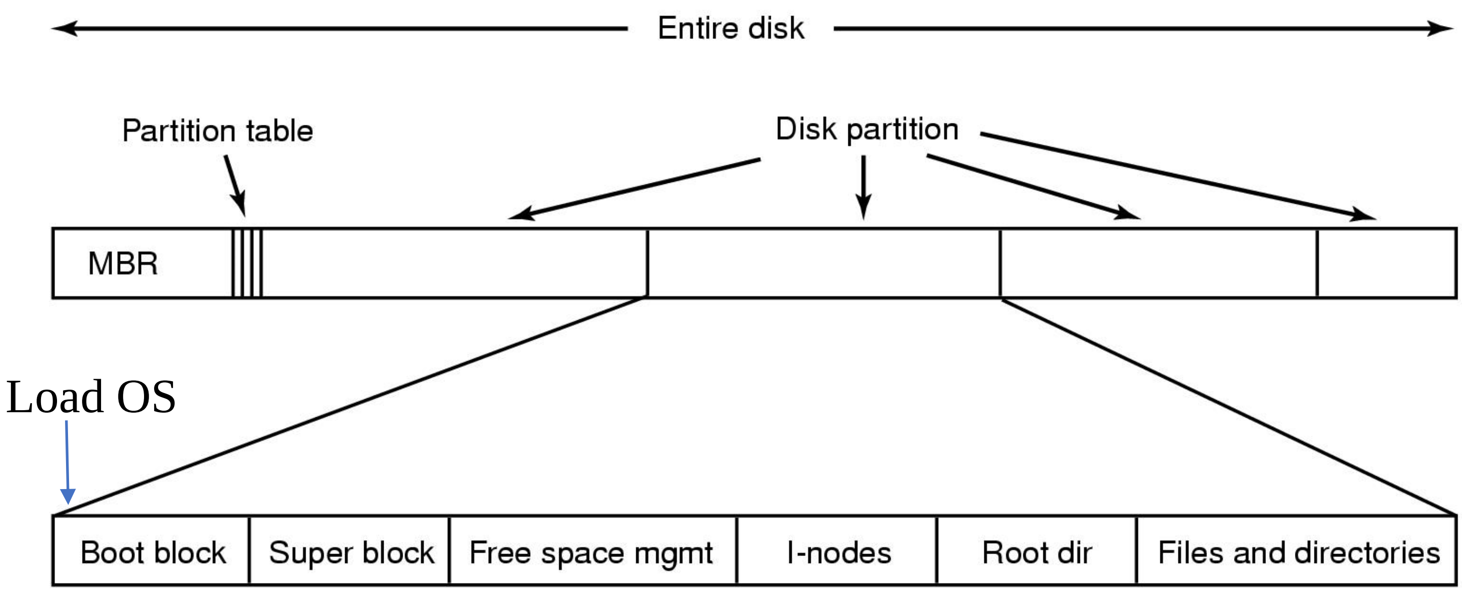
② **Superblock:** contains the key parameters about the file system.

③ Free blocks information

④ I-nodes tells all about the file.

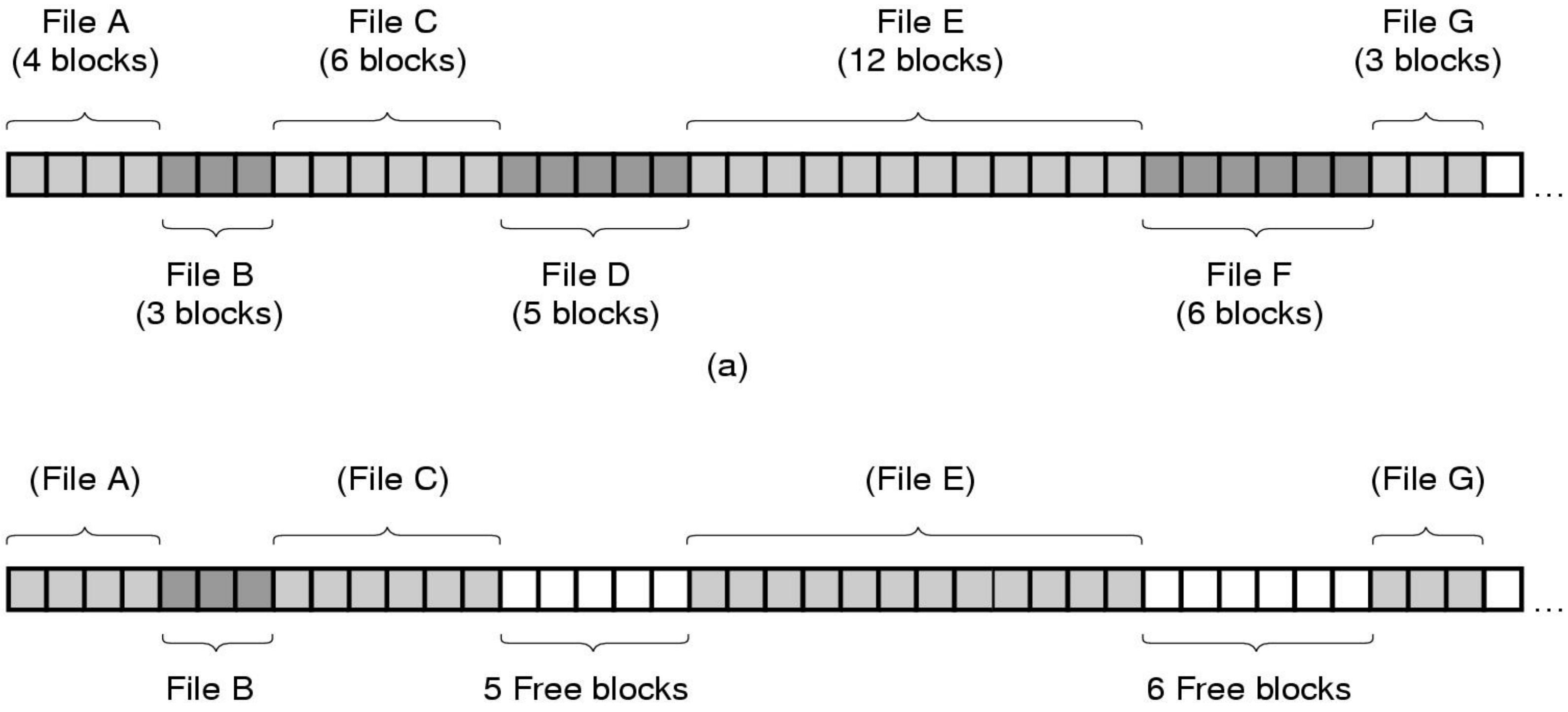
⑤ Root directory

⑥ Directories and files



File Allocation

- **Contiguous Allocation:**
  store each file as contiguous block of data.



File Allocation

- **Contiguous Allocation**
  - **Advantages:**

    Simple to implement;

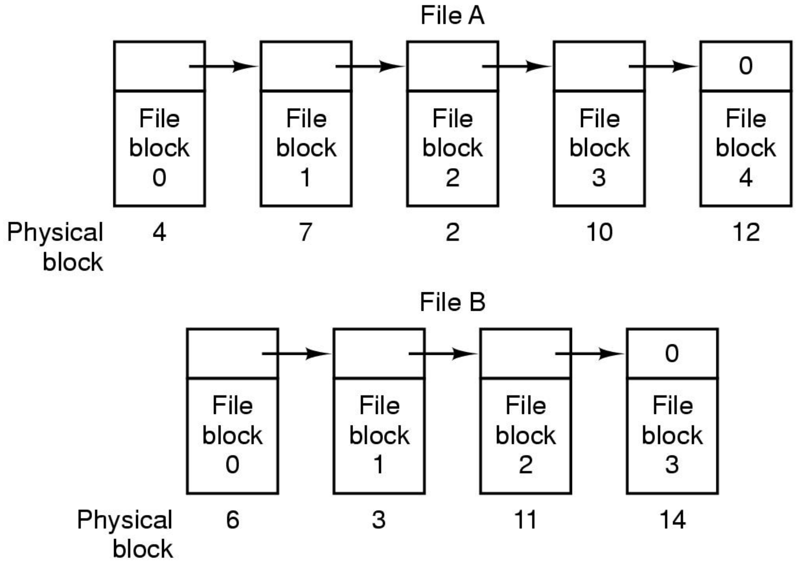
    Read performance is excellent.

  - **Disadvantages:**
    - ✓Disk fragmentation
    - ✓The maximum file size must be known when file is created.
  - **Example:** CD-ROMs, DVDs

File Allocation

- **Linked List Allocation:**

  keep linked list of disk blocks



- **Disadvantages ?**

  ① Slow random access speed

  ② The amount of data in a block is not a power of 2

Linked List Allocation using an index

- Take table pointer word from each block and put them in an index table, **FAT (File Allocation Table**) in memory.

☐ Advantages?

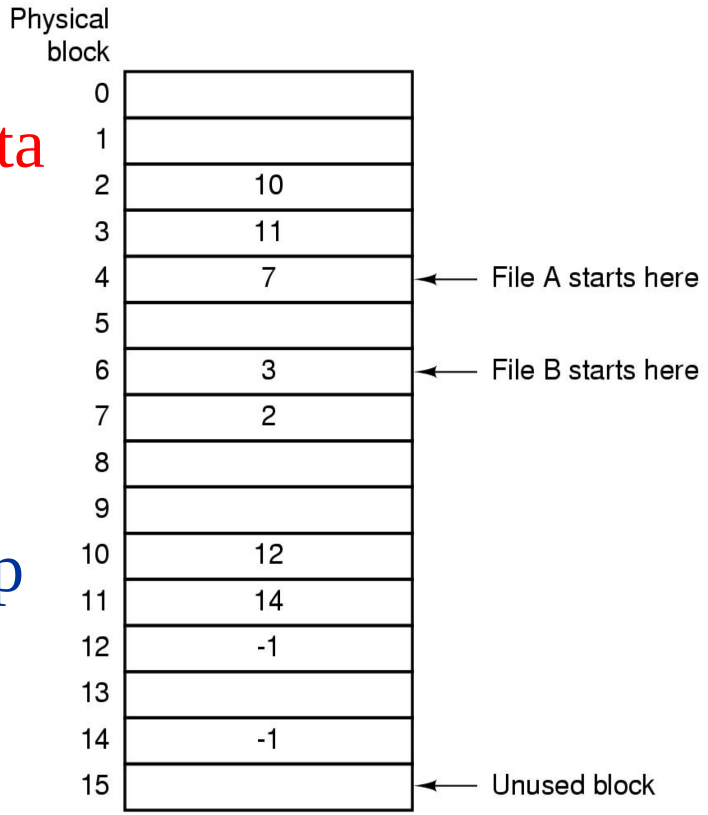
① The entire block is available for data

② Stored in memory, fast

☐ Drawbacks?

Occupies a large amount of memory.

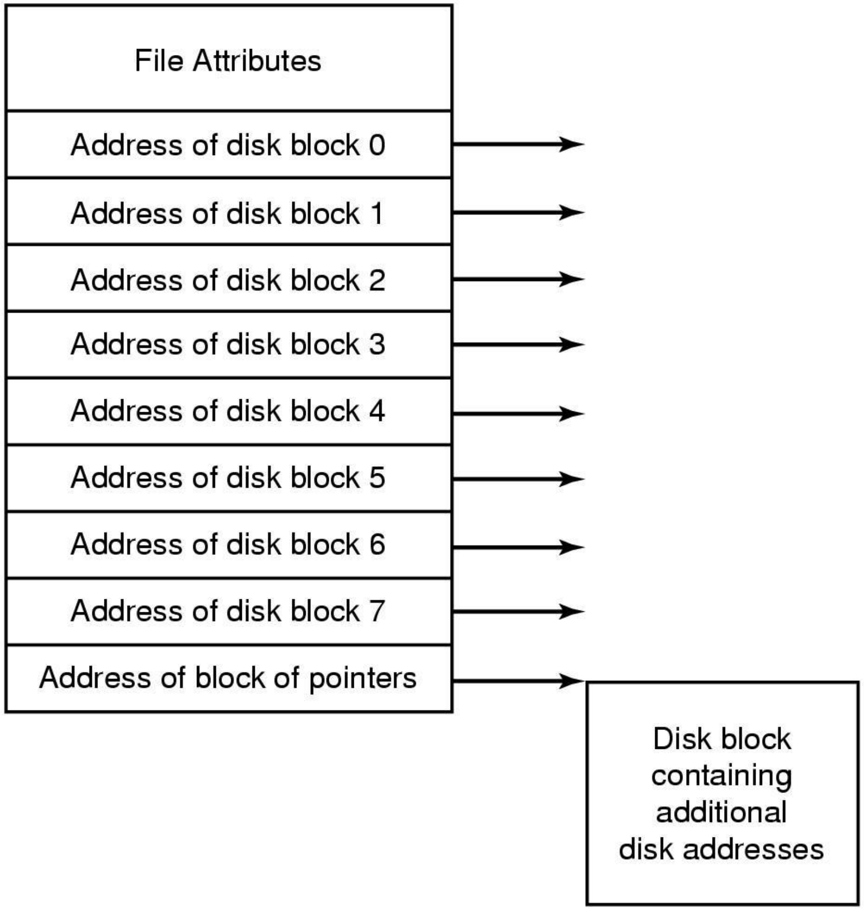
For 200-GB disk, the table will take up 600M or 800 M memory.



File System Implementation

- I-node (index-node):

  lists the attributes and disk addresses of the file's blocks.



Virtual File Systems

- **Definition:** the Virtual File System (or the **Virtual Filesystem Switch**) is the software layer in the kernel that provides the filesystem interface to user space programs.
- **Same API for different types of file systems**

① Separates file-system generic operations from implementation details

② Syscalls program to VFS API rather than specific FS interface

Disk space management

- **Strategies for storing an *n* byte file:**

  ① Allocate n consecutive bytes of disk space

  If the file grows it will have to be moved on the disk, it is an expensive operation and causes external fragmentation.

  ② Allocate a number [n/k] blocks of size k bytes each

  Blocks do not need to be adjacent.

How to determine block size?

- **When block size increase, disk space utilization decrease**

  Internal fragmentation, space efficiency decrease

- **When block size decrease, data transfer rate decrease**

  Time efficiency decrease

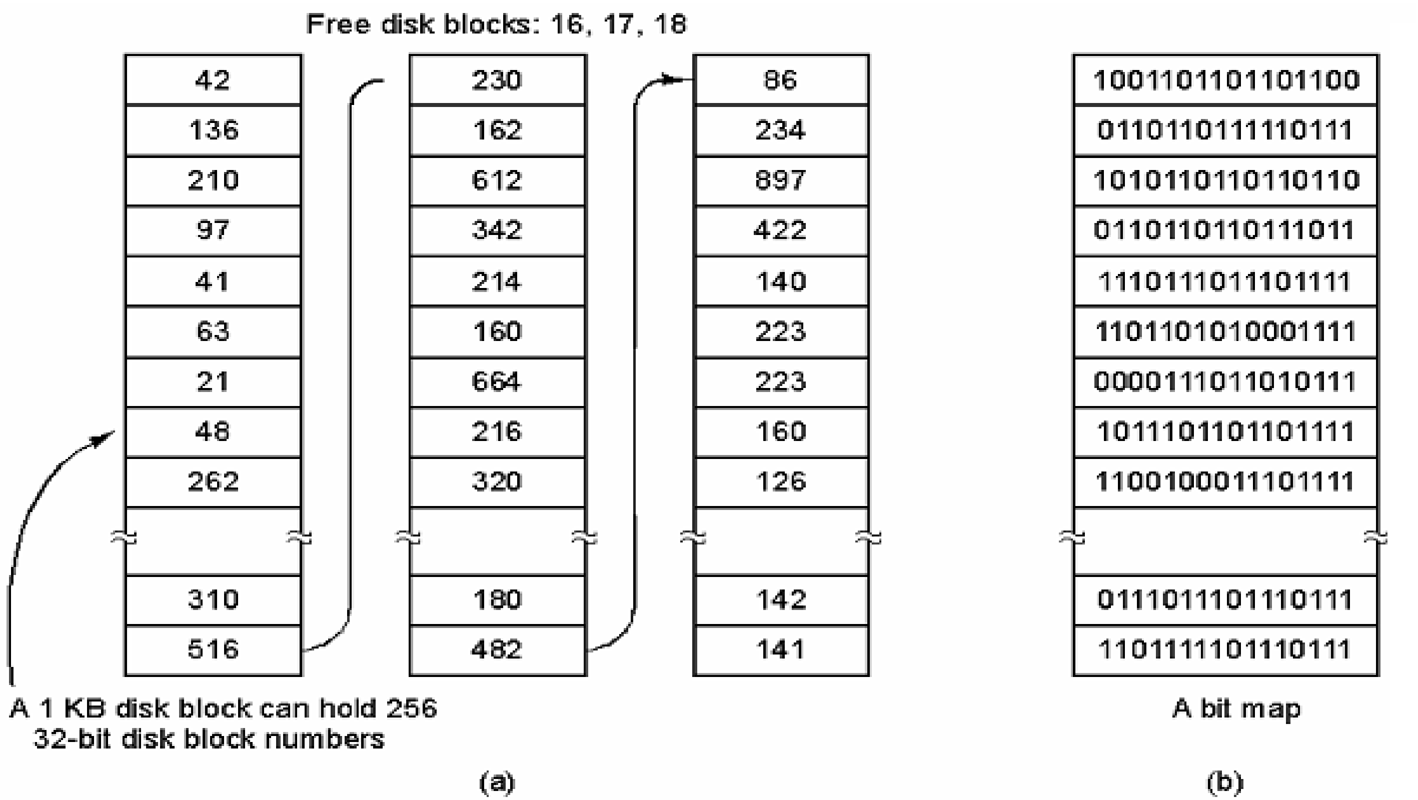
  usual size k = 512bytes, 1k (UNIX), or 2k

Keeping Track of Free Blocks

- **Use linked list of disk blocks:**

With 1 KB block and 32-bit disk block number.

- **Use bit-map:** Free blocks ->1, Allocated blocks -> 0



(a) (b)

Keeping Track of Free Blocks

- **Use linked list of disk blocks**: Each block holds as many free disk block numbers as will fit.

  With 1 KB block and 32-bit disk block number ➔ 1024 * 8/32 = 256 disk block numbers ➔ 255 free blocks (and) 1 next block pointer.

- **Use bit map**: A disk with (n) blocks requires a bitmap with (n) bits

  ✓Free blocks are represented by 1's

  ✓Allocated blocks represented by 0's

  ✓16GB disk has 2^{24} 1-KB and requires 2^{24} bits ➔ 2048 blocks

  ✓Using a linked list = $2^{24}/255$ = 65793 blocks.

File System Backup

- Backups are made to handle: recover from disaster or stupidity.
- Considerations of backups
  - ✓Entire or part of the file system
  - ✓**Incremental dumps**: dump only files that have changed
  - ✓Compression
  - ✓Backup an active file system
  - ✓Security

File System Backup

- **Two strategies for dumping a disk:**
  ① **Physical dump**: starts at block 0 to the last one.
  Advantages: simple and fast
  Disadvantages: backup everything
  ② **Logical dump**: starts at one or more specified directories and recursively dumps all files and directories found that have changed since some given base date.

File System Consistency

- Most OS have a utility program, called a file system checker, to test the consistency of a file system.

E.g., *fsck* in UNIX, *sfc* in Windows

- Two types of consistency checks can be made:

  (a) block consistency

  (b) file consistency

I/O Device

- **Two common kinds of I/O devices:**

  ① **Block device:** stores information in fixed-size blocks.

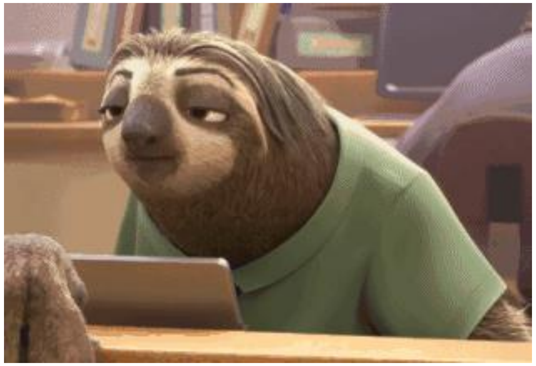
  ② **Character device**： delivers or accepts a stream of characters, without regard to any block structure.

- Special device: e.g., clocks.

Principles of I/O Hardware

- **I/O devices cover a huge range in speeds**

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

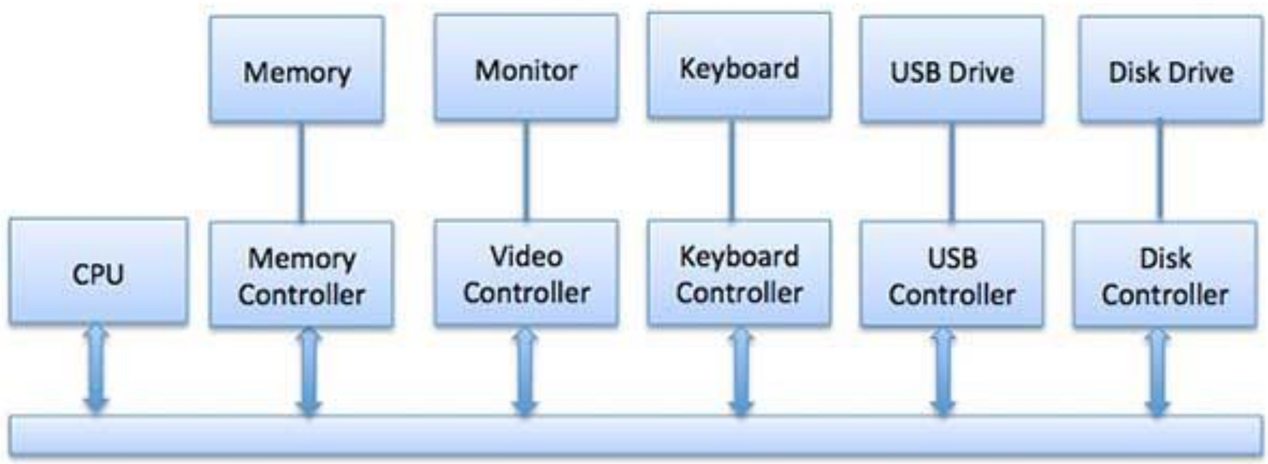


Device Controllers

- **Components of I/O devices:**

① Mechanical component ;

② Electronic component：i.e., device controller



Device Controllers

- A device controller is a part of a computer system that makes sense of the signals going to, and coming from the CPU.

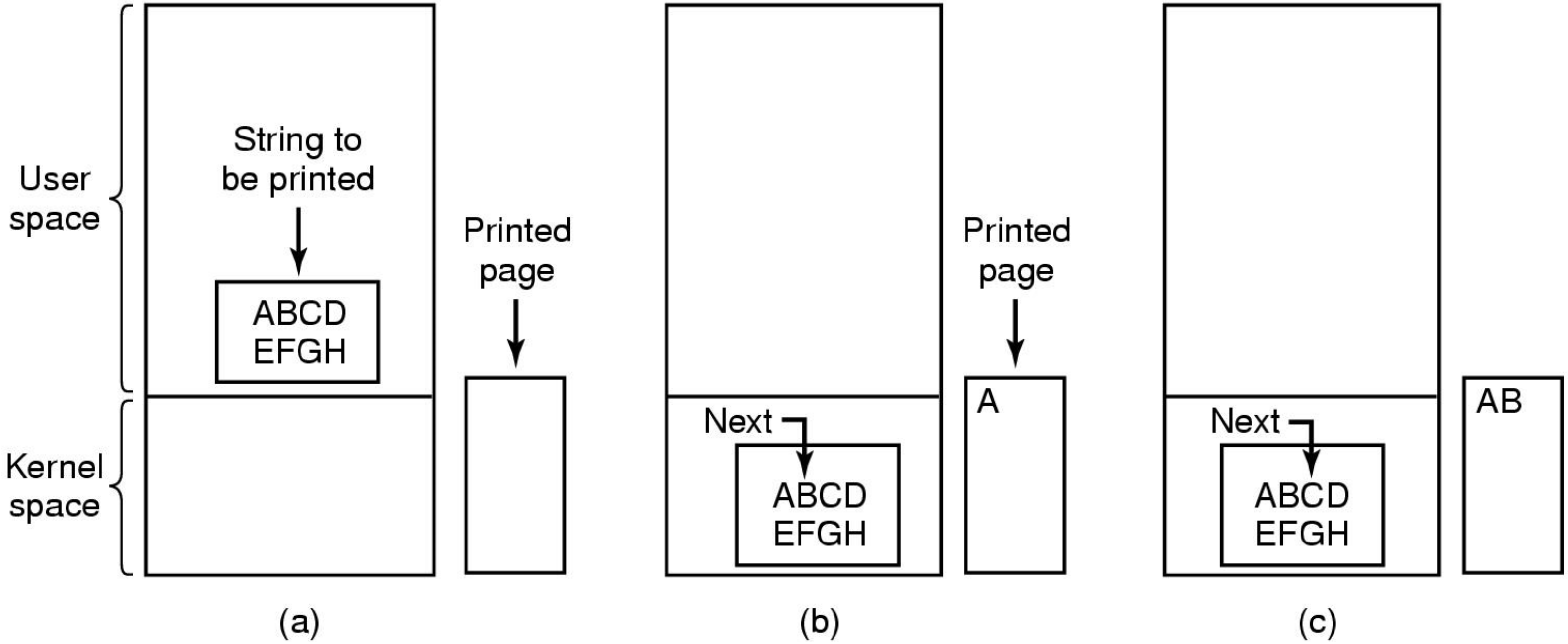
- Each device controller has a **local buffer** and some **registers**. It communicates with the CPU by interrupts. A device's controller plays as a bridge between the device and the operating system.

Programmed I/O

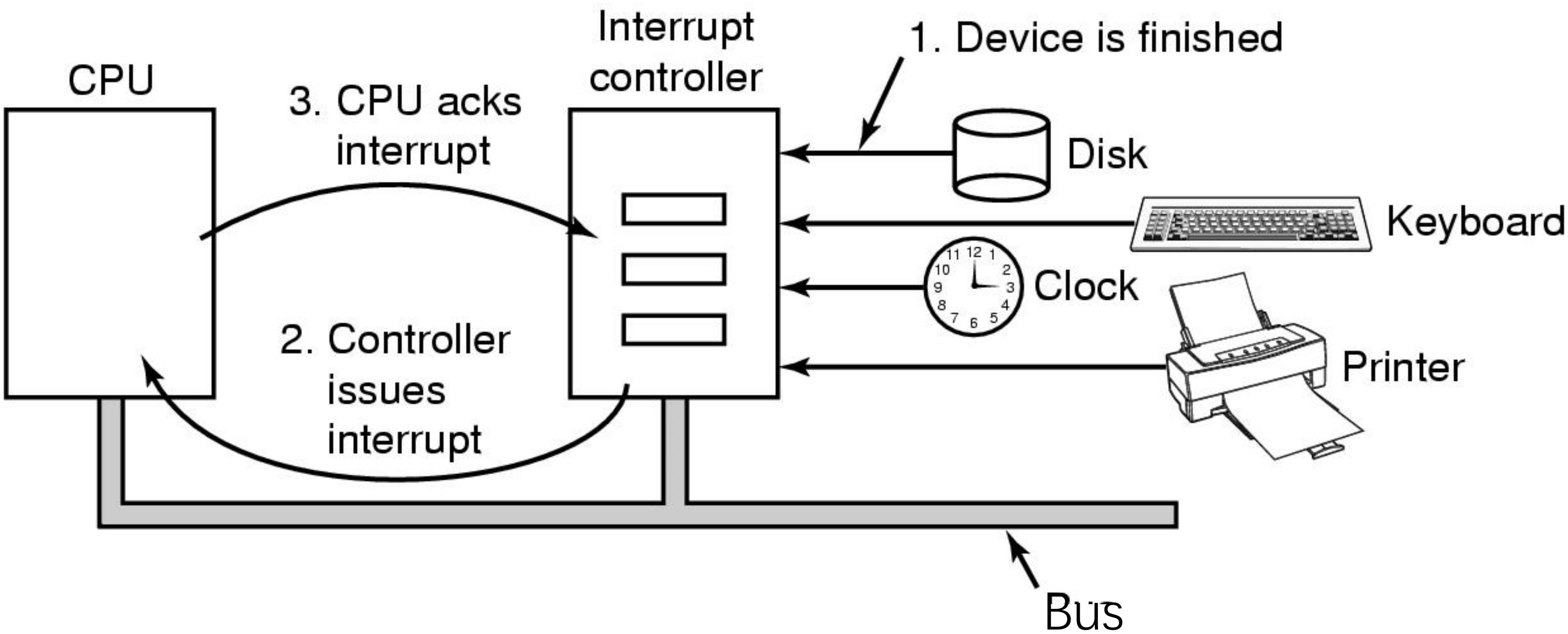
●Programmed input/output (PIO)

① A method of transferring data between the **CPU** and a peripheral.

② Software running on the CPU uses instructions to perform data transfers to or from an I/O device.



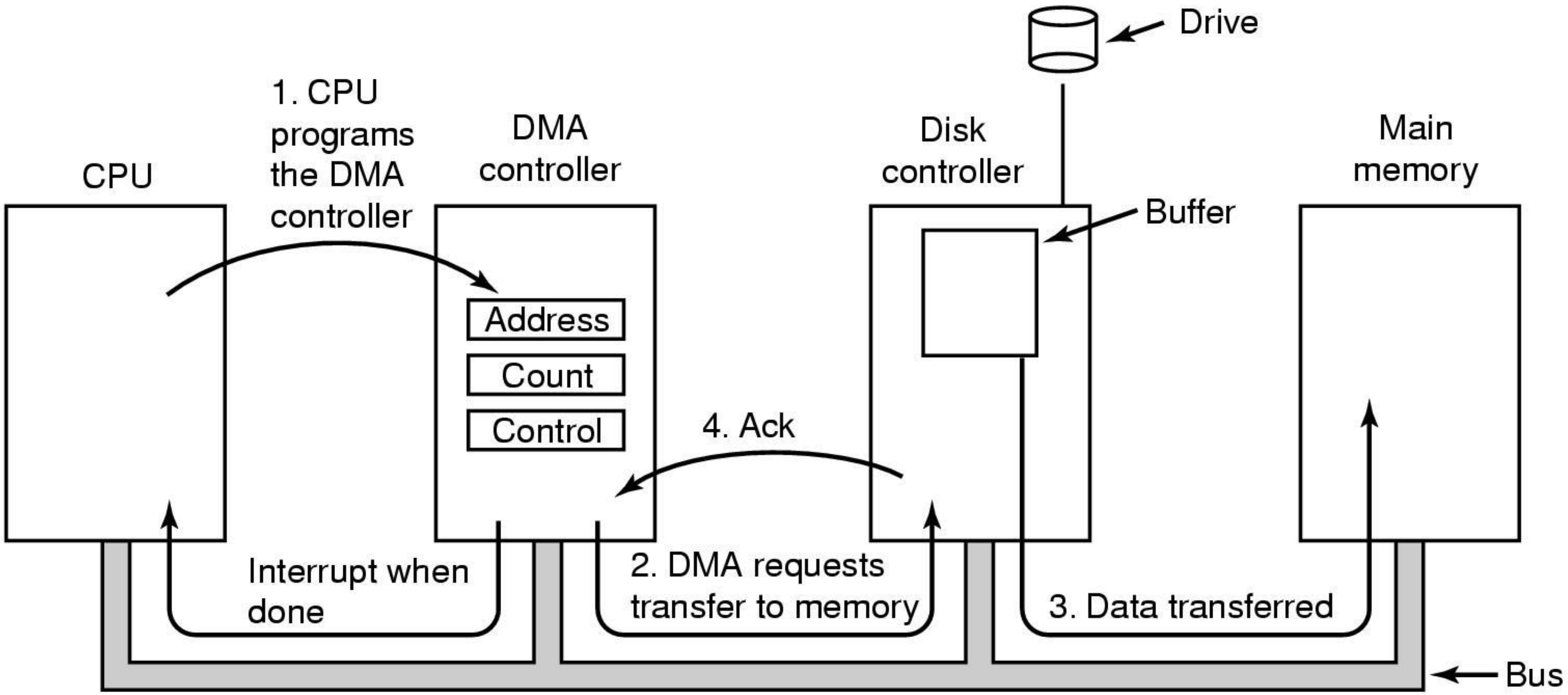
Interrupt



How interrupts happens?

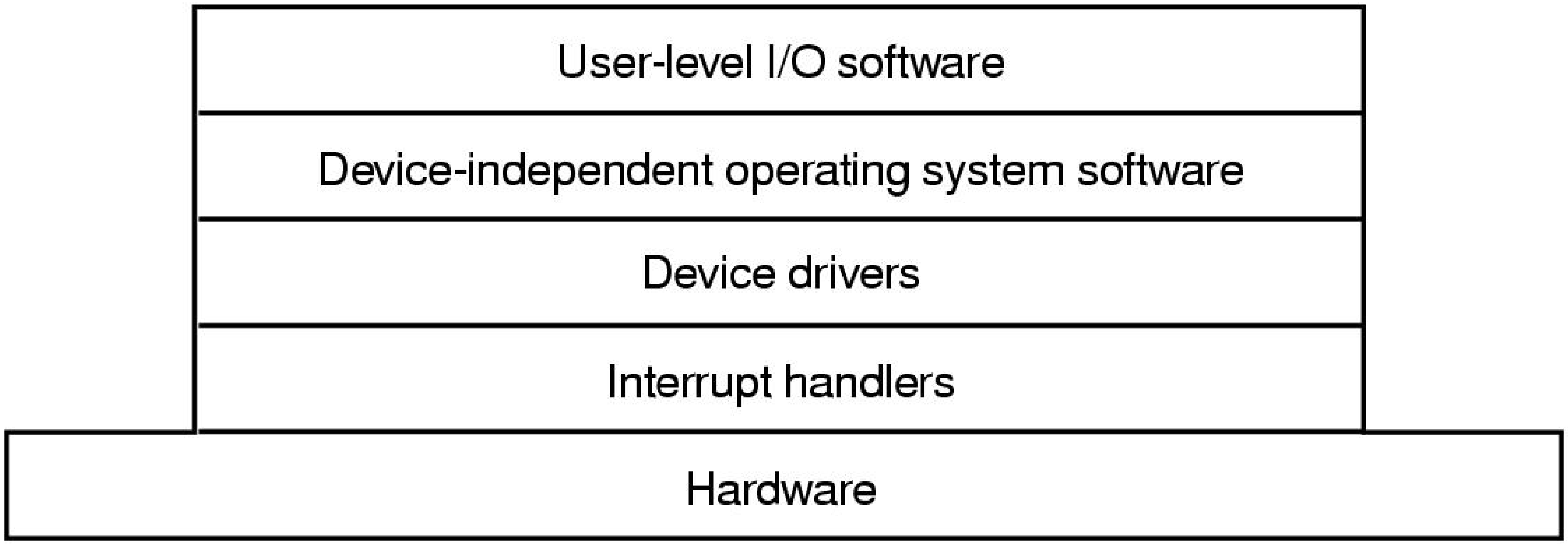
Connections between devices and interrupt controller actually use interrupt lines on the bus rather than dedicated wires

Direct Memory Access (DMA)



Operation of a DMA transfer

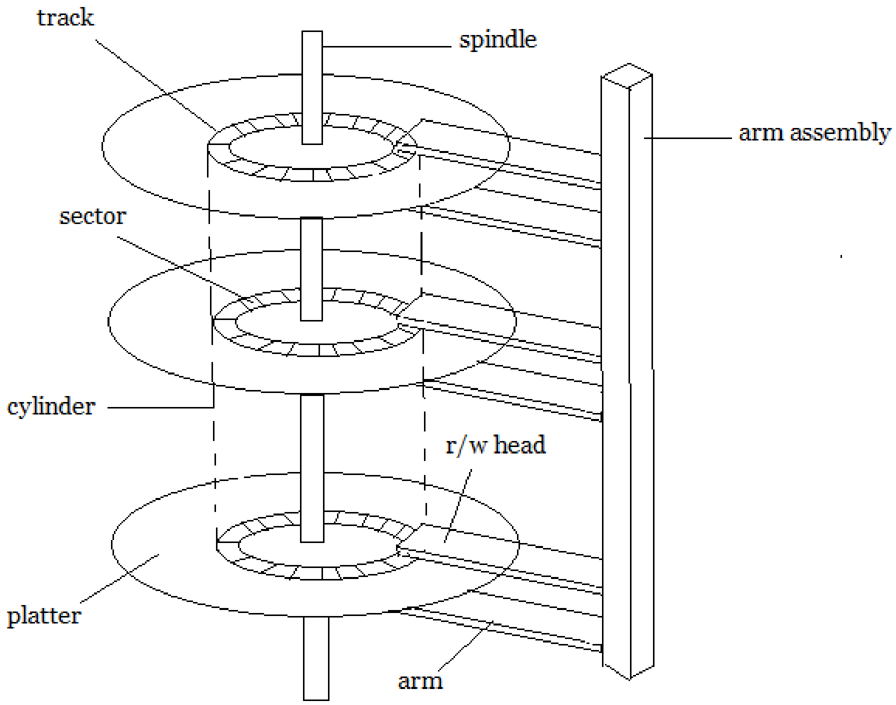
I/O Software Layers



Layers of the I/O Software System

Magnetic Disk

- Hard disks and floppy disks
- Organized into cylinders, tracks, and sectors.



Disk Formatting

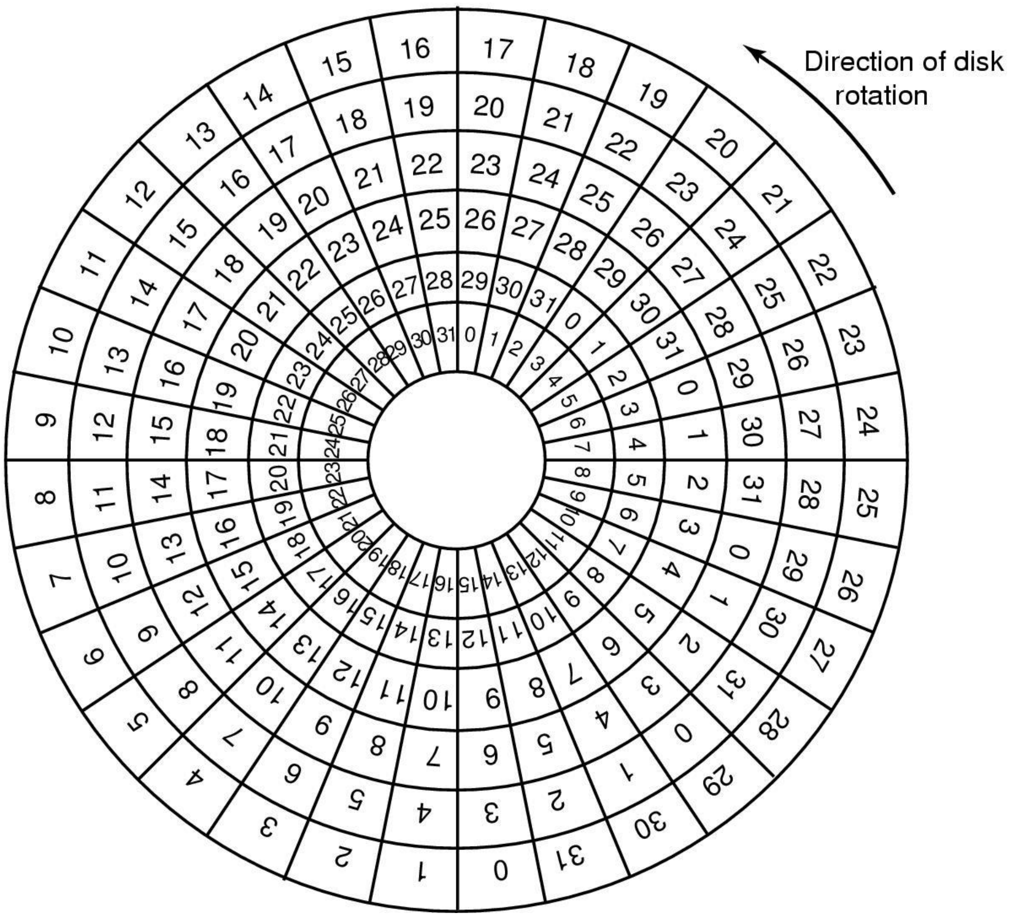
- A low-level format operation should be done on a disk before the disk can be used.
- Each track consists of a number of sectors, with short gaps between the sectors.

512 Bytes **16 Bytes**

A disk **sector**

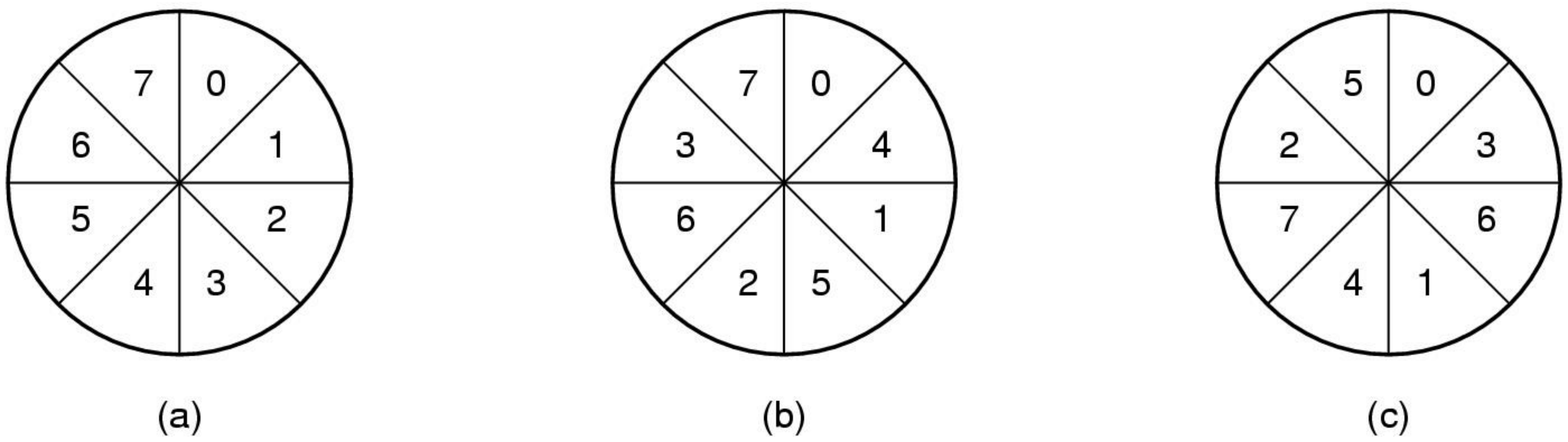
Cylinder Skew

Cylinder skew: the position of sector 0 on each track is offset from the previous track when the low-level format is laid down.



Disk Interleaving

Motivation: when the copy to memory is complete (need some time cost), the controller will have to wait almost an entire rotation time for the second sector to come around again.



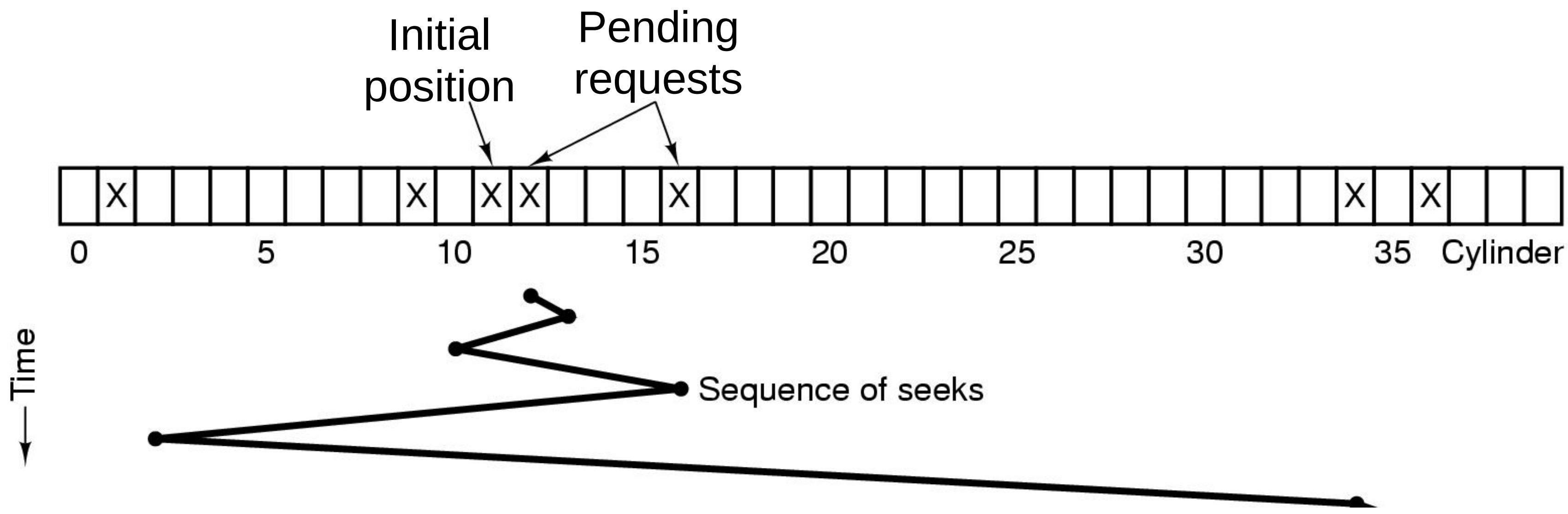
(a) No interleaving; (b) Single interleaving; (c) Double interleaving.

Disk Arm Scheduling Algorithms

- Time required to read or write a disk block determined by 3 factors
  - ✓ Seek time
  - ✓ Rotational delay
  - ✓ Actual transfer time
- Seek time dominates
- Error checking is done by controllers

Disk Arm Scheduling Algorithms: (Shortest Seek First, SSF)

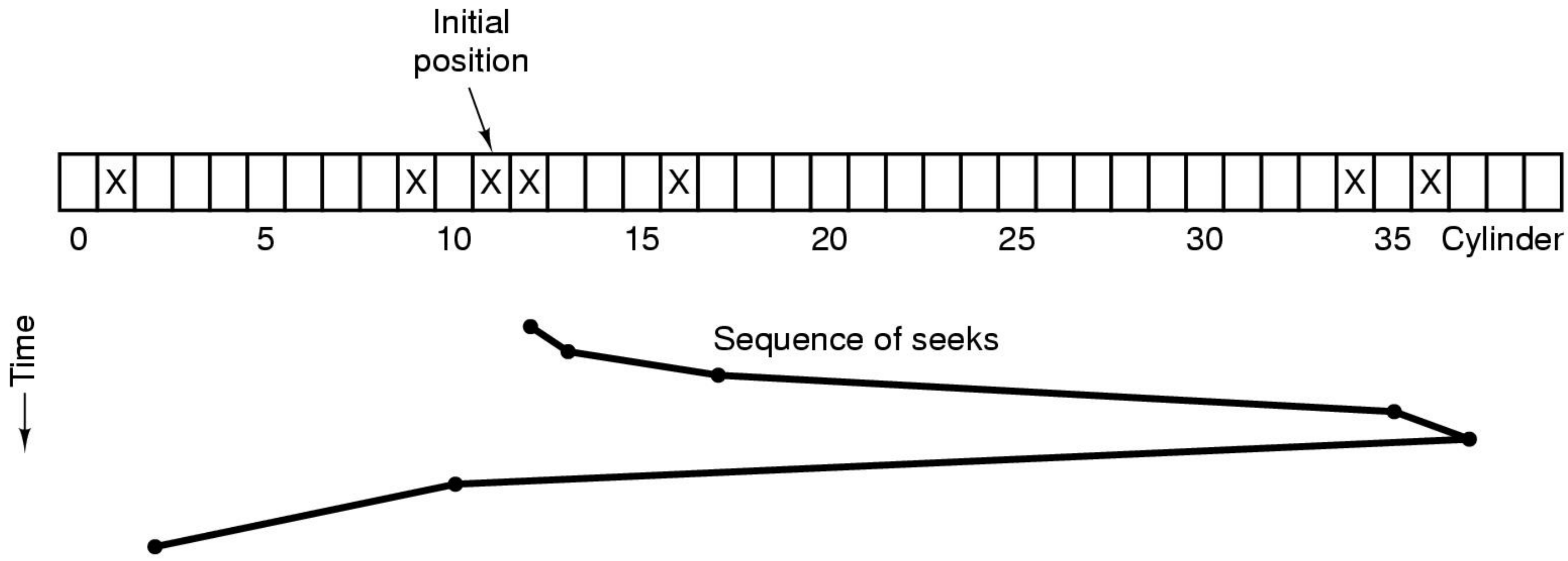
Requests： 11, 1, 36, 16, 34, 9, and 12



Shortest Seek First (SSF) disk scheduling algorithm

Disk Arm Scheduling Algorithms: (Elevator Algorithm)

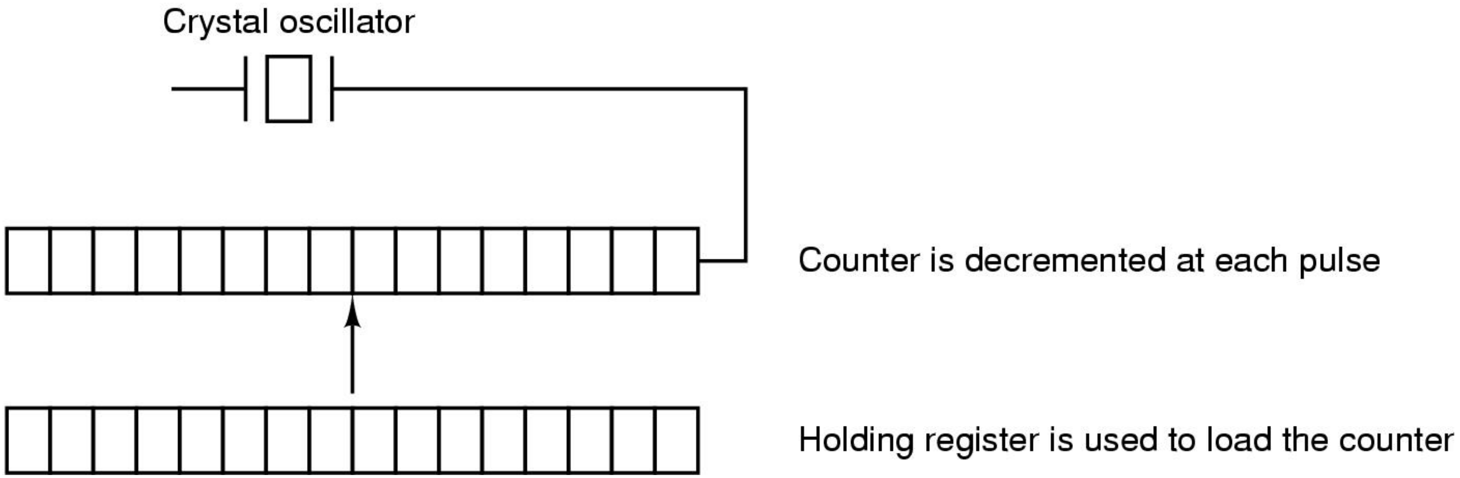
Requests： 11, 1, 36, 16, 34, 9, and 12



The elevator algorithm for scheduling disk requests

Clock Hardware

- The crystal oscillator can generate a periodic signal in the range of several hundred MHz.
- Two modes: one-shot mode, and square-wave mode.
- Clock ticks: periodic interrupts caused by the programmable clock.

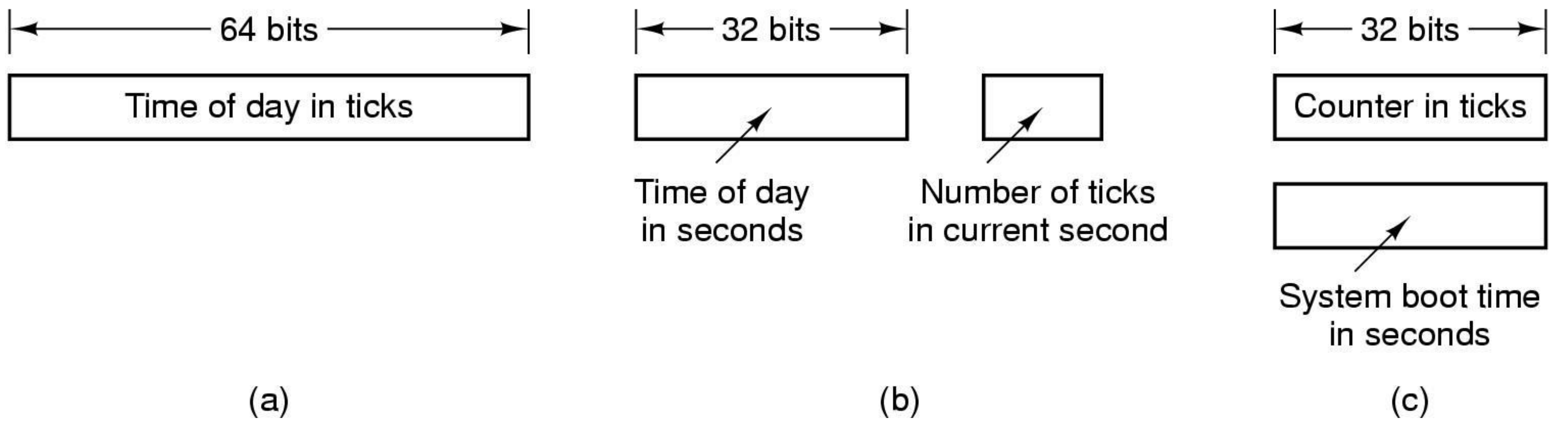


A programmable clock.

Clock Software

- The functions of clock driver

✓ Maintaining the time of day.
✓ Preventing processes from running too long.
✓ Handling the alarm system calls (e.g., ACK).
✓ Others.



Three ways to maintain the time of day.

Power Management (1)

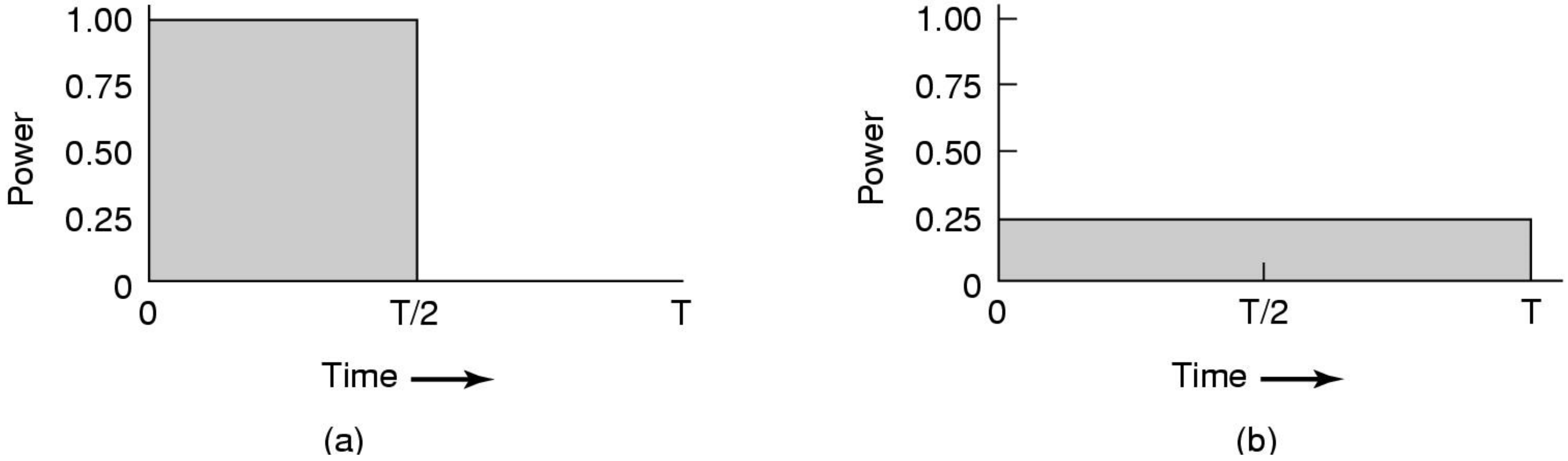
Device	Li et al. (1994)	Lorch and Smith (1998)
Display	68%	39%
CPU	12%	18%
Hard disk	20%	12%
Modem		6%
Sound		2%
Memory	0.5%	1%
Other		22%

Power consumption of various parts of a notebook computer.

The most common method to save battery is to design the devices to have multiple states:

On, Sleep, and Off.

Power Management (2)



(a) Running at full clock speed.

(b) Cutting voltage by two cuts clock speed by two and power consumption by four.

Power Management (3)

- The user can run longer on a given battery by accepting some quality degradation.
  - ✓Abandon the color information and display the video in black and white.
  - ✓Use radio link to send task to other devices.
  - ✓Trading image quality to reduce the transmission overload.

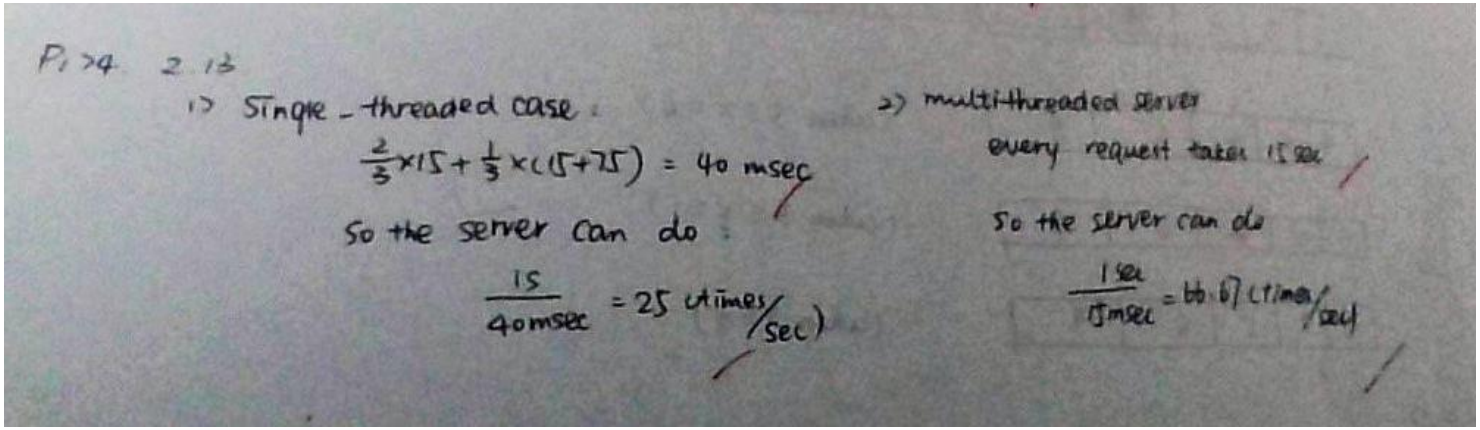
Final Example

Dec. 16, 2019, 8:50am-10:50am

Address: to be announced.

Answer Skills

- Organize your answers and make the important points clear.



P1>4 2.13

1> Single-threaded case:

$\frac{2}{3}\times 15 + \frac{1}{3}\times(15+75) = 40$ msec

So the server can do

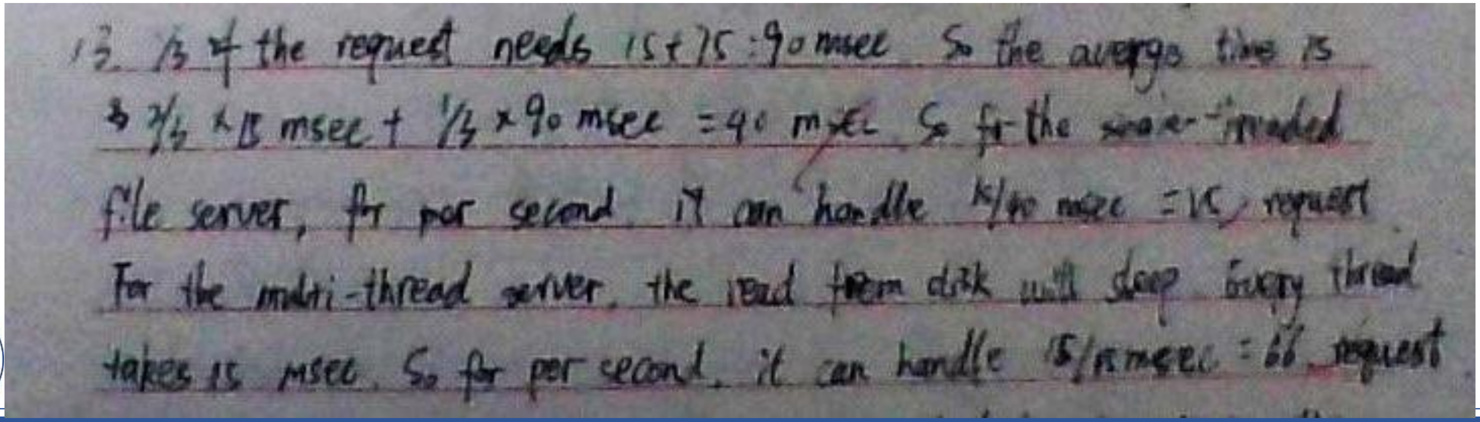
$\frac{1s}{40msec} = 25$ (times/sec)

2) multithreaded server

every request takes 15 sec

So the server can do

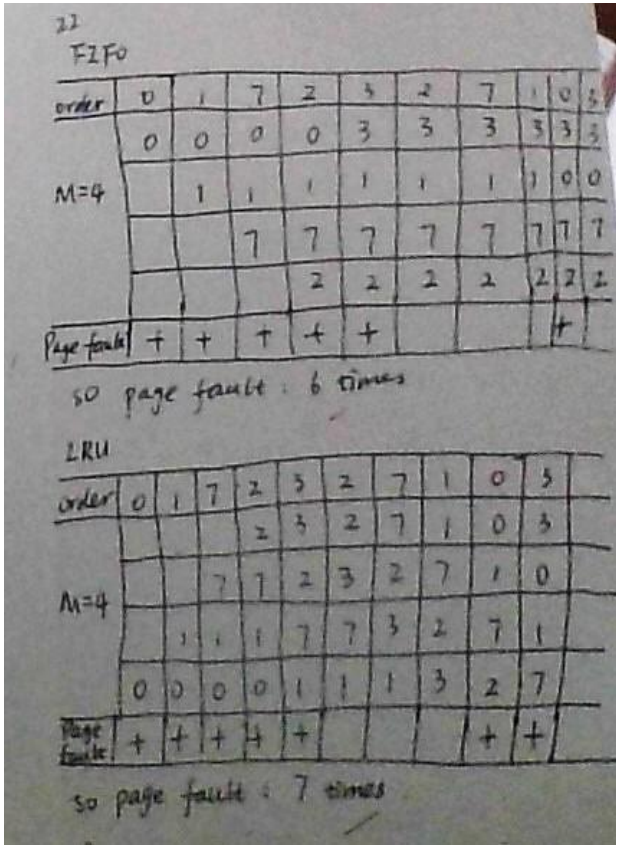
$\frac{1 sec}{15msec} = 66.67$ (times/sec)



13. 1/3 of the request needs 15+75=90 msec. So the average time is 2/3 × 15 msec + 1/3 × 90 msec = 40 msec. So for the single-threaded file server, for per second it can handle 1s/40 msec = 25 request.
For the multi-thread server, the read from disk will sleep. Every thread takes 15 msec. So for per second, it can handle 1s/15msec = 66 request

Answer Skills

- Explain your answers.



22

FIFO

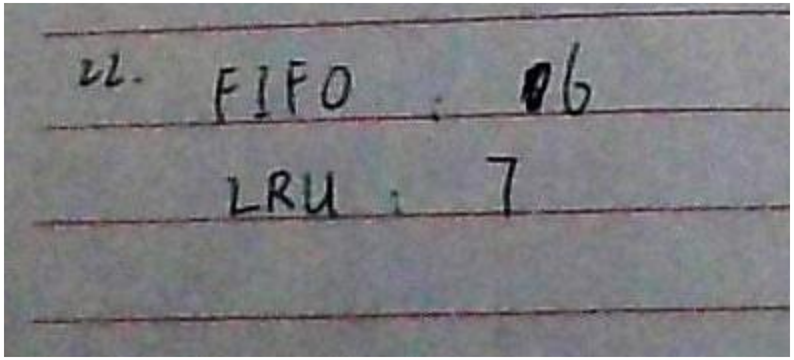
order	0	1	7	2	3	2	7	1	0	3
	0	0	0	0	3	3	3	3	3	3
M=4		1	1	1	1	1	1	1	0	0
			7	7	7	7	7	7	7	7
				2	2	2	2	2	2	2
Page fault	+	+	+	+	+				+	

so page fault : 6 times

LRU

order	0	1	7	2	3	2	7	1	0	3
				2	3	2	7	1	0	3
M=4			7	7	2	3	2	7	1	0
		1	1	1	7	7	3	2	7	1
	0	0	0	0	1	1	1	3	2	7
Page fault	+	+	+	+	+				+	+

so page fault : 7 times



22. FIFO : 6

LRU : 7

Problem Types

- ✓Type1: Explain the meaning of items (each item 4 points),

e.g., Thread, Page Table

- ✓Type2: Questions, e.g.,

e.g., Each process has three states. Draw a diagram to show the transitions between these three states. If there are multiple processes in the ready state, the scheduler will use a scheduling strategy to select one process to run the CPU. Describe the basic ideas of the following process scheduling strategies: Round robin, Priority scheduling, and shortest job first. (15 points)