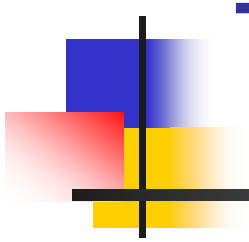
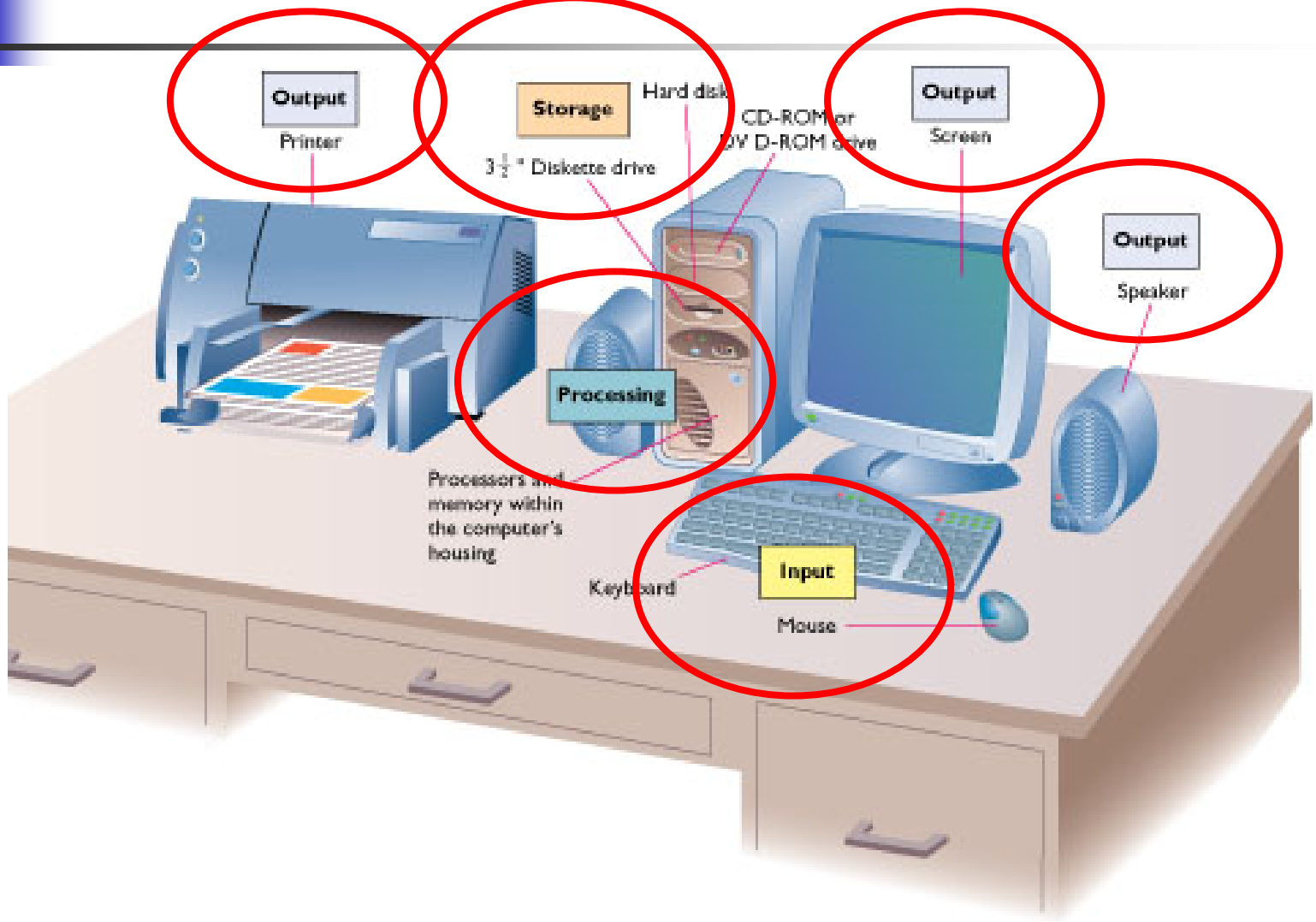
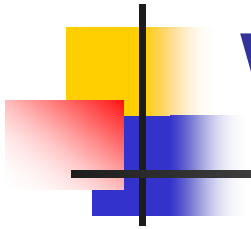


The functionality

Managing more than Operating

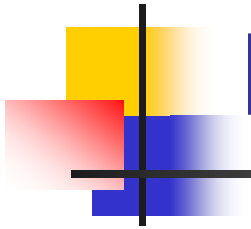
Remember This?





What to Manage

- Processing
  - CPU and Memory
- Storage
- Input and Output Devices

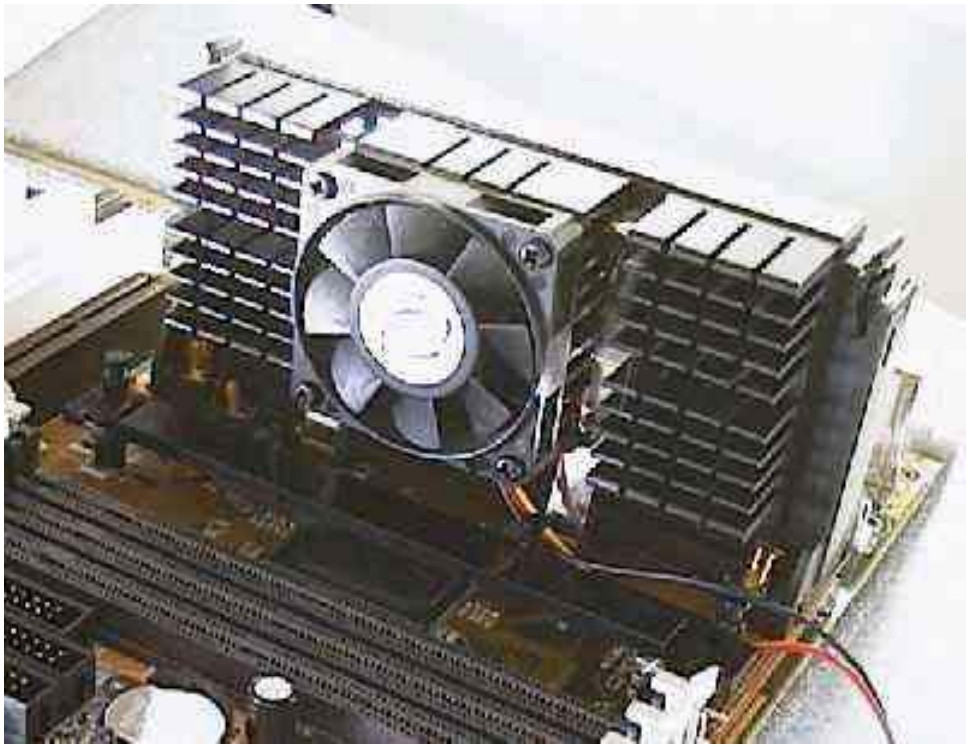


Functions

- CPU - Process management
- RAM - Memory management
- Storage – File system
- I/O – Device drivers

Process Management

CPU at Work

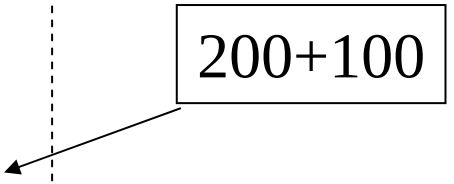


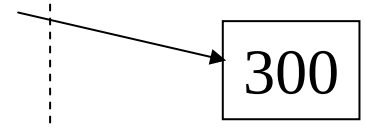
200+100

300

In Details

1. $(200)_{10}$ -> $(011001000)_2$
2. $(100)_{10}$ -> $(001100100)_2$
3. $(011001000)_2 + (001100100)_2$ -> $(100101100)_2$
4. $(100101100)_2$ -> $(300)_{10}$





Computer's View ⟷ User's View

Granularity

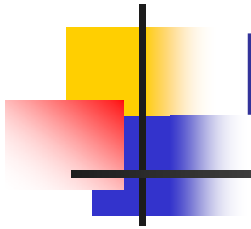
1. $(200)_{10}$ -> $(011001000)_2$
2. $(100)_{10}$ -> $(001100100)_2$
3. $(011001000)_2 + (001100100)_2$ -> $(100101100)_2$
4. $(100101100)_2$ -> $(300)_{10}$

200+100

Instruction

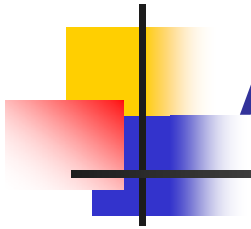
Process

Computer's View <----------> User's View



Process

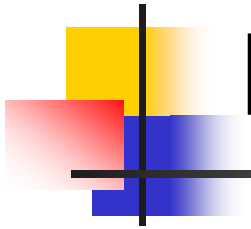
- A task started by an user or another process
  - Starting MS Word
  - Cut and paste in MS Word
  - Save files
- Each task could take the CPU a while to complete



Average Computer

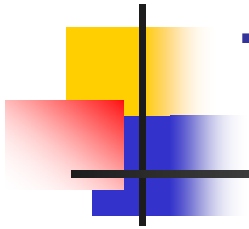
- 1 CPU
- Therefore, one instruction at a time

- However, it is not necessary that one process runs from the beginning to the end without interruption
- Have you tried to start multiple applications all at once?



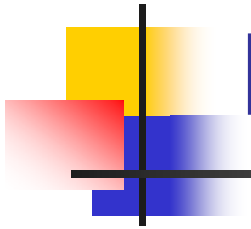
Running Model

- Uni-programming
  - Must terminate one before the next one can start
  - Ex. MS-DOS
- Multi-programming (Multi-tasking)
  - Multiple processes progressing at the same time
  - CPU still works on one process at a time
  - Can get around slow processes
  - Ex. Pretty much all other OS's

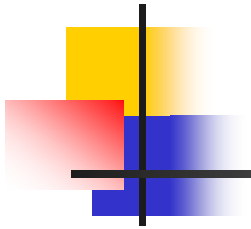


Types of Multi-programming

- Event-driven
  - Switch from one process to another by triggering events
- Time-sharing
  - Switch from one process to another based on the share of CPU time

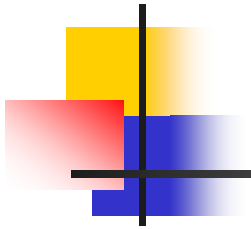


Ex. Jorge (CPU) and 3M (processes)



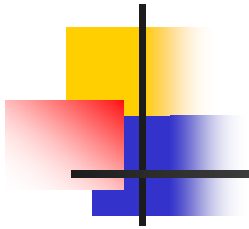
Event-driven

- When processing needs to be temporarily suspended, an interrupt is generated
- This is a signal to the operating system to evaluate the cause of the interrupt and determine who should now have CPU time



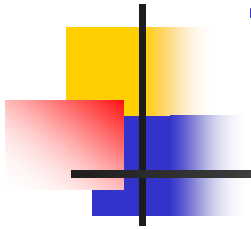
Event-driven Example

- Two programs are running – Payroll and Inventory Management
- Payroll needs to read an employee record
- Payroll generates an interrupt
- Normal processing is temporarily suspended
- The CPU looks at the interrupt and initiates the read operation
- While waiting for the read to complete, the CPU begins processing the Inventory Management program



Event-driven Example (cont.)

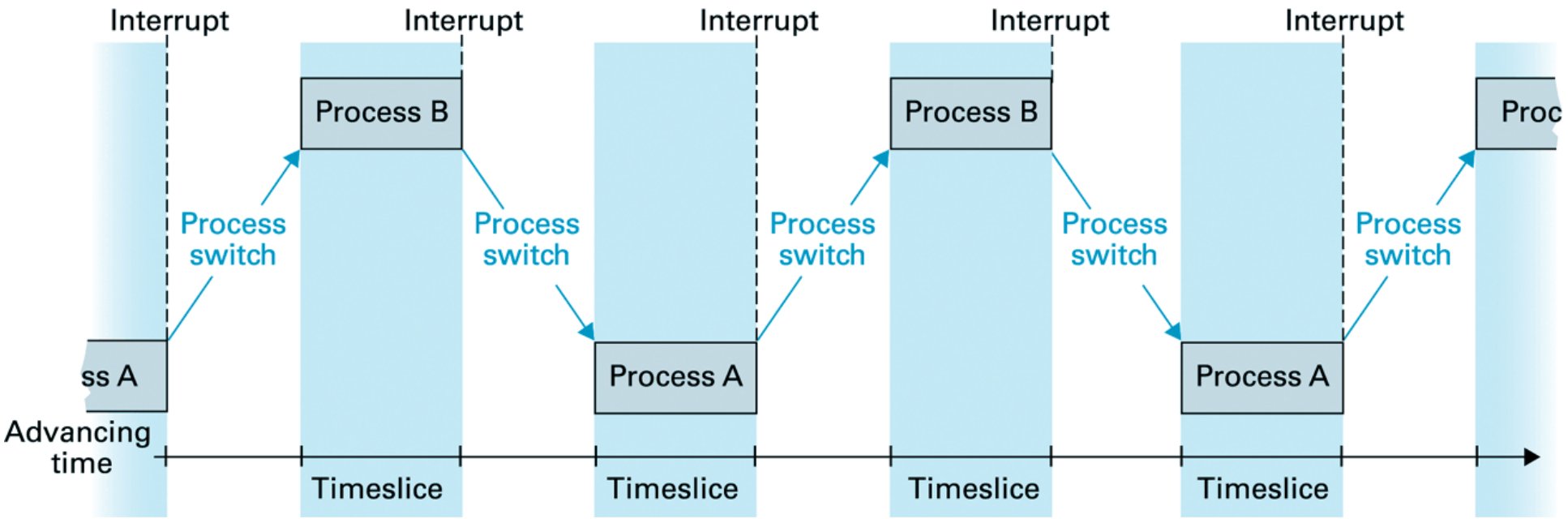
- When the read operation is complete, another interrupt is generated
- Normal processing is temporarily suspended
- The CPU looks at the interrupt and determines its cause
- The CPU will either continue processing the Inventory Management program or return to the Payroll program depending upon their priority

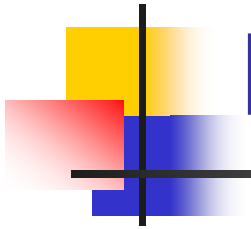


Time-sharing

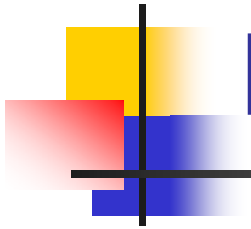
- A small fraction of CPU time is allocated to the program
- The time slice ends
- The CPU begins processing a different program
- Response time can vary depending upon the number of processes on the system

Time sharing



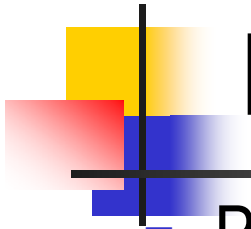


Ex. Jorge (CPU) and 3M (processes)



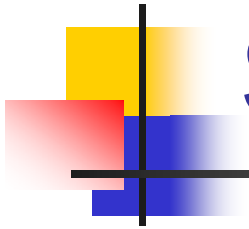
Priority

- Can be in many forms
  - Time share
  - Foreground, background
  - User/system assigned



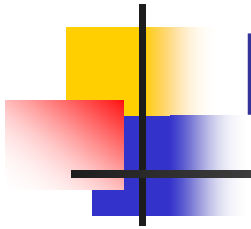
Foreground and Background

- Programs are placed in either **Foreground** or **Background**
- Programs in **Foreground** have priority for CPU time
- While performing read / write operations for the Foreground program, the CPU gives time to a program in Background
- Programs are placed in a holding queue while waiting to run



Scheduling

- Processes are sorted by the priority
- The highest-priority one gets processed when interrupt occurs
- The interrupted process gets re-prioritized and inserted into the sort list
- This is so called process scheduling

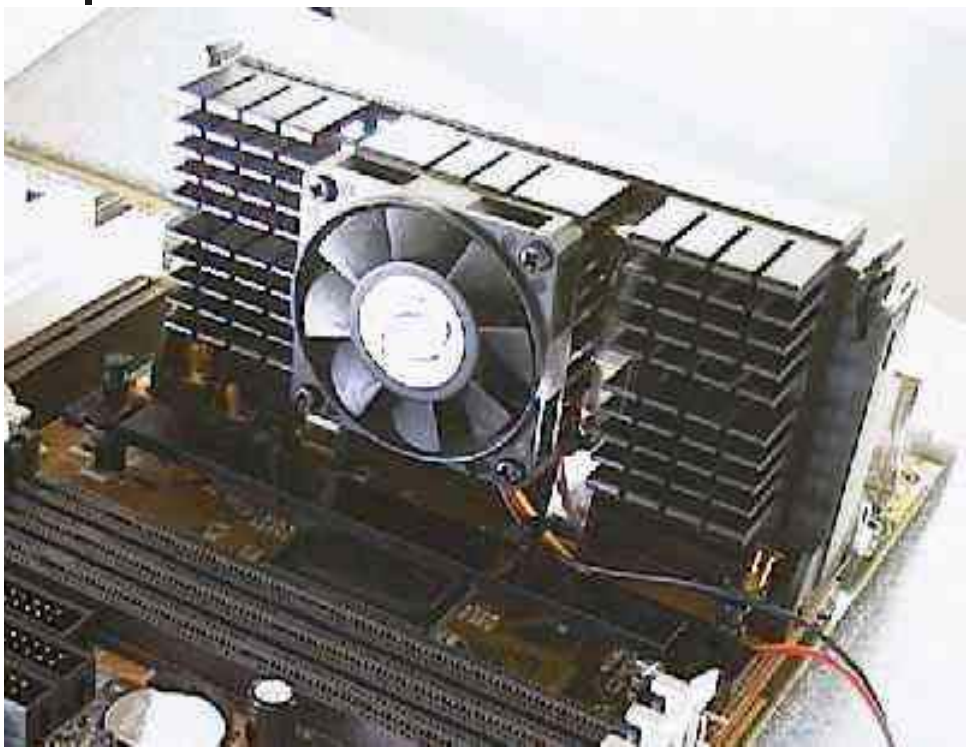


Parallel Computers

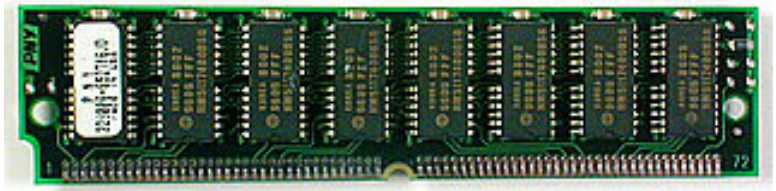
- Multiple CPUs
- Can process several programs simultaneously
- One program can be divided (with caution) and executed on multiple CPUs
  - Speed up
- Compared with pipelining
  - Inside a CPU but several instructions at the same time

Memory Management

Not Quite as Simple

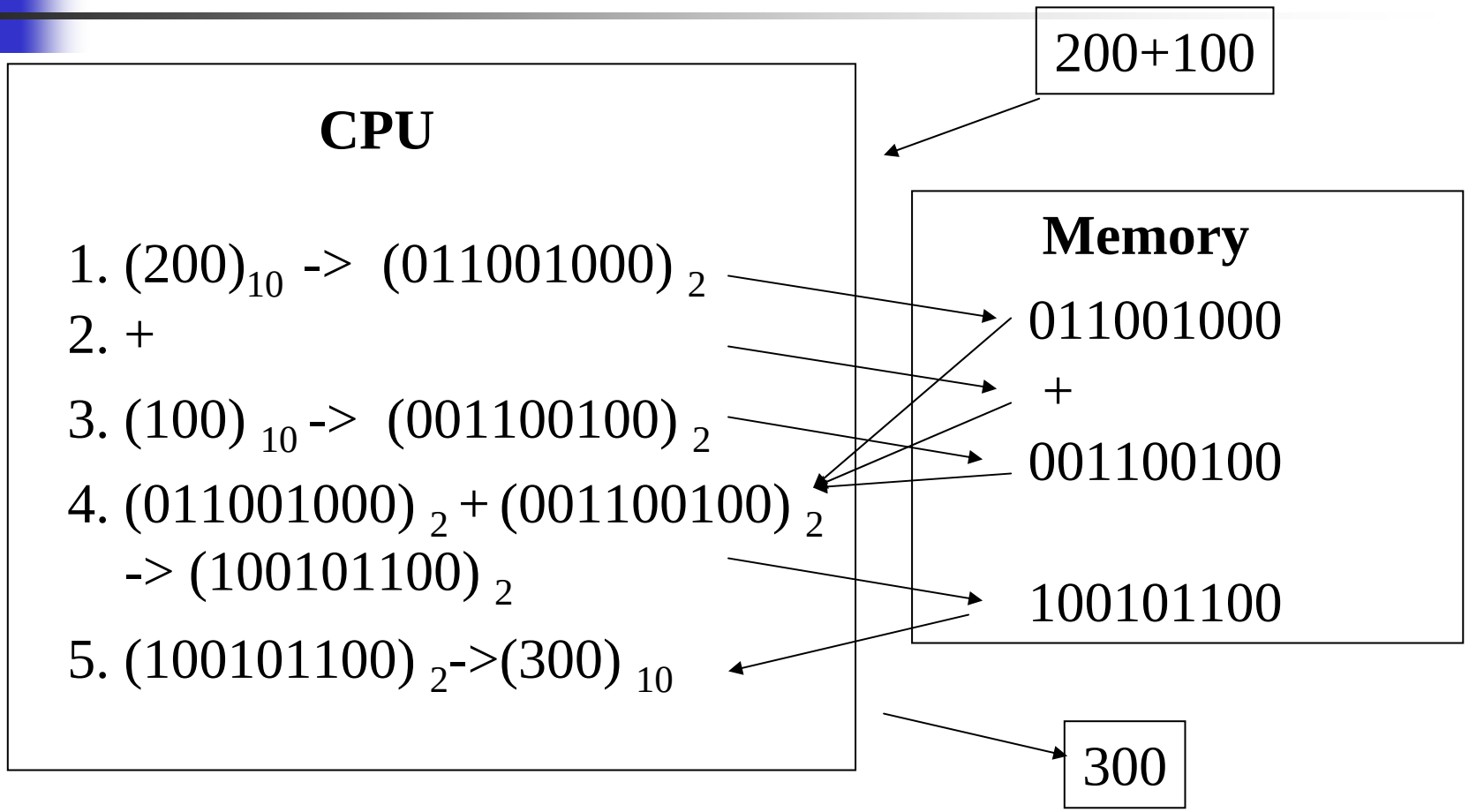


200+100

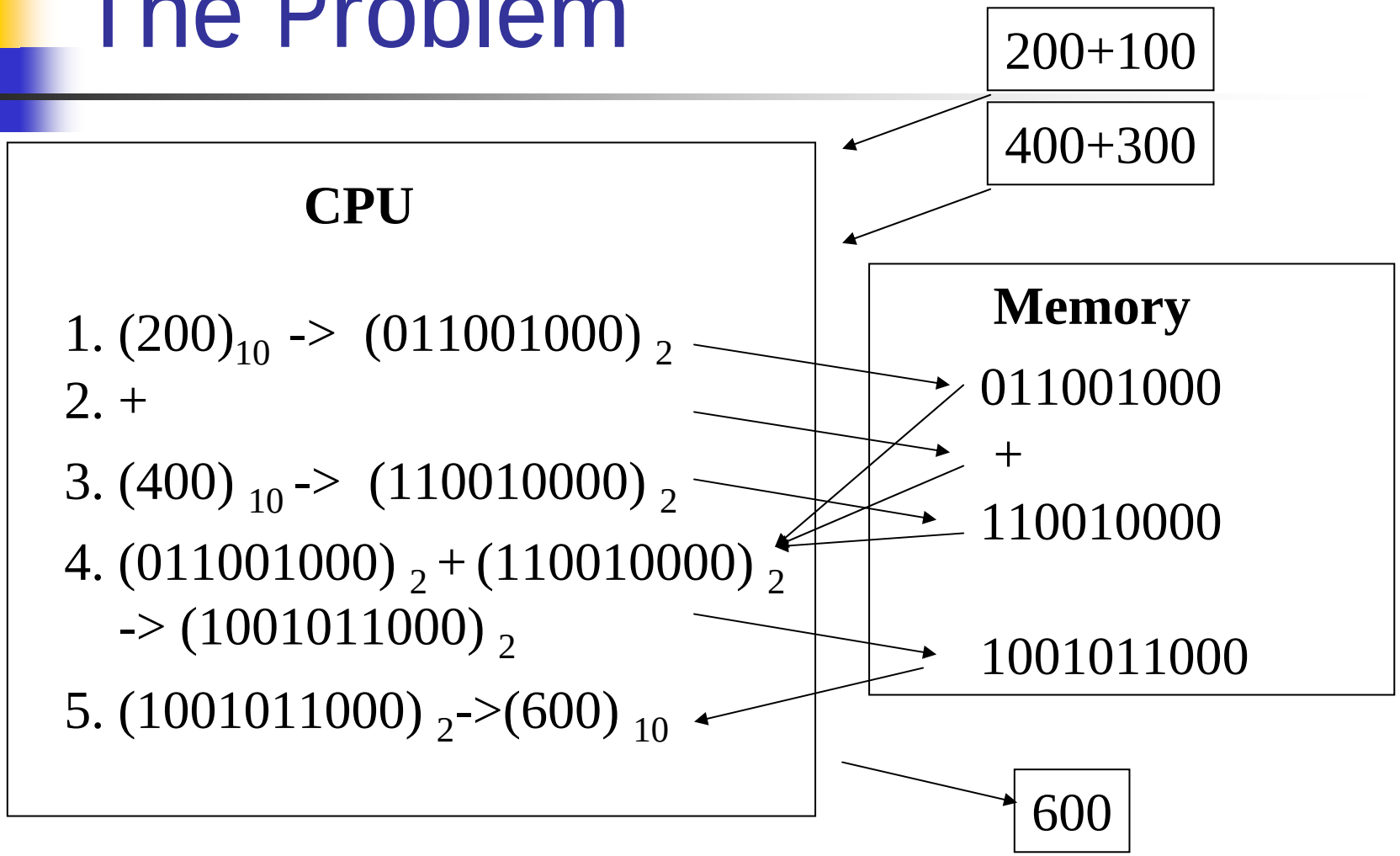


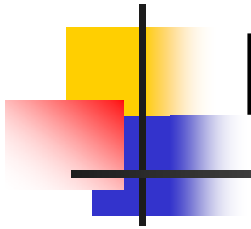
300

In Reality


200+100
CPU
1. (200)10 -> (011001000) 2
2. +
3. (100) 10 -> (001100100) 2
4. (011001000) 2 + (001100100) 2
-> (100101100) 2
5. (100101100) 2->(300) 10
Memory
011001000
+
001100100
100101100
300

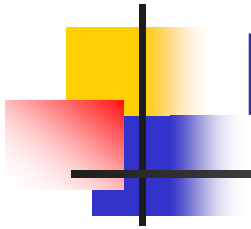
The Problem


200+100
400+300
CPU
1. (200)10 -> (011001000) 2
2. +
3. (400) 10 -> (110010000) 2
4. (011001000) 2 + (110010000) 2
-> (1001011000) 2
5. (1001011000) 2->(600) 10
Memory
011001000
+
110010000
1001011000
600



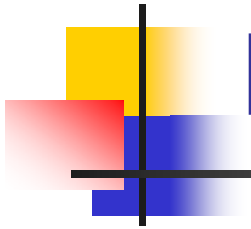
Program

- Program must be in memory to be executed
- Memory space for each program must not overlap



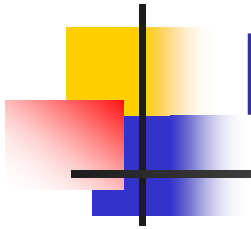
Memory Management

- The process of providing separate memory space to programs



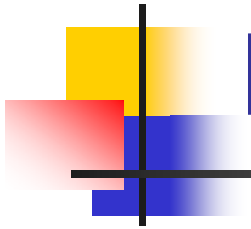
MM Methods

- Partitioning
- Paging
- Virtual memory



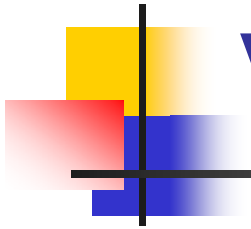
Partitioning

- Divide memory into partitions
- Fixed-size partitions
- Variable-size partitions



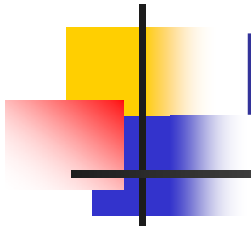
Fixed-size Partitioning

- What should be the size?
- Can’t be too small
  - The partition must accommodate the largest possible program
- Can’t be too large
  - May cause wasted memory space



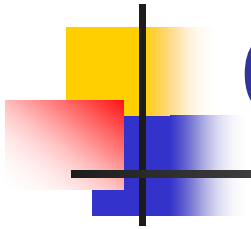
Variable-size Partitioning

- Sequential memory allocation
  - Program memory space interleaving
  - Tedious link list
- Sequential memory block allocation
  - Less memory space interweaving
  - Manageable link list
  - Memory blocks are referred to as **page frames**

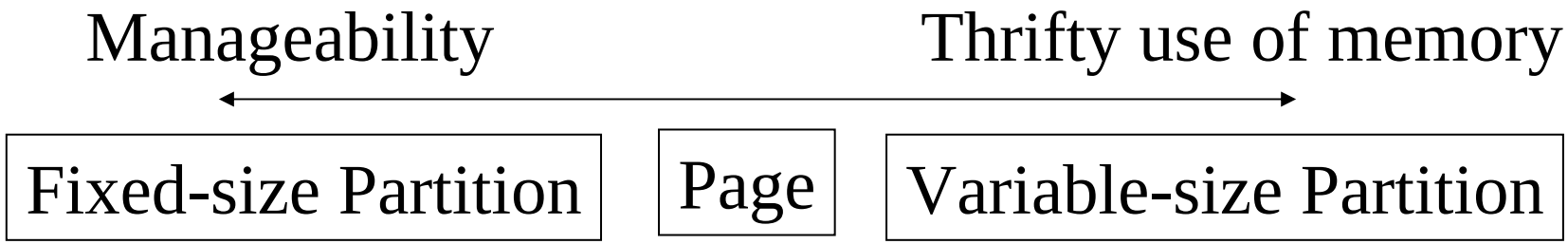


Paging

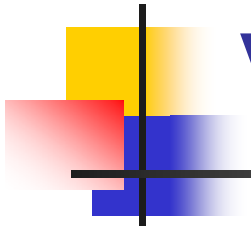
- Divide the program into equal-size pieces (pages)
- Store each piece in equal-size memory spaces (page frames)
- Typical size is 2KB or 4KB
- Create an index to each page and store in a Page Table



Comparison



Run out of RAM space!
Some parts of a program might not really be used…



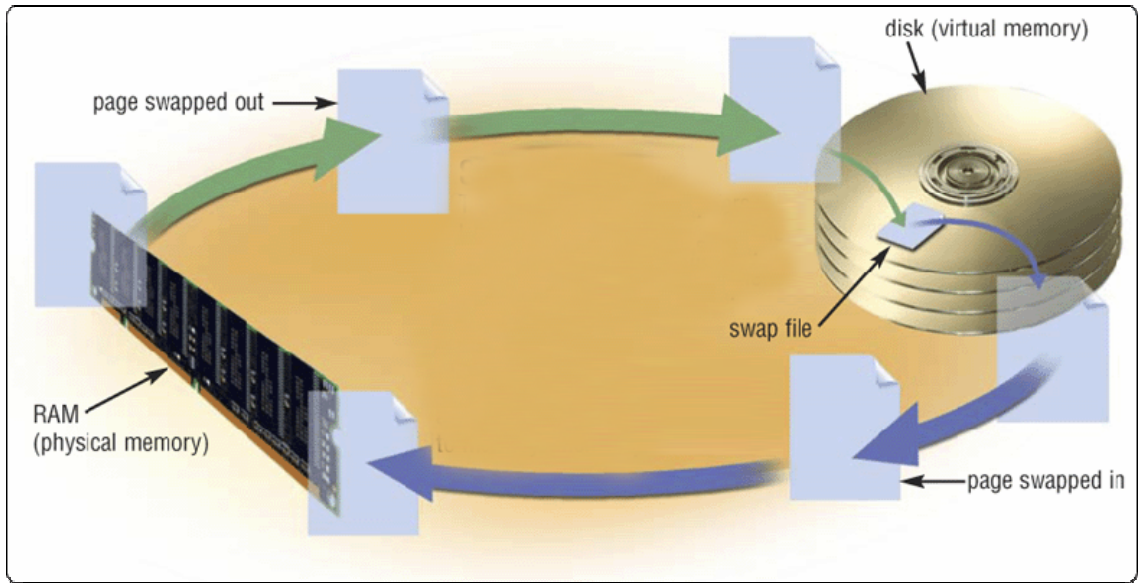
Virtual Memory

- A portion of the program is placed in memory
- The remainder is on disk
- Pages on disk will be brought into memory as needed (one page at a time)
- Referred to as the **Paging Process**

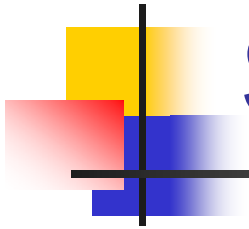
Virtual Memory

- **With virtual memory (VM), portion of hard disk is allocated to function as RAM**

Step 1. The operating system transfers the least recently used (or oldest) data and program instructions to disk because memory is needed for other functions.

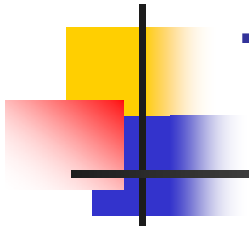


Step 2. The operating system transfers data and program instructions from disk to memory when they are needed.



Swapping

- In there's no free space on the physical memory, some pages need to be discarded
- This is referred to as the swapping process
- Many ways to select the discarded pages
  - Oldest
  - Least Recently Used (LRU)

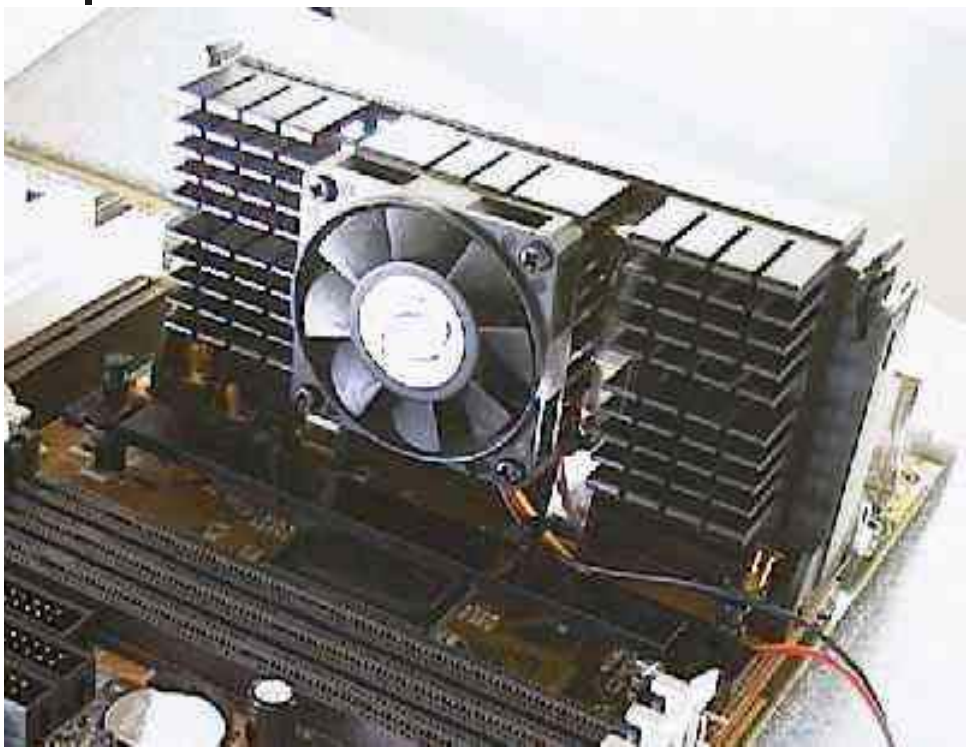


Thrashing

- Too large a portion of CPU time is spent locating the correct page and bringing it into memory

File System

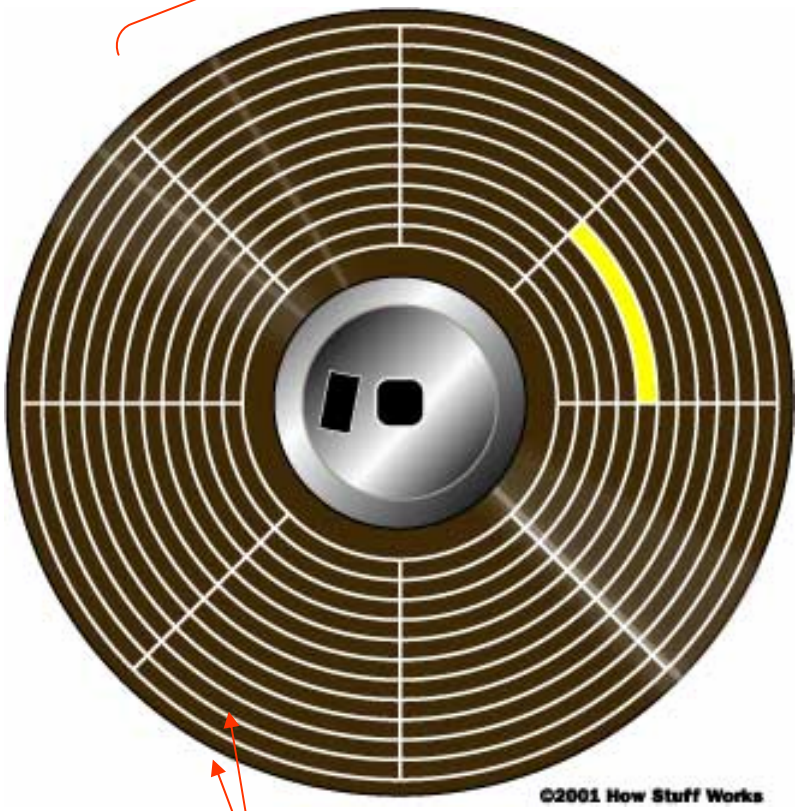
Again

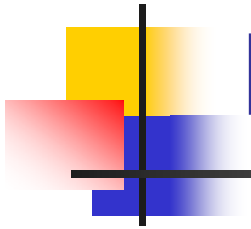


200+100

300

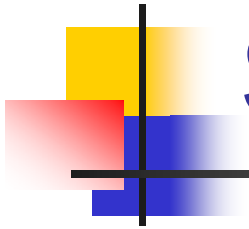
Disk Structure





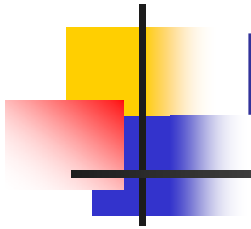
Direct/Random Access

- Files are not physically stored in any order
- Update in place
  - Read/write to the file's place on disk
- Such storage devices are called Direct-Access Storage Device (DASD)
  - For example, a hard disk



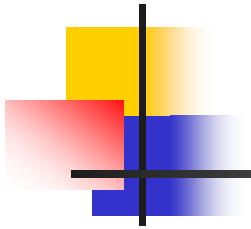
Sequential

- Records are stored and accessed in order
- All files prior to the one requested must be read
- For example magnetic tapes



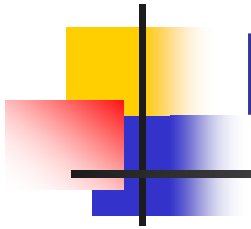
Locating Files

- Sequential
  - Have to go through previous files anyway
  - No intelligence
- Direct access, the old way
  - Hashing – apply a formula to the filename to produce the address
  - Collision – same address for different filenames
  - A simplified example



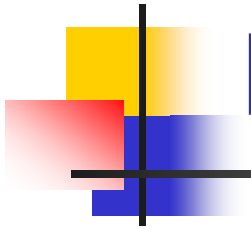
Indexing

- Direct Access, the new way
- Files are stored sequentially or randomly
- Index is generated that contains filename and address



Records in a File

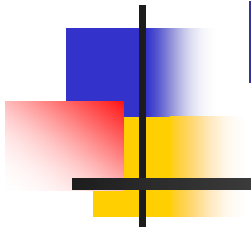
- Similar to Files on a Disk
- Records can be stored sequentially or randomly
- Index is generated containing record key and address



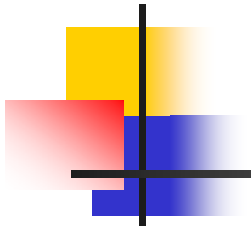
Directory vs. File

- Directory – index of files
- File – index of records

- And so on so forth
  - Record – index of sub-records
- And vice versa
  - Super-directory – index of directories



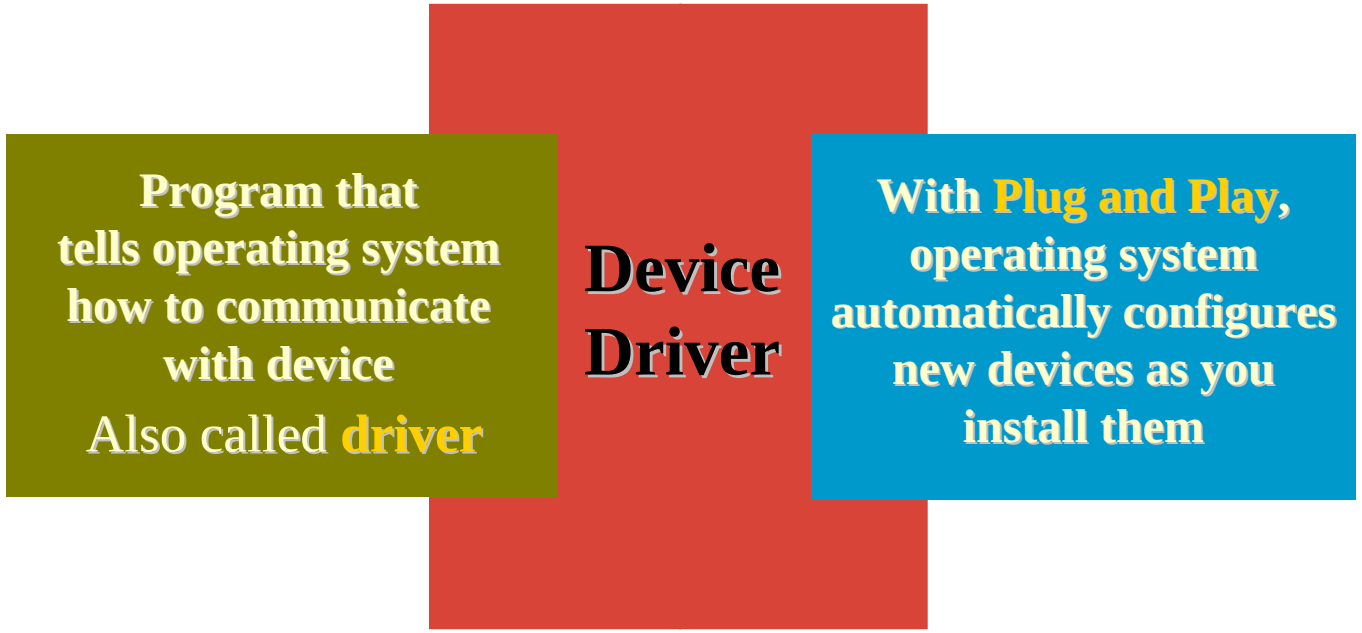
Input/Output Management

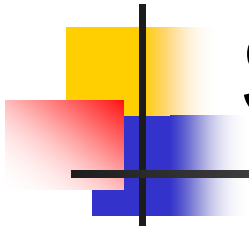


I/O Management

- OS keeps track of the I/O requests
- OS processes I/O requests in order received
- Except print jobs

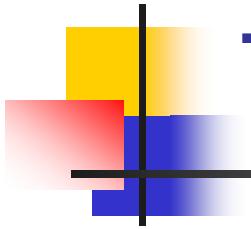
Device Driver





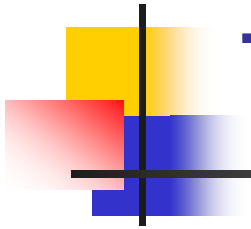
Sharing a Printer

- A printer is shared by multiple active processes
- Printouts are generated in pieces as the CPU gives each concurrent program some time



The Problem

- The current program may generate a few print lines
- The CPU moves to the next program
- The second program may generate a few print lines, etc.

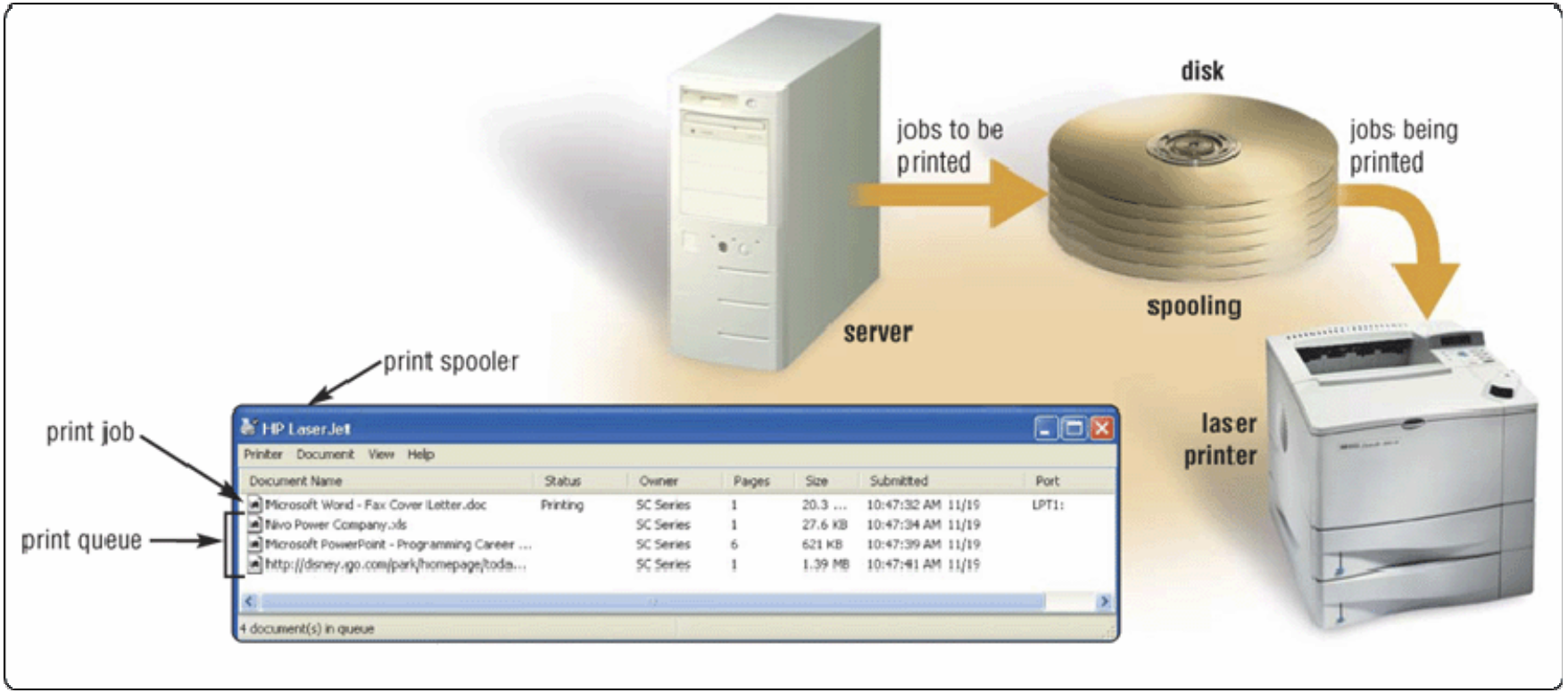


The Solution

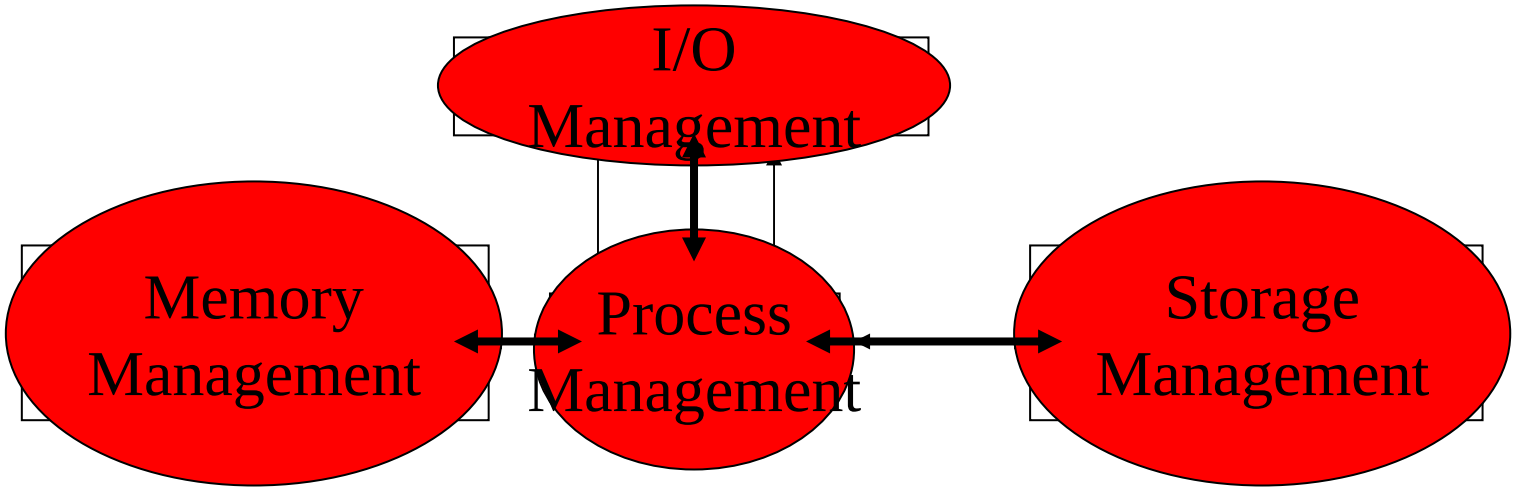
- Spooling
- Each program thinks it is writing to the printer
- The program actually writes to the hard disk
- When the program is complete, the file on the hard disk is sent to the printer

Printer Spooling

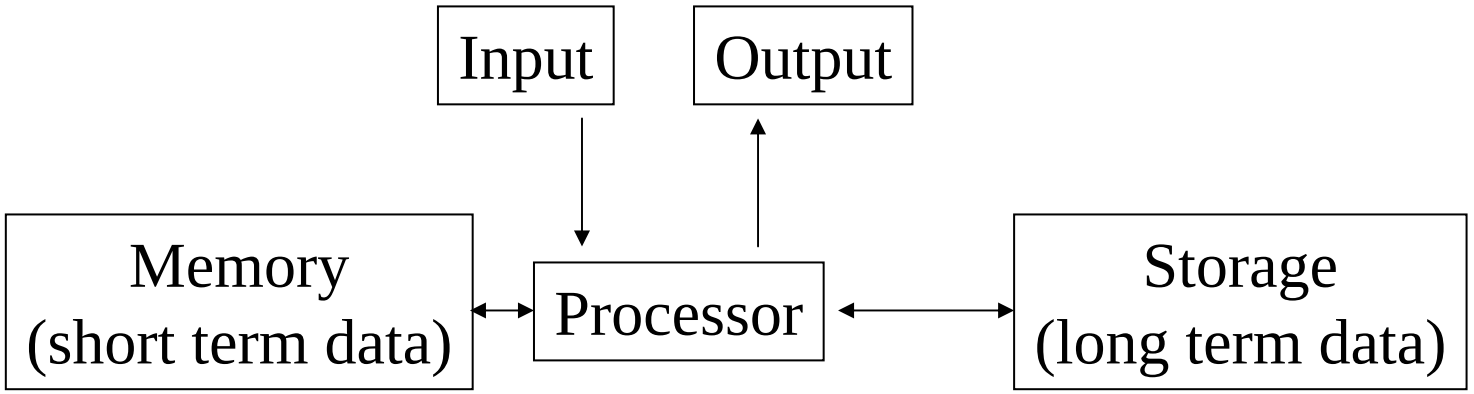
- **Sending print jobs to buffer instead of directly to printer**
- **Print jobs line up in queue**



Computer System



Computer System



Operating System