CS 444 Operating Systems

Chapter 5 Input and Output

J. Holly DeBlois

September 30, 2024

Device Characterization

- Block devices
  - Stores information in fixed-size blocks
  - Transfers are in units of entire blocks
- Character devices
  - Delivers or accepts stream of characters, without regard to block structure
  - Not addressable, does not have any seek operation
- Gray area
  - Tapes, clock

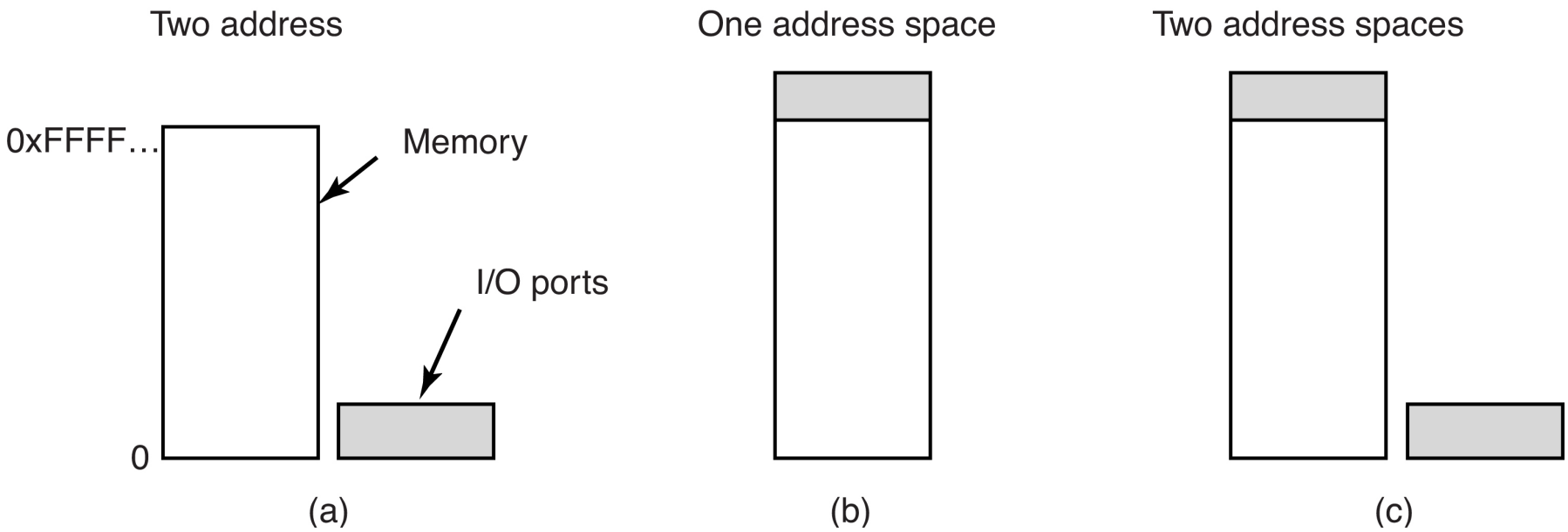
Data Rates of Typical Devices

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner at 300 dpi	1 MB/sec
Digital camcorder	3.5 MB/sec
4x Blu-ray disc	18 MB/sec
802.11n Wireless	37.5 MB/sec
USB 2.0	60 MB/sec
FireWire 800	100 MB/sec
Gigabit Ethernet	125 MB/sec
SATA 3 disk drive	600 MB/sec
USB 3.0	625 MB/sec
SCSI Ultra 5 bus	640 MB/sec
Single-lane PCIe 3.0 bus	985 MB/sec
Thunderbolt 2 bus	2.5 GB/sec
SONET OC-768 network	5 GB/sec

Device Components

- The mechanical component
- The electronic component
  - Device controller
- Example: a 1TB disk
  - 2M sectors of 512 bytes per track
  - The bits in a sector
    - preamble: head number, cylinder number, sector number, sector size, synchronization info
    - 4,096 bits of data
    - error correction code (ECC)

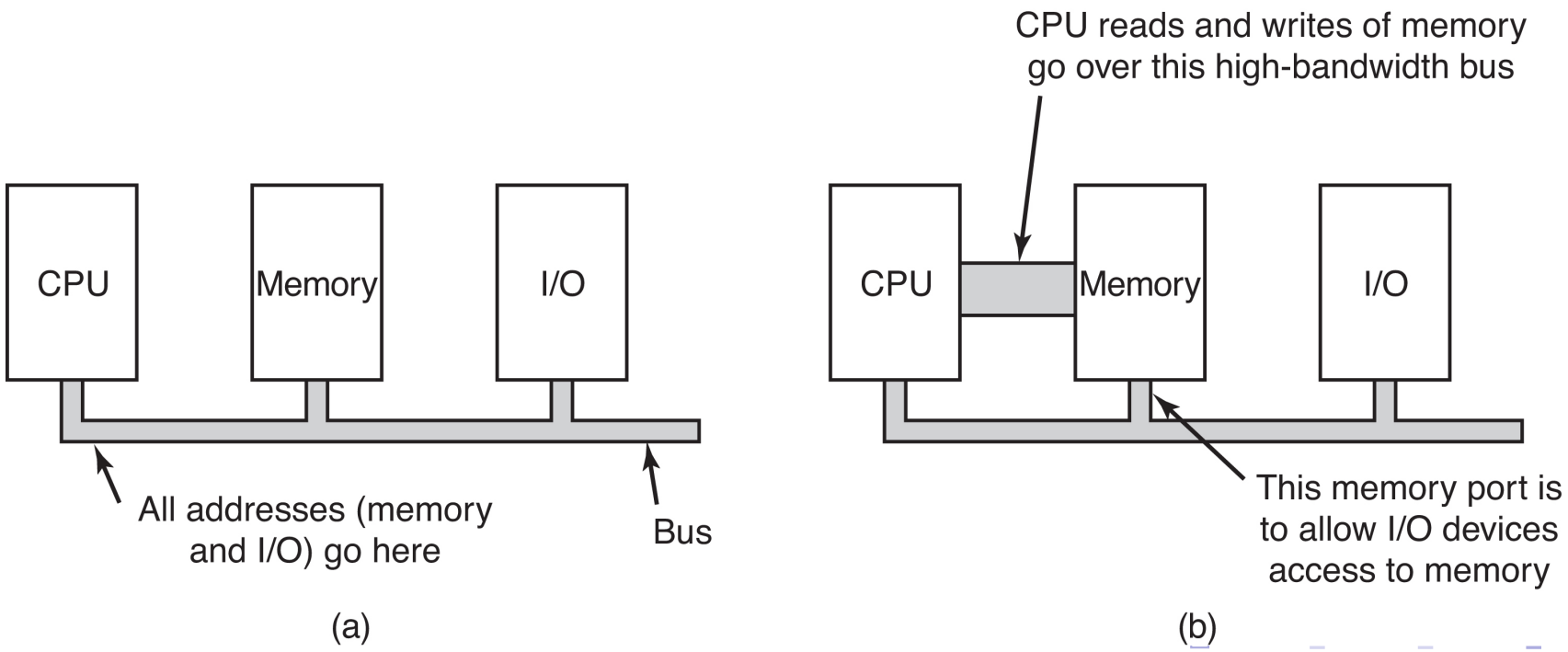
Approaches of Device Interface



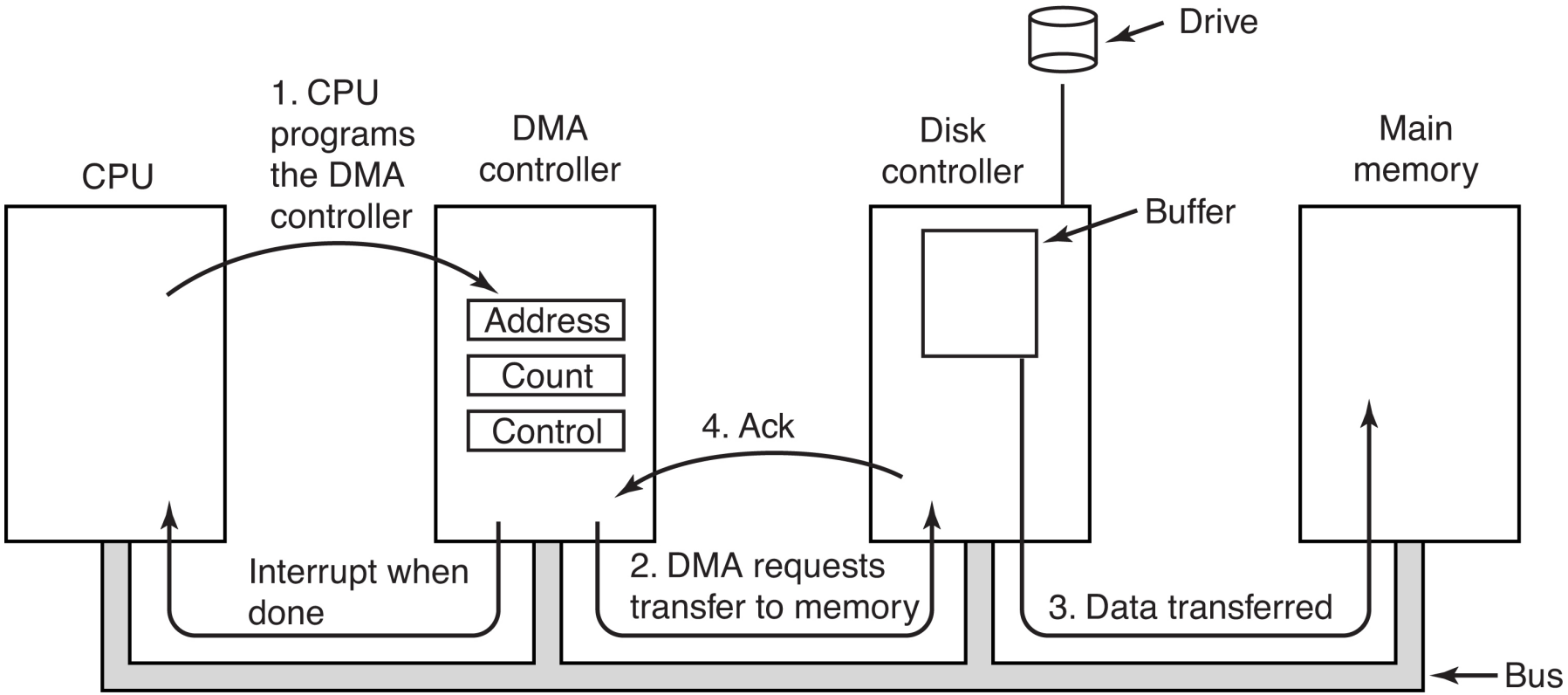
- Separate I/O and memory space
- Mainframes, IBM 360
- I/O ports, special assembly instructions
  - IN REG, PORT
  - OUT PORT, REG
- Memory-mapped I/O
- Hybrid

Memory-Mapped I/O

- Advantages
  - Device registers are variables in C
  - Device drivers can be written entirely in C
  - OS grants access
  - CPU assembly instructions can work on device registers directly
- Disadvantages
  - Mapped memory can't be cached
  - One address space, multiple buses



Direct Memory Access

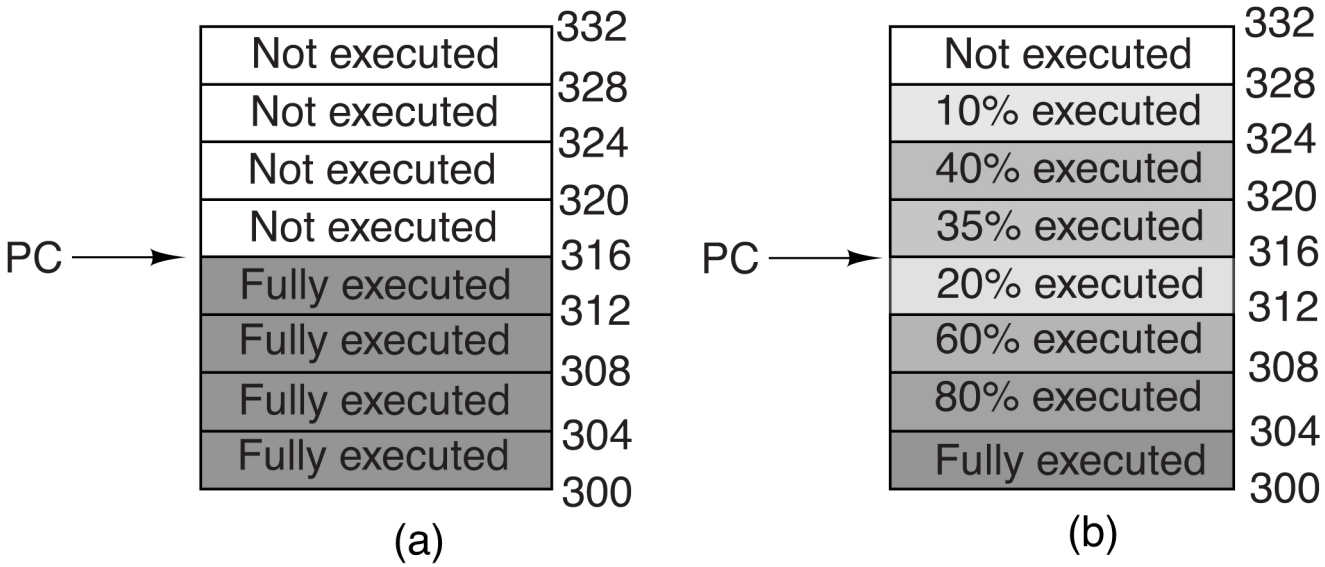


- DMA controller is located on the motherboard
- Part of the southbridge — slower, older technology
- Northbridge: CPU, RAM, GPU

DMA Operating Modes

- Two operating modes
- Fly-by mode
  - Device controller transfers data to RAM directly
  - Cycle stealing: one word at a time
  - Burst mode: block mode
- Device controller sends data to DMA
  - DMA writes to destination
  - More flexible

Precise vs. Imprecise Interrupts



- x-86 uses precise interrupt
- Four properties of a precise interrupt:

1. The PC saved in a known place
2. All instructions before that pointed to by PC have fully executed
3. No instruction beyond that pointed to by PC has been executed
4. Execution state of instruction pointed to by PC is known

I/O Software

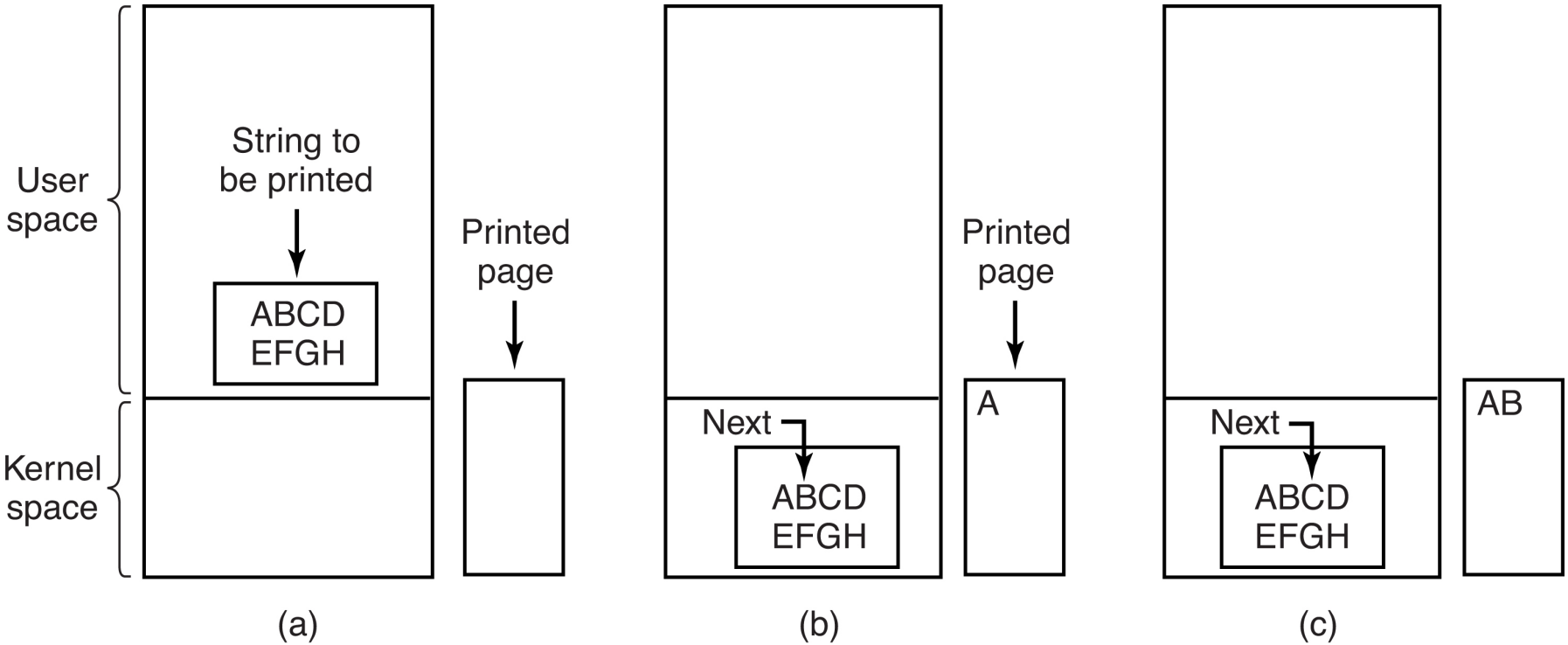
- Goals of I/O software
- Device independence
  - Uniform naming
- Error handling
- Synchronous vs. asynchronous
  - Blocking vs. interrupt-driven
- Buffering

Three Approaches of I/O Software

- Programmed I/O
- Interrupt-driven I/O
- DMA I/O

Programmed I/O

- CPU does all the work
- Steps in printing a string:



Code for Programmed I/O

```
copy_from_user(buffer, p, count);                /* p is the kernel buffer */
for (i = 0; i < count; i++) {                    /* loop on every character */
        while (*printer_status_reg != READY) ;   /* loop until ready */
        *printer_data_register = p[i];           /* output one character */
}
return_to_user( );
```

- Writing a string to the printer using programmed I/O
- Running in kernel mode
- Busy waiting — polling

Interrupt-Driven I/O

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

- Code executed at the time the print system call is made

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count – 1;
     i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

- Interrupt service procedure for the printer

DMA I/O

```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

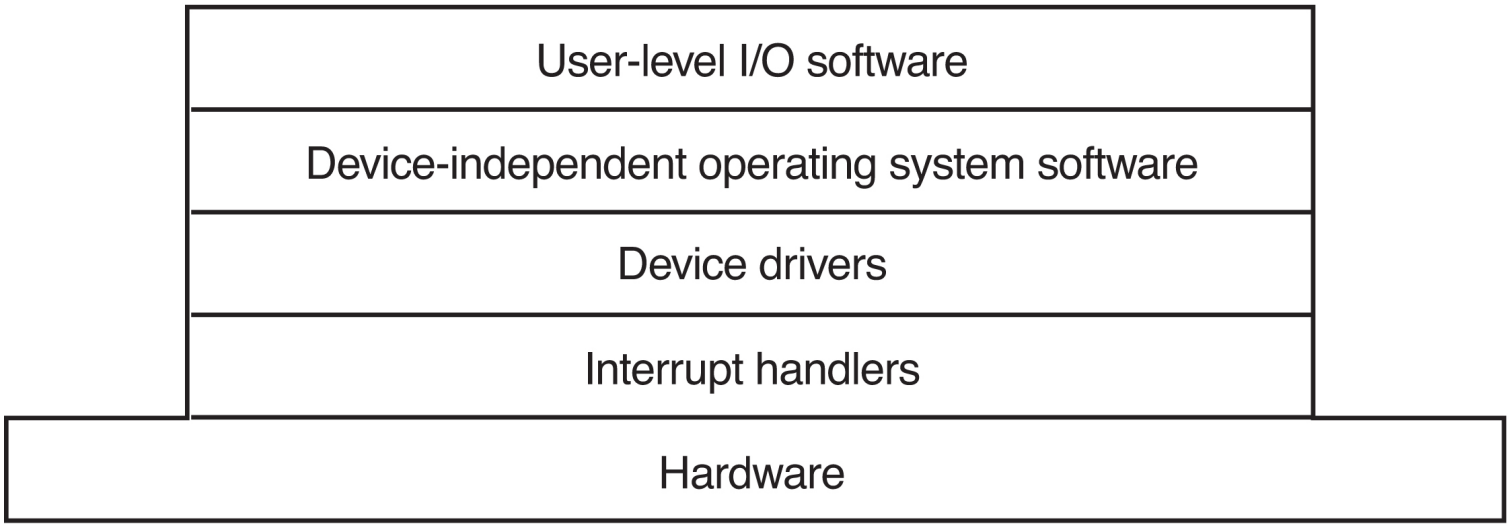
- Code executed when the print system call is made

```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

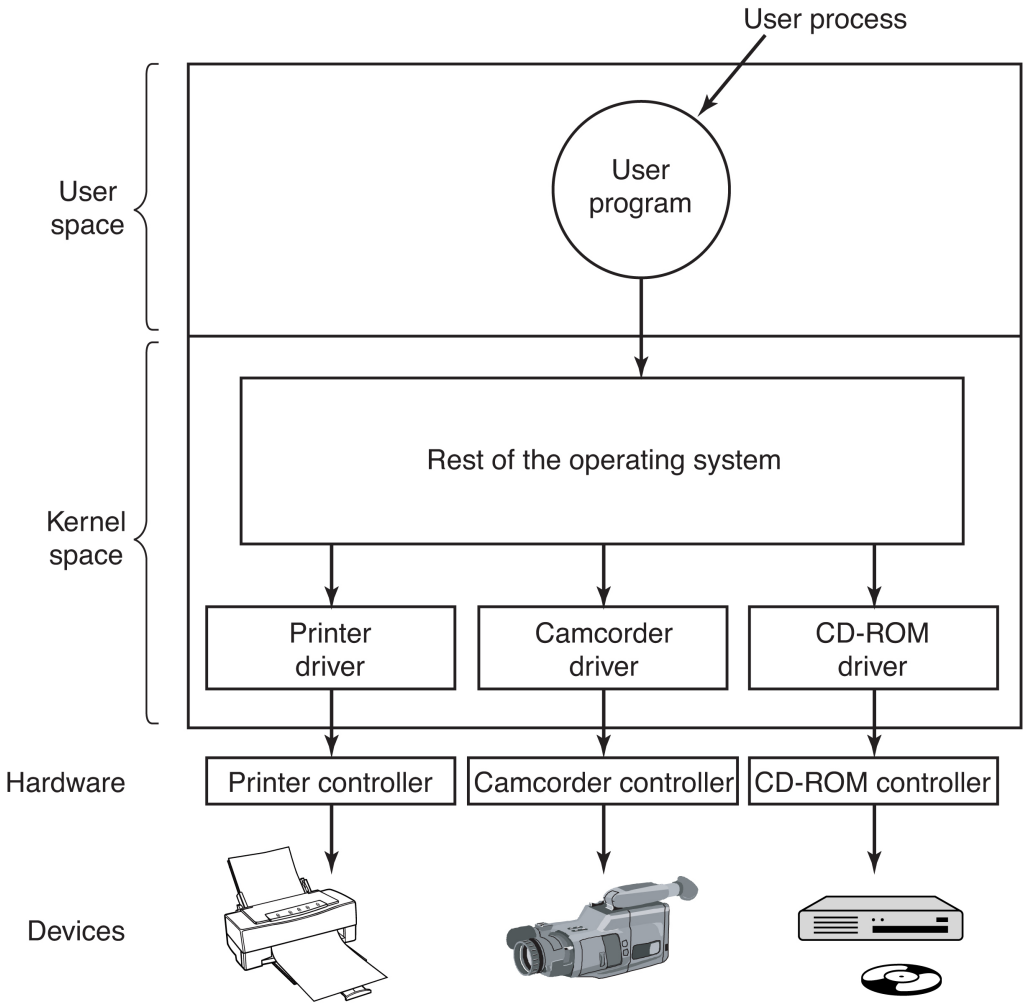
(b)

- Interrupt service procedure

I/O Software Layers



Device Drivers



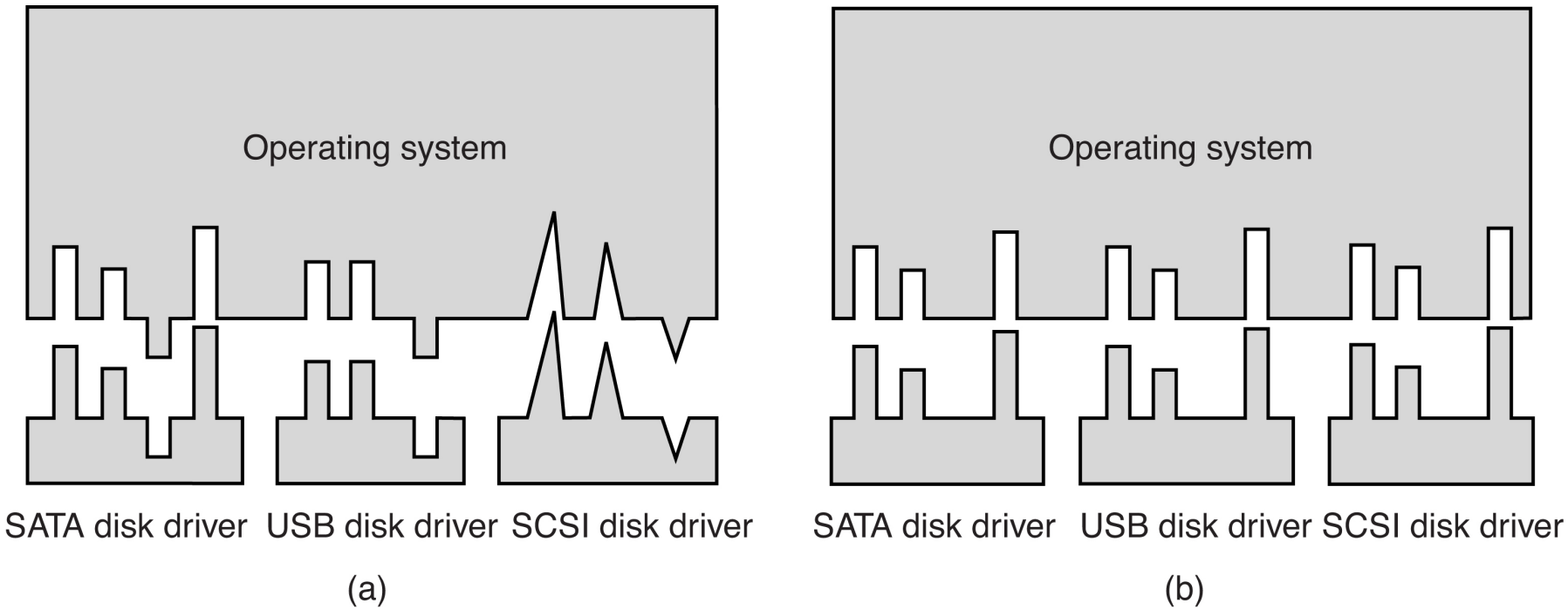
- Logical positioning of device drivers
- All communication between drivers and device controllers goes over the bus
- Tasks:
  - Initialize the device
  - Log events
  - Check input parameters
  - Translate abstract to concrete
  - Check for error
  - Return status to OS
  - Must be reentrant

Device-Independent I/O Software

- Functions of device-independent I/O software
- Uniform interfacing for device drivers
- Buffering
- Error reporting
- Allocating and releasing dedicated devices
- Providing a device-independent block size

Uniform Interfacing for Device Drivers

- The OS defines a set of functions that all drivers must supply

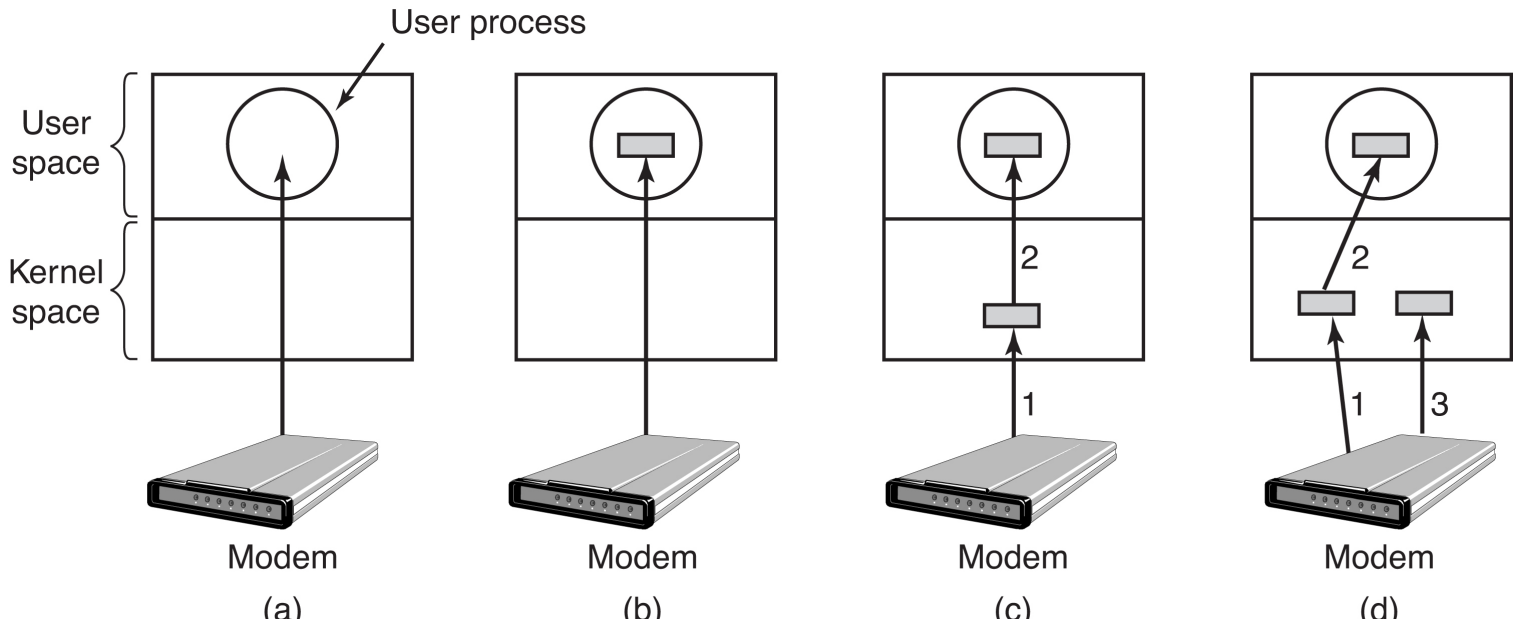


- Without a standard driver interface
- With a standard driver interface

Uniform Naming

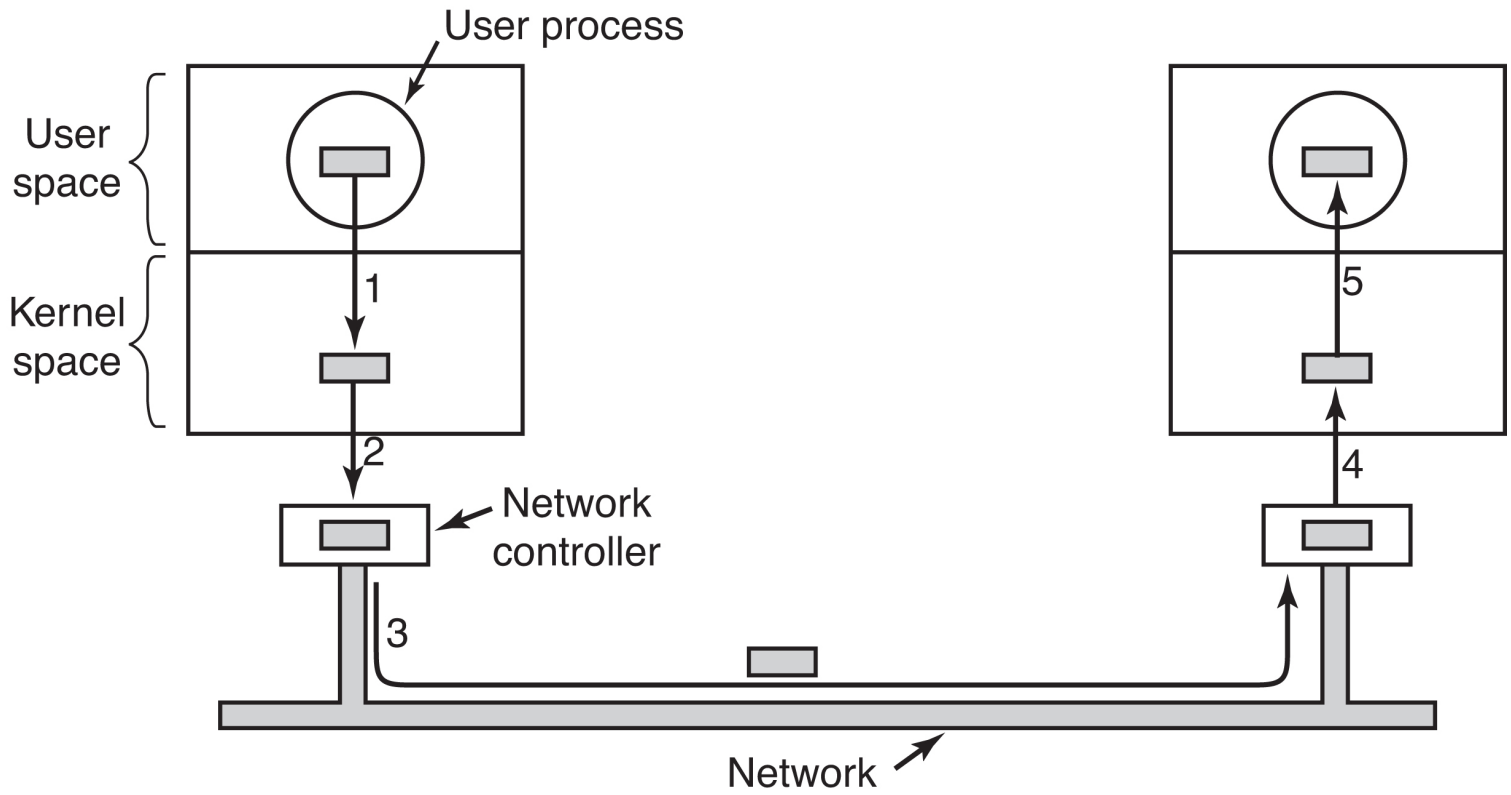
- Unix device name, for example, `/dev/sda`
- It specifies the i-node for a special file
- This i-node has the major and minor device numbers
- Major device number: which driver
- Minor device number: which unit
- Protection is specified by `rwx` for user, group, and other

Buffering



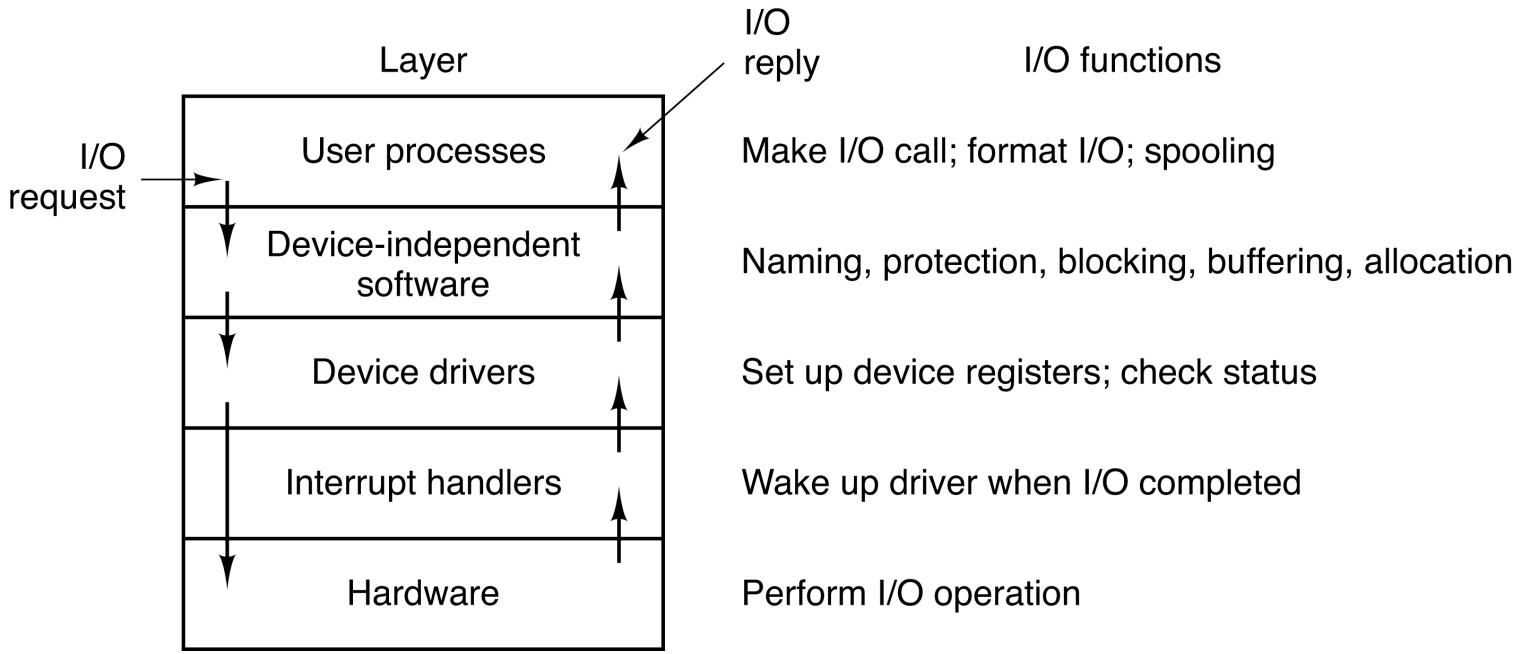
- Unbuffered input
- Buffering in user space
- Buffering in the kernel followed by copying to user space
- Double buffering in the kernel
  - Circular buffering

Buffering May Lead to Poor Performance



- Networking may involve making many copies of a packet

I/O Software in User Space



- Library function `printf()`
- System call `write()`
- Many layers in user space I/O software

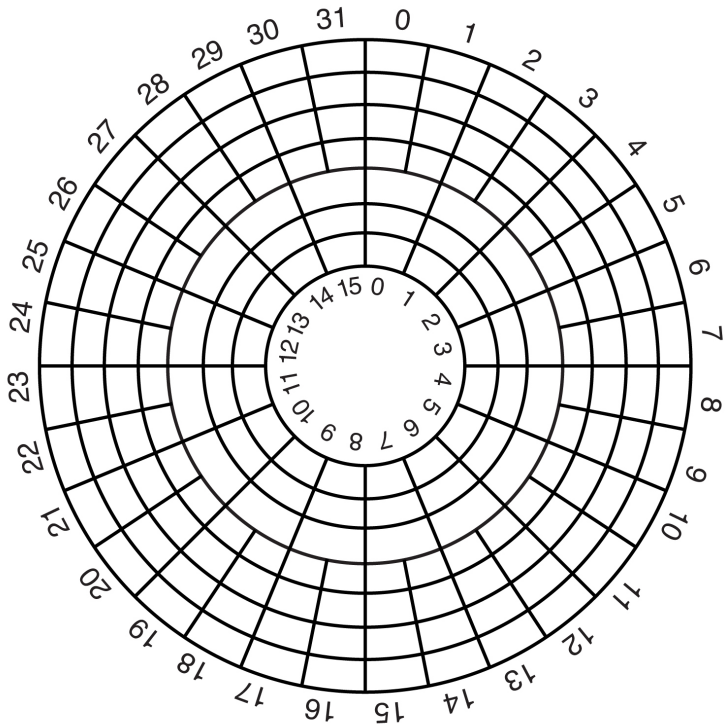
Magnetic Disks 3 Decades Apart

Parameter	IBM 360-KB floppy disk	WD 3000 HLFS hard disk
Number of cylinders	40	36,481
Tracks per cylinder	2	255
Sectors per track	9	63 (avg)
Sectors per disk	720	586,072,368
Bytes per sector	512	512
Disk capacity	360 KB	300 GB
Seek time (adjacent cylinders)	6 msec	0.7 msec
Seek time (average case)	77 msec	4.2 msec
Rotation time	200 msec	6 msec
Time to transfer 1 sector	22 msec	1.4 μsec

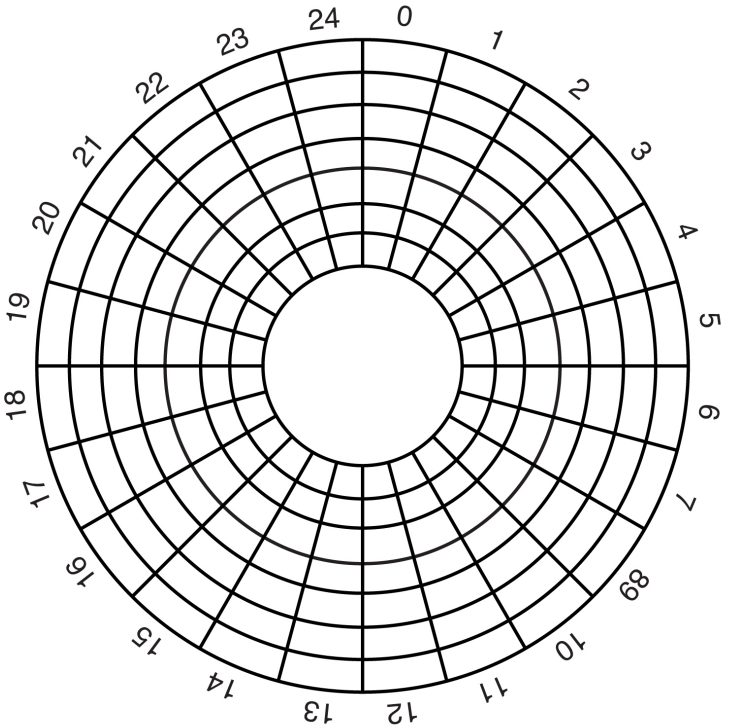
Disk Sector Coordinates

- Cylinder
  - Same track number, on both sides of platters
  - Track 0 at the outer rim
- Head
  - 2 heads per platter
- Sector
  - Track 0 sector 0, MBR
  - MBR partition entries are 32 bits, if 512 B/sector, max 2 TB
  - GUID partition table, GPT, works with larger capacity disks
    - Globally unique identifier, 128 bits
    - Universally unique identifier, UUID
- (x, y, z)

Physical and Virtual Disk Geometry



- The physical geometry with two zones

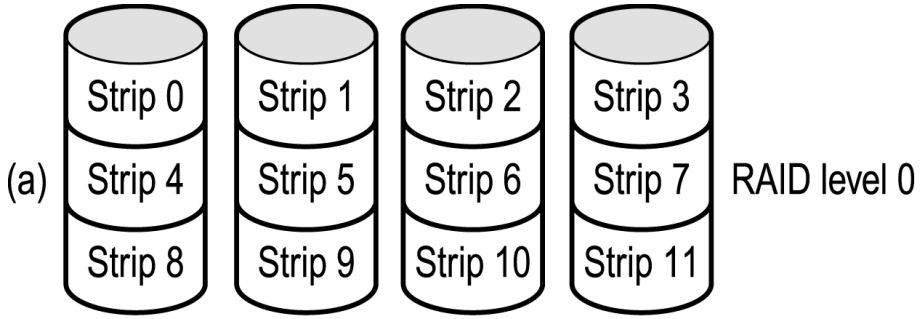


- A virtual geometry presented to the OS

RAID

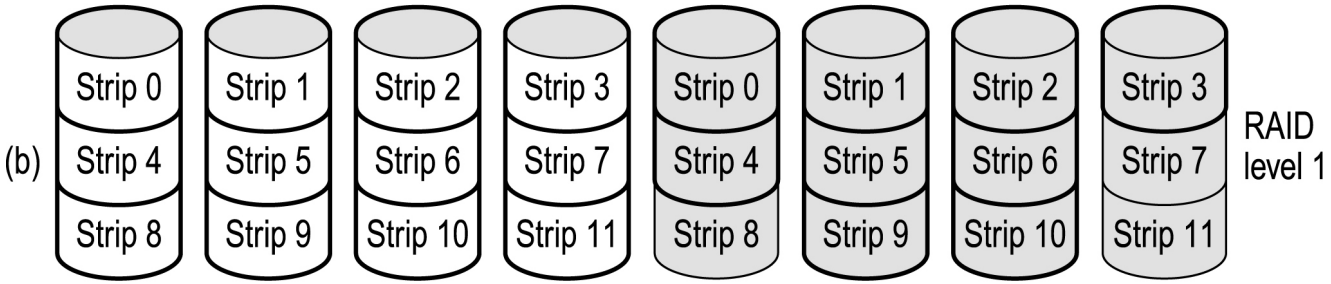
- Redundant array of inexpensive disks
  - To replace a "single large expensive disk"
  - Redundant array of independent disks
- Redundancy to increase reliability
  - Disk failure: mean time to failure (e.g. 20,000 hours) is an exponential distribution
  - Number of disk failure in a unit of time is a Poisson distribution
  - We will study queueing theory later in the semester
- RAID levels: 0 through 6

RAID 0



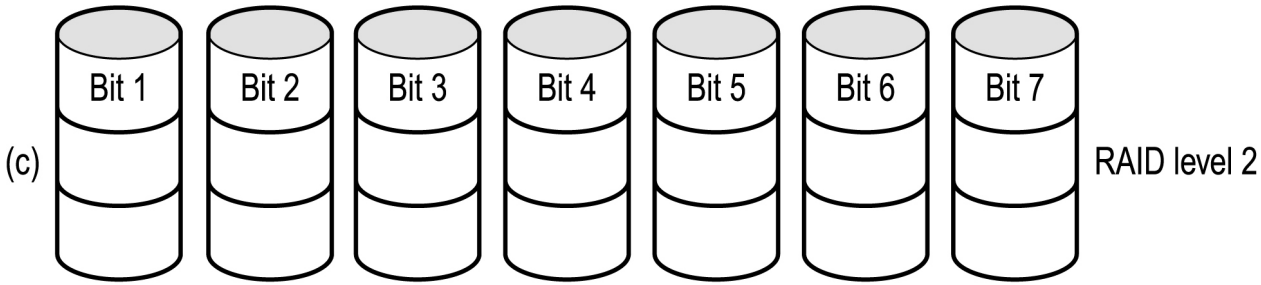
- No redundancy — a misnomer
- A file is split into stripes — striping
  - 1 stripe is 1 disk sector
- The stripes are sequentially distributed to the disks
- Increased I/O throughput for big files
- Decreased reliability
  - Decreased overall mean time to failure
  - Just 1 disk failure leads to data loss

RAID 1



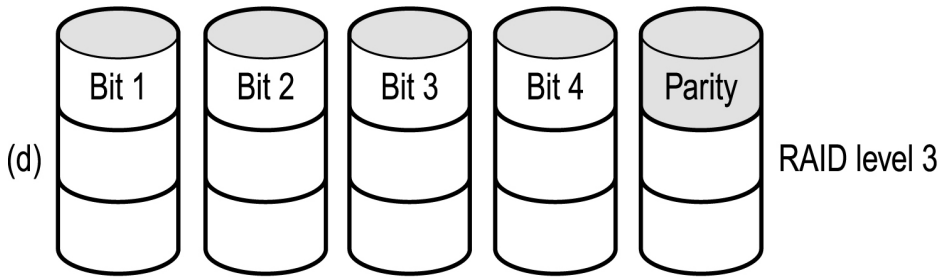
- Mirroring, duplicating
- Increased I/O throughput for big files
- Tolerating 1 disk failure
  - Replace the failed disk
  - Copy from the good disk

RAID 2



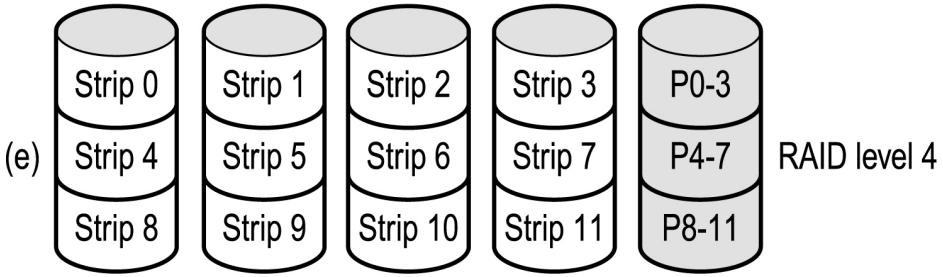
- Hamming(7, 4) code detects and corrects errors
- Split a nibble — 4 bits — to 7 drives
  - Identical drives spin at the same rate
  - 4 times throughput
- Large overhead, $3/7 = 43\%$ capacity is used for parity
  - Hamming(15, 11): $4/15 = 27\%$ overhead
  - Hamming(31, 26): $5/31 = 16\%$ overhead
- Tolerating one disk failure

RAID 3



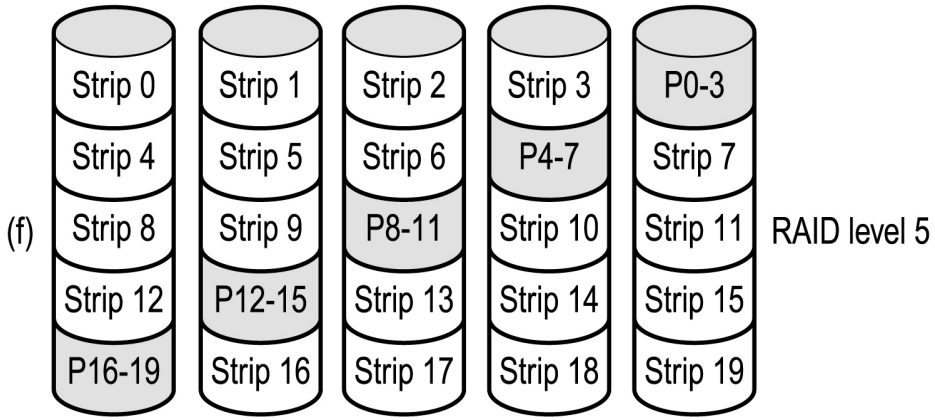
- Simplified version of RAID 2
- A single parity drive
- Hardware tells us which drive is dead
- No need for Hamming code to locate it

RAID 4



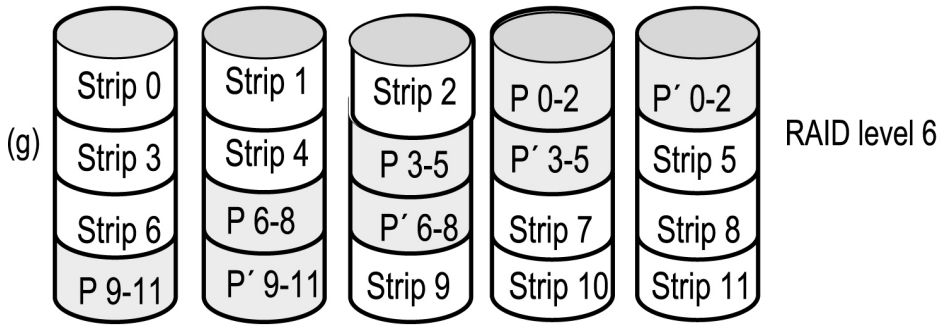
- Improved version of RAID 3
- Using sectors (stripes) rather than bits as the unit of parity calculation
- When 1 sector (stripe) is updated, all other data disks must be read, and the parity stripe is updated
- Alternatively, compute the parity of old data stripe, new data stripe, and old parity stripe, then rewrite the parity stripe
  - 2 reads and 2 writes per update
- The parity drive is heavily used and likely to fail first

RAID 5



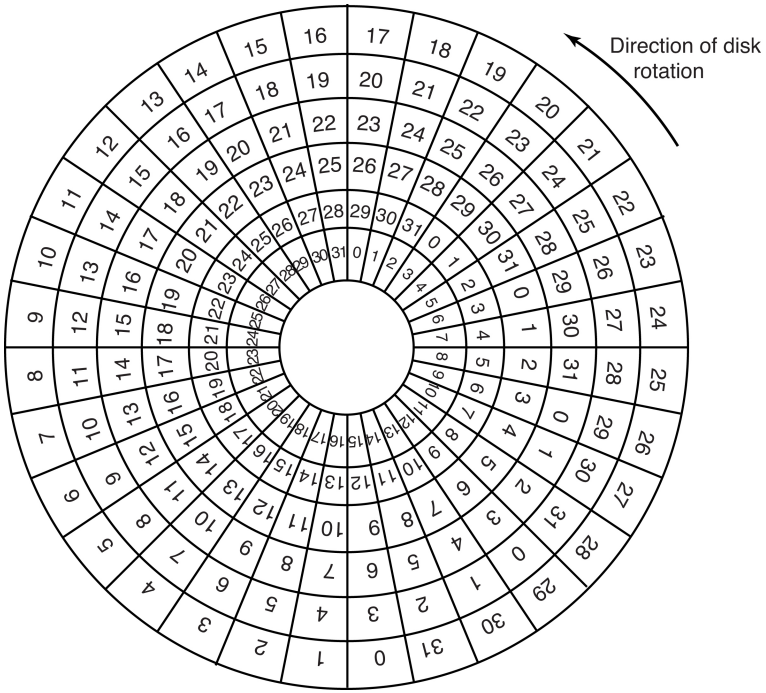
- Improved version of RAID 4
- Distribute the parity stripes over all drives
- When 1 drive fails, reconstruction is complicated

RAID 6

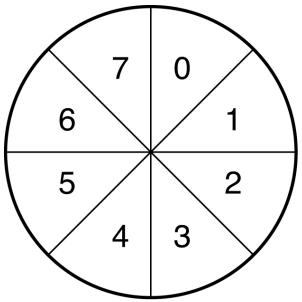


- Similar to RAID 5
- 2 parity drives
- Better fault tolerant

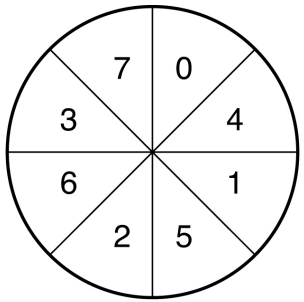
Disk Low-Level Format



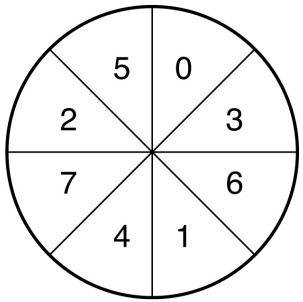
- Each sector has a preamble, 512B data, and ECC
- Cylinder skew
- Single interleaving
- Double interleaving
- Use a big buffer to avoid interleaving



(a)



(b)



(c)

Disk High-Level Format

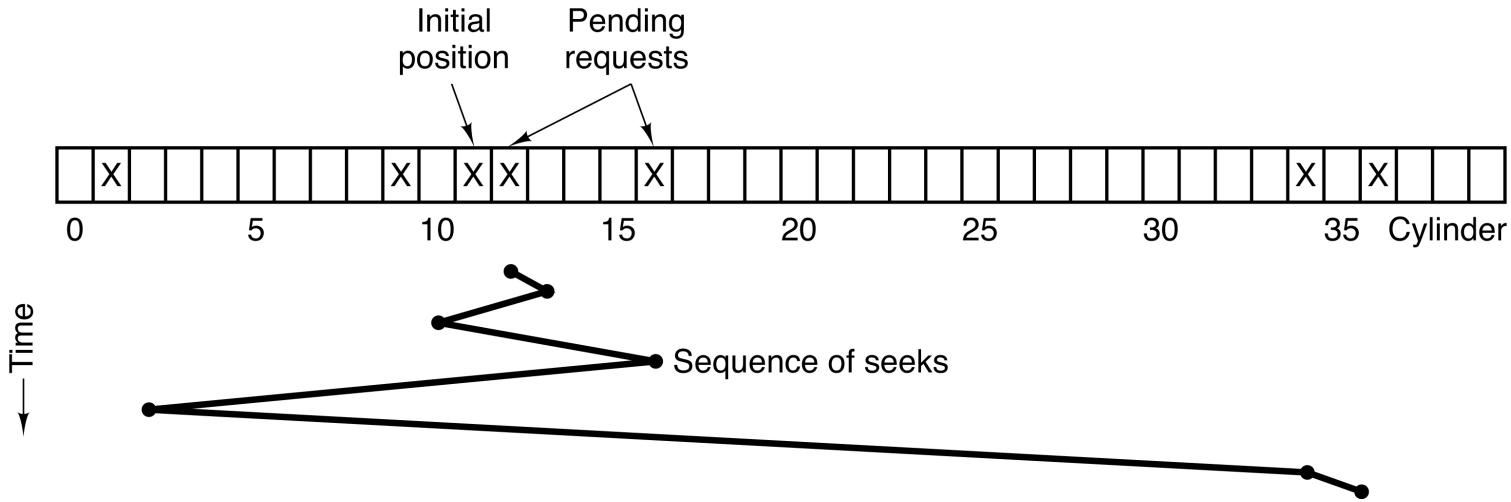
- Boot block
- Free storage admin — linked list or bitmap
- Root directory
- An empty file system
- An entry in the partition table identifies which FS is used

Disk I/O Time

- Seek time: the time to move the arm to the proper cylinder
- Rotational delay: how long for the proper sector to come under the head
- Data transfer time
- The first two dominate the total time

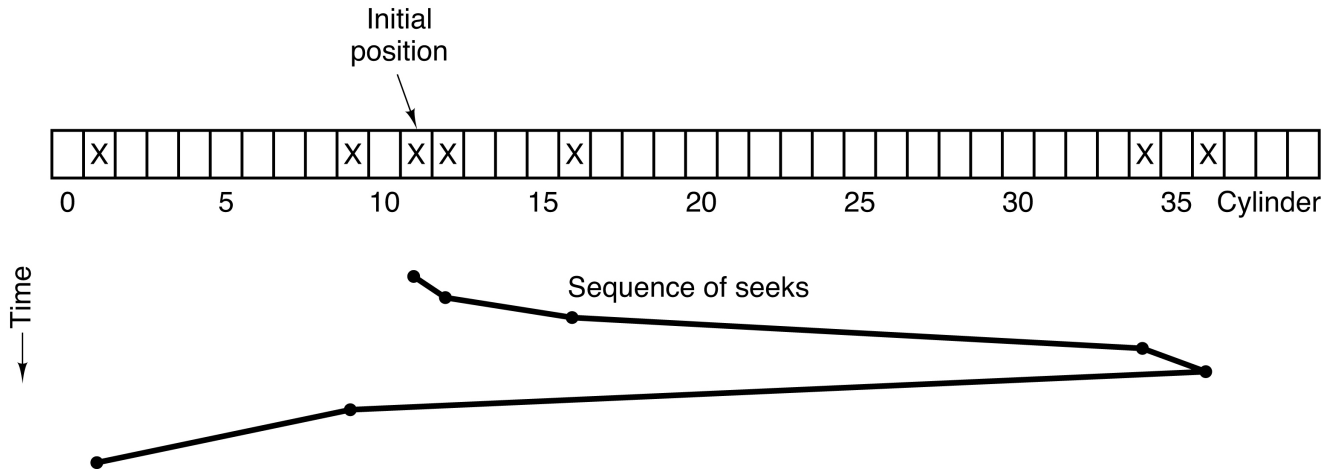
Disk Arm Scheduling Algorithms

- FCFS, no scheduling is needed
- Shortest seek first



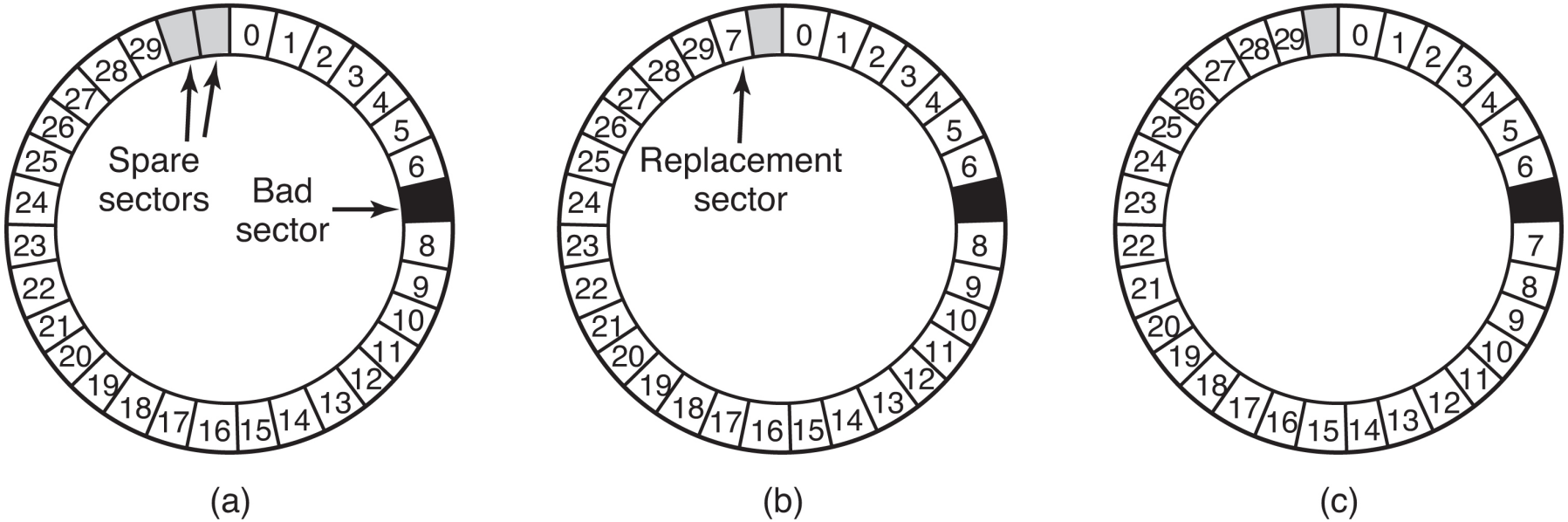
- An issue with SSF: under heavy load, the disk arm tends to stay in the middle
- Data requests for outer and inner cylinders will wait for a long time

The Elevator Algorithm



- When going up, keep going up to the highest floor requested
- When going down, keep going down to the lowest floor requested
- Variation: Go up all the way while fulfilling requests, then zip all the way down and go up again
- Another variation: when seek time is much faster than rotational delay, steal to the next cylinder and catch a sector then steal back

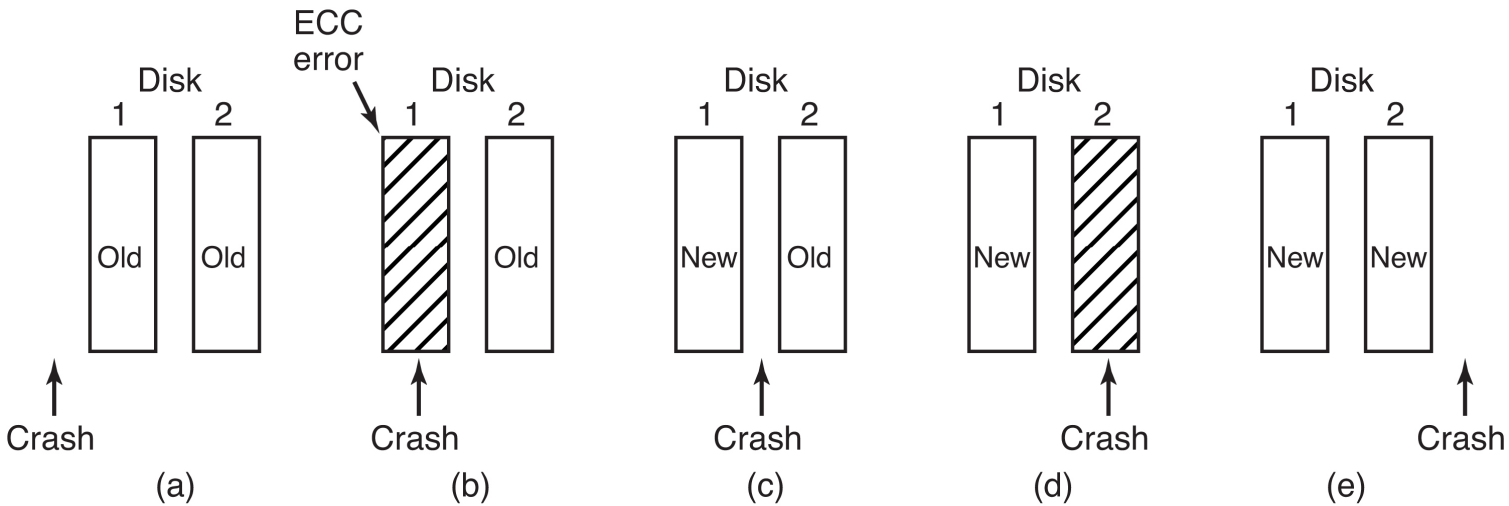
Error Handling



- Substitute a spare sector for a bad one, or
- Shift all sectors to bypass the bad one

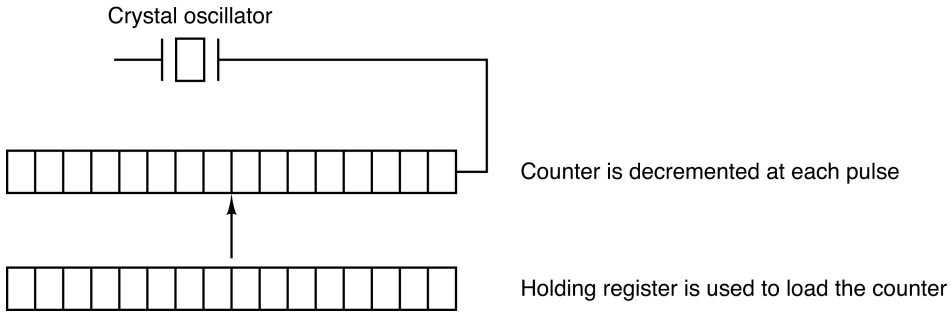
Stable Storage

- Uses pair of identical disks
- Either can be read to get same results
- Operations defined to accomplish this
  - Stable writes
  - Stable reads
  - Crash recovery

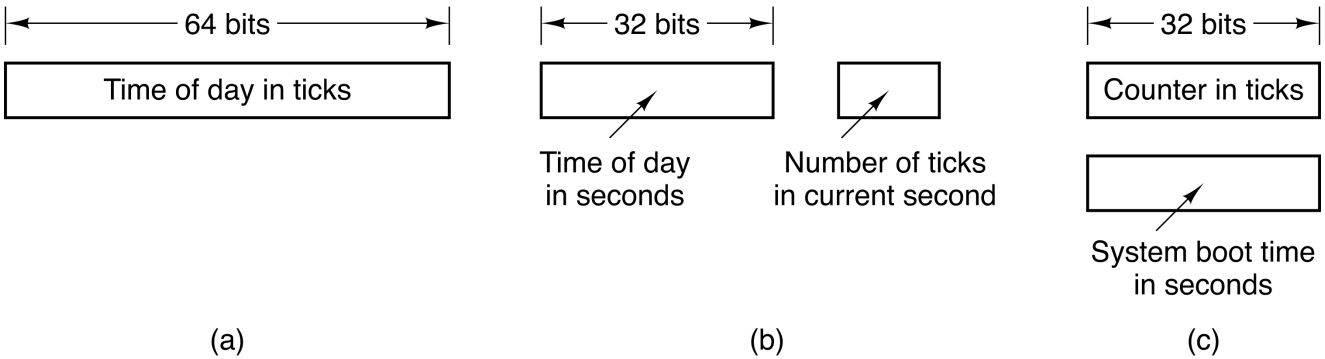


Clock

- Clock hardware



- Clock software



- Year 2038 problem

Clock Driver

- Maintaining the time of day
- Preventing processes from running longer than allowed
- Accounting for CPU usage
- Handling alarm system call from user processes
- Providing watchdog timers for parts of system itself
- Profiling, monitoring, statistics gathering

User Interface

- Keyboard
- Mouse
- Monitor
- X Window
- CS 615