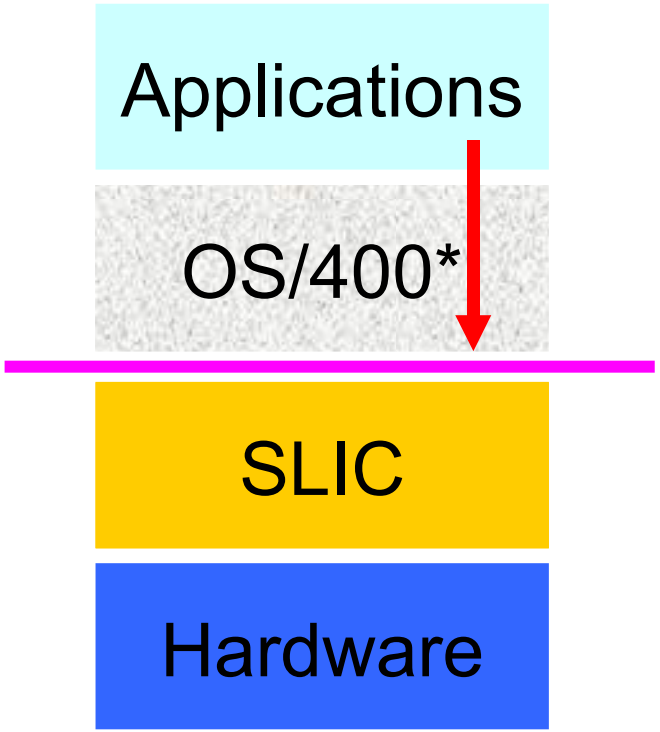
# iSeries Architecture – Part 2

*Module 3*

# The Five Sacred Architecture Principles of System i

- Technology Independance
- Object-based Design
- Hardware Integration
- Software Integration
- Single-Level Store

# Technology Independent Machine Interface




Direct access to hardware is not allowed!

## Machine Interface (MI)

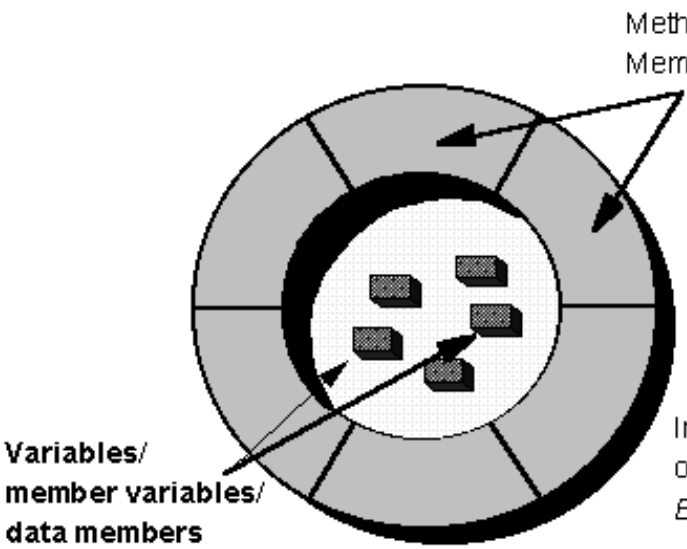
- or Technology Independent Machine Interface (TIMI)
- logical not physical interface

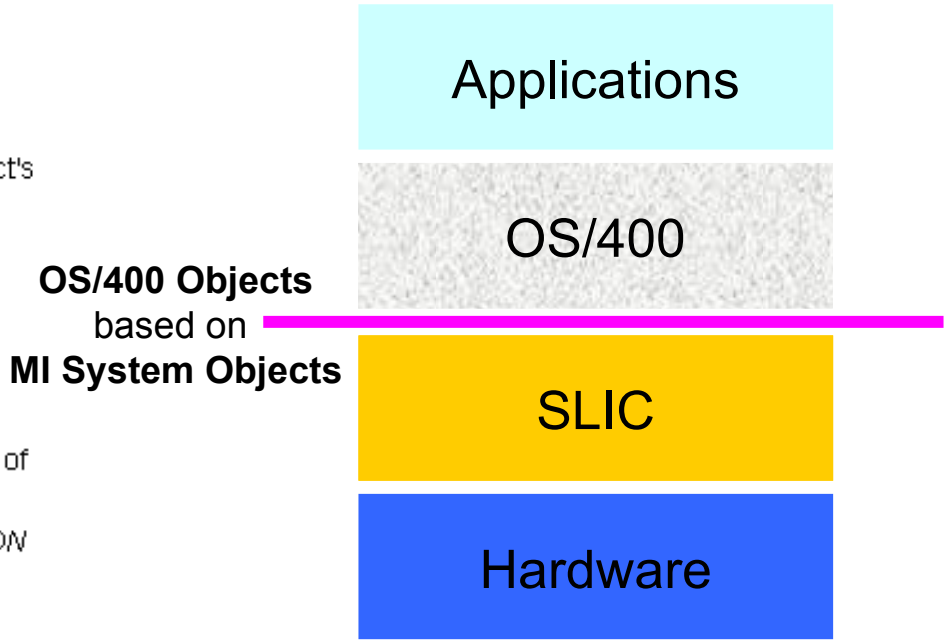
## System License Internal Code (SLIC)

- insulate applications from underlying hardware
- SLIC is hardware dependent!

* called i5/OS on i5 systems

# Remember from Module 2



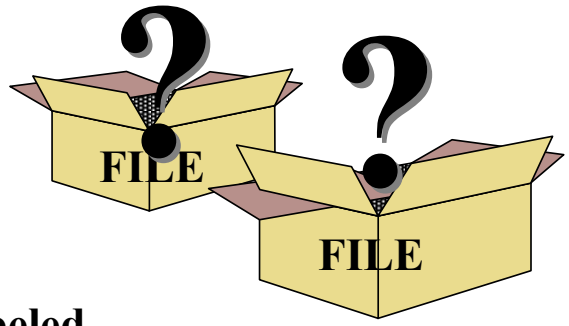

- **Objects are used for all user or system data structures**
- **Encapsulation of objects**
  - -Only valid functions allowed for each object
  - -Improved data integrity and security
    - -*PGM - executed
    - -*FILE – processed
- **Object-based OS (iSeries objects don‘t inherit from other object types)**

# 3.1 Object-based Design

# Storing Informations

## Conventional Systems

- A string of bytes can be almost anything
- Anything in permanent store is a file

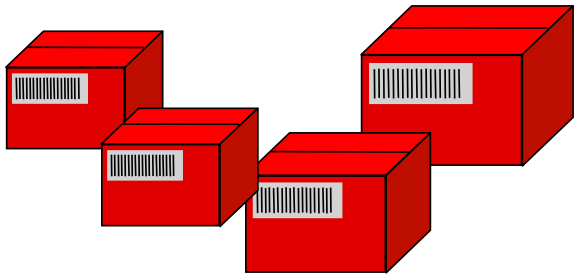



**Unlabeled**

Program?
Data File?
Control File
Batch File?


## System i

-Object Based Architecture
-Informations are encapsulated

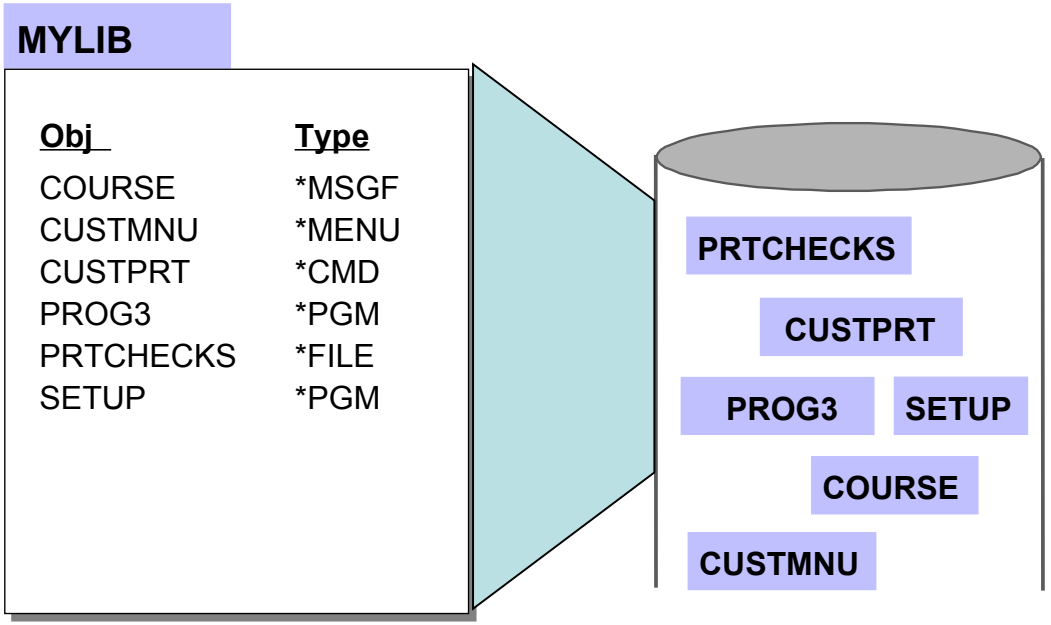


**Object header**

Type = File
Type = Program
Type = System Object
Type = User Profile
Type = Command

***There exists no virus for i5/OS.***

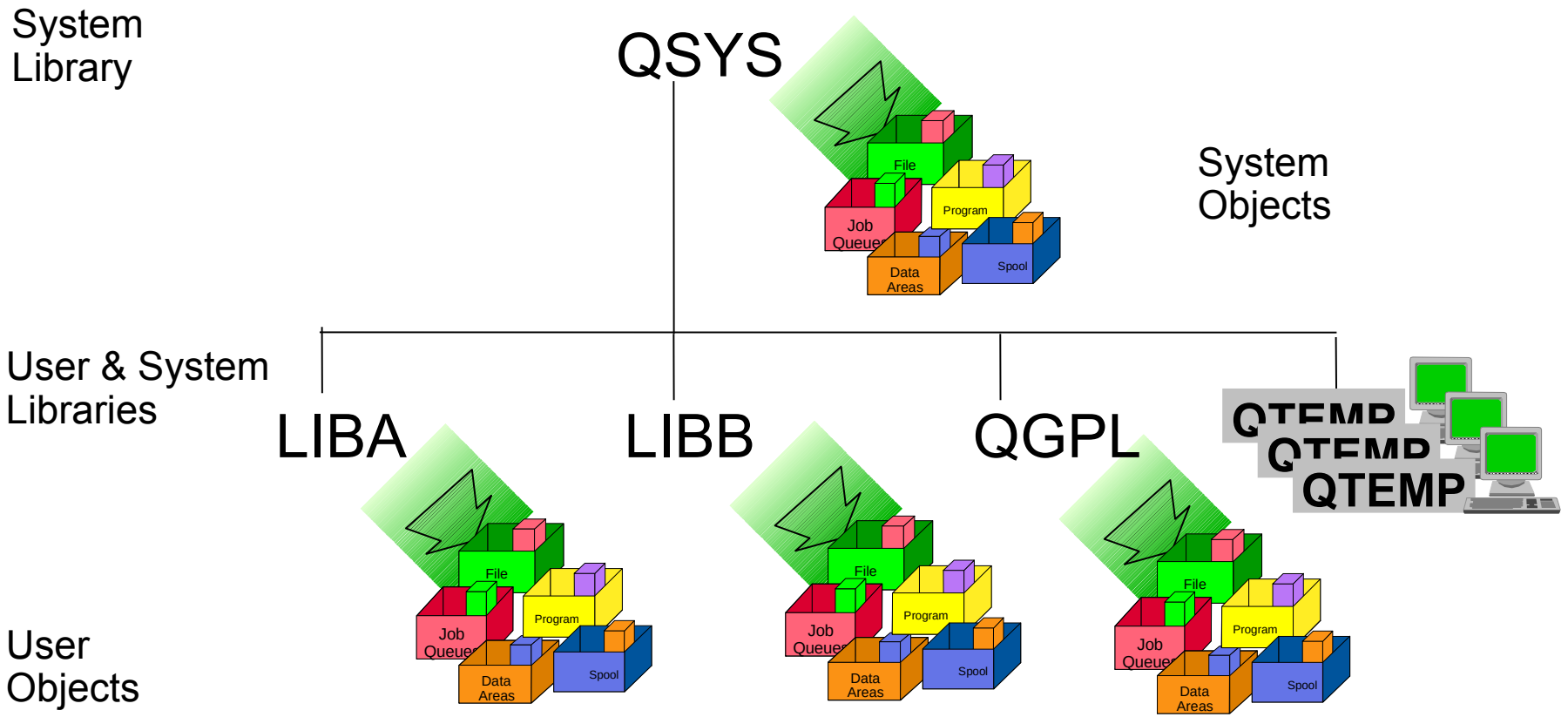
# What Is a Library?



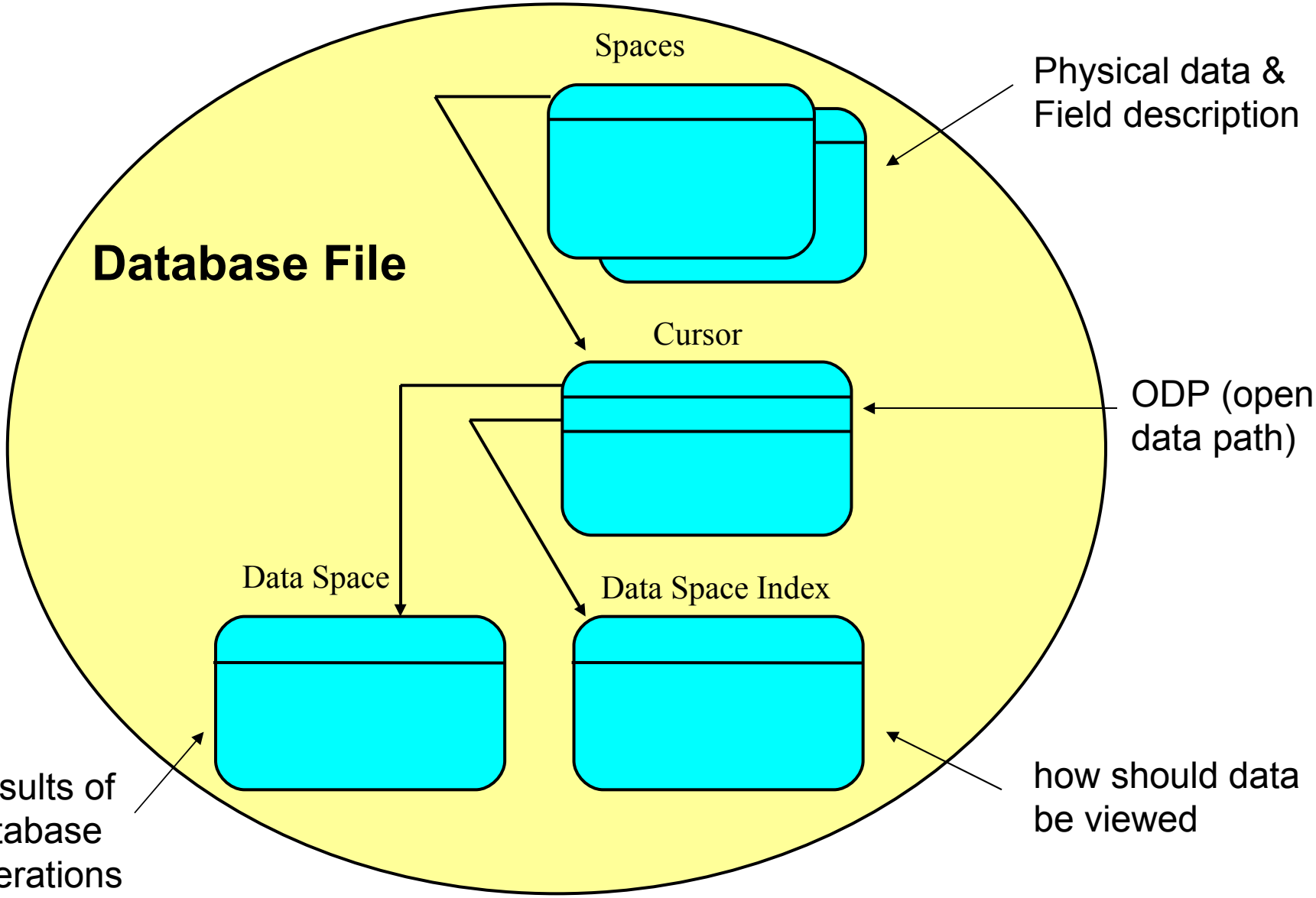

**Libraries are used to organize objects**

- For security reasons
- For backup reasons
- By application
- By owner
- By object type: program versus files
- By use: production versus test

# i5/OS Object File System

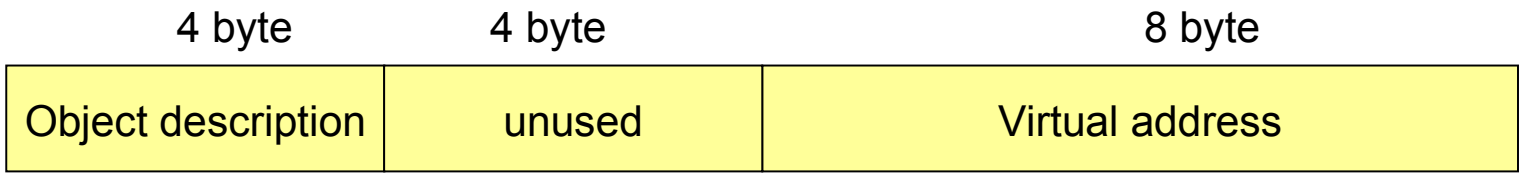


# i5/OS Object Composition Example



# MI Pointer

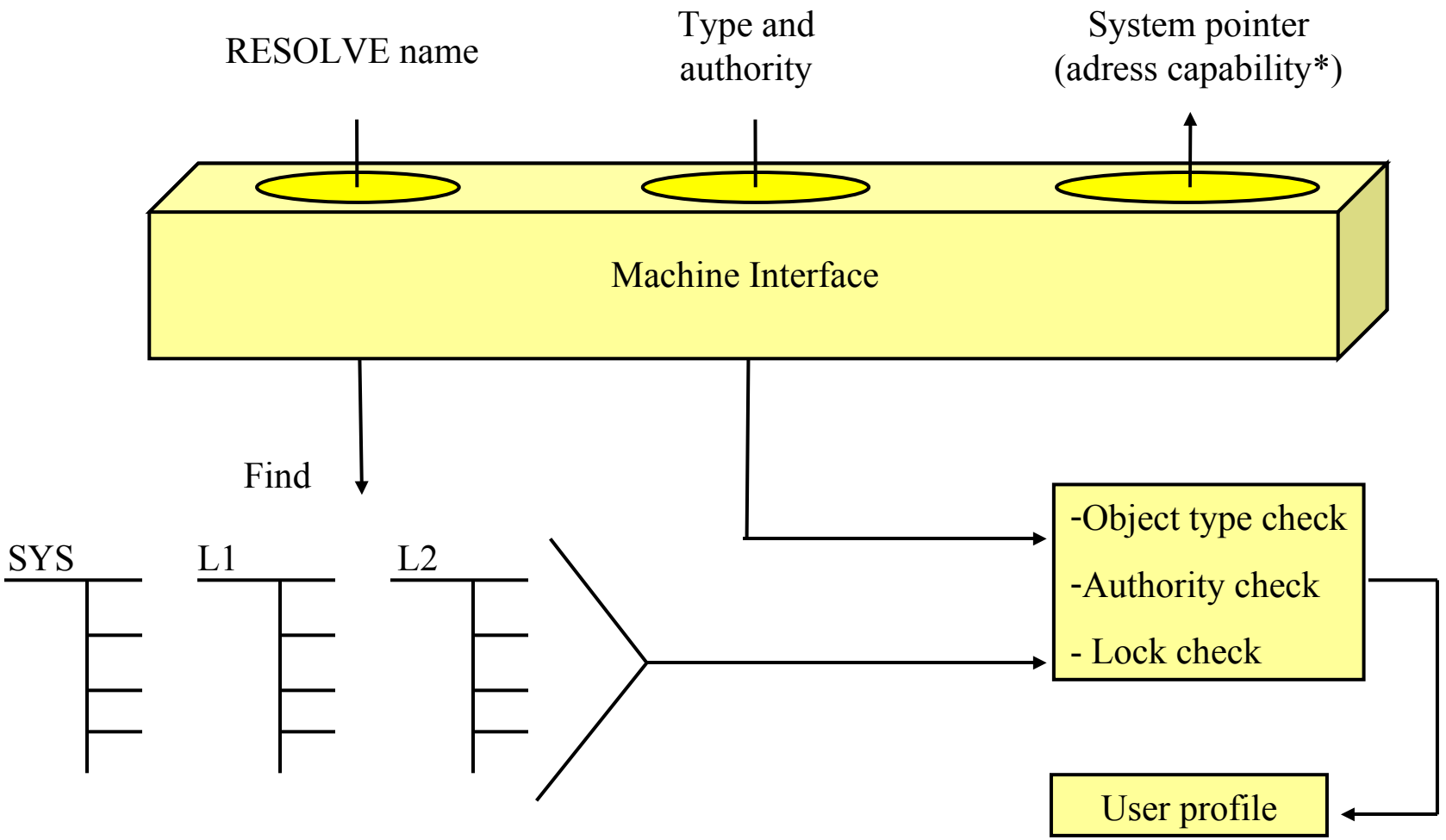
- 16 Byte long




Status bits
- type of pointer
  - System pointer
  - Space pointer
  - Data pointer
  - …
- Informations about the object
  - System pointer -> object type
  - Data pointer -> type of date
  - …
- Authorities
  - Could only be changed by OS in system state

Remarks:
- unused bits could be used to expand to 96-bit addresses without effecting any program above the MI
- type and object informations could also moved out of the pointer to go beyond 96 bits

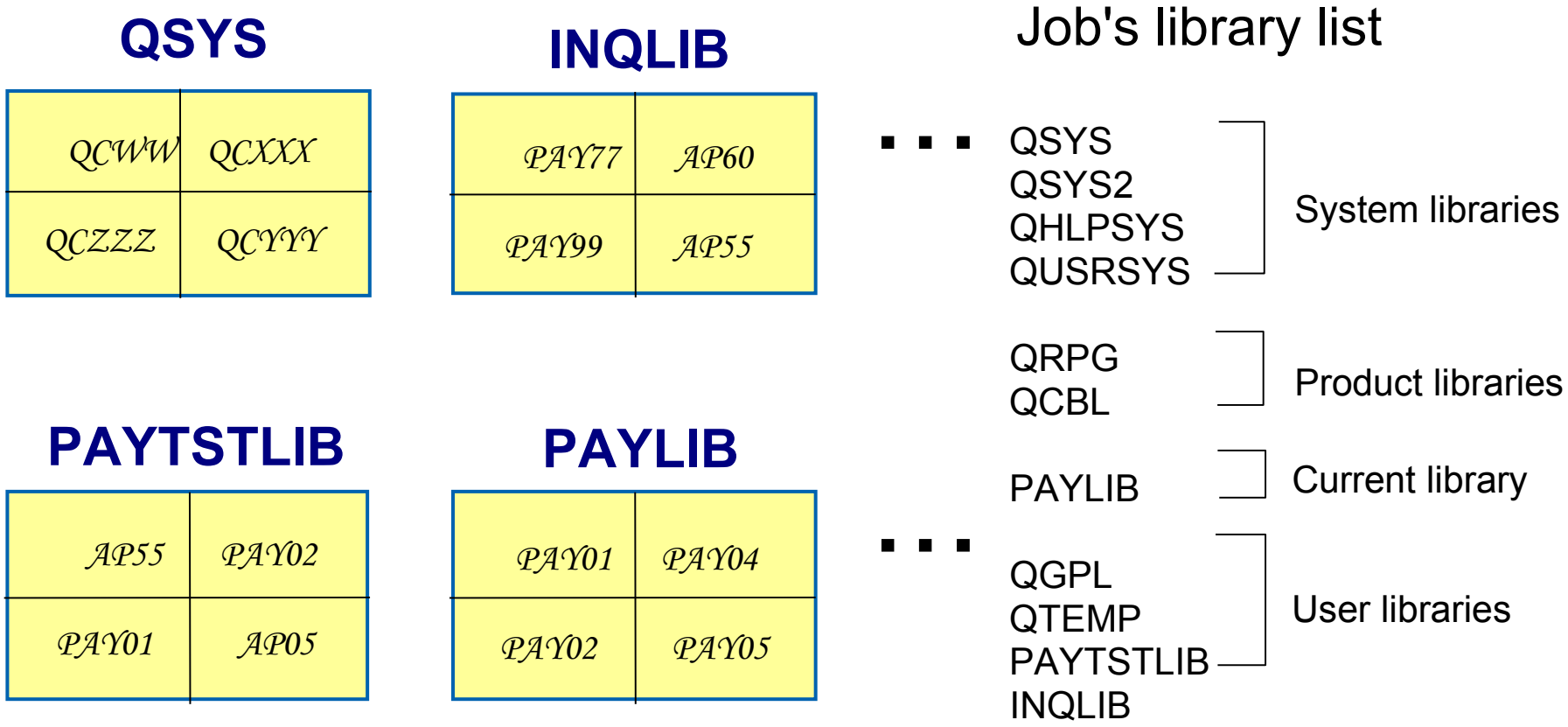
# Object Identification: Library, Name, Type



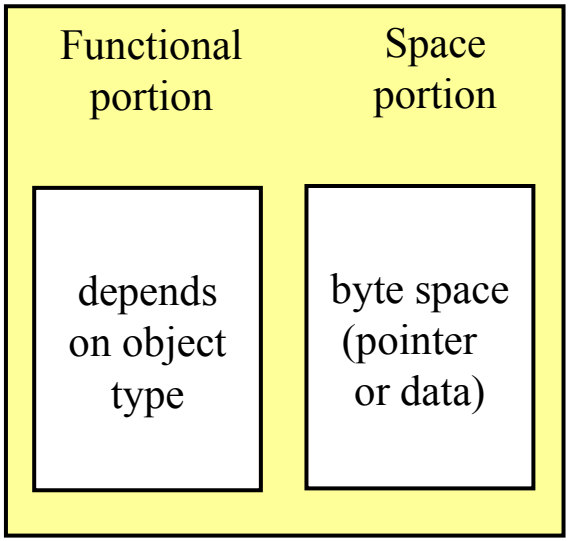

*A pointer containing the object address and the object authority.

# Finding an Object

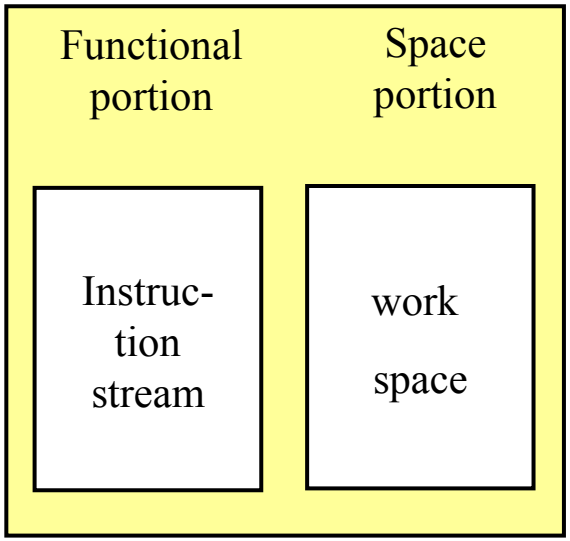
Simple name: CALL PAY02
Qualified name: CALL PAYTSTLIB/PAY02



# Internal Structure of System Objects



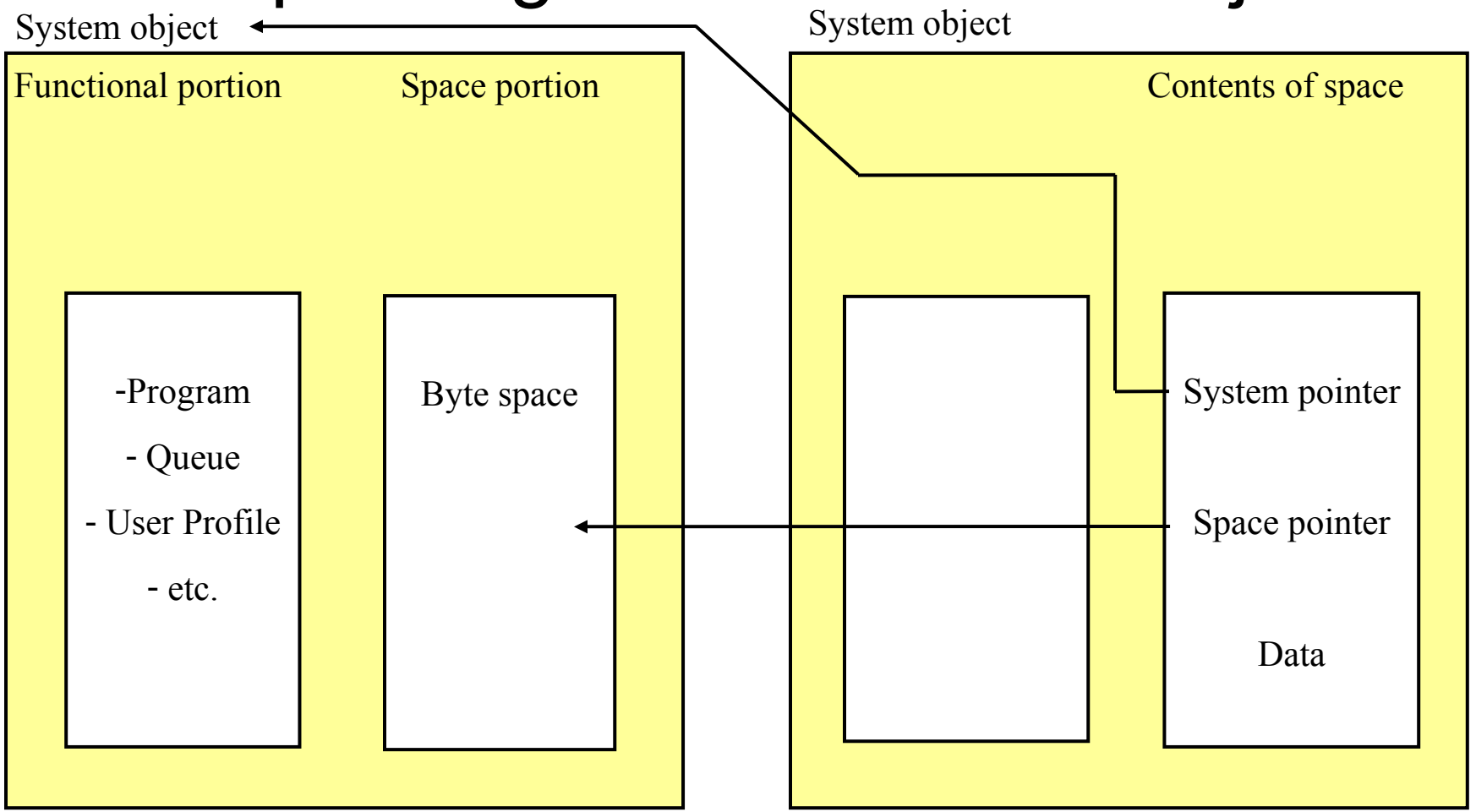

All system objects except spaces




Example: Program

**Spaces have no functional portion!**

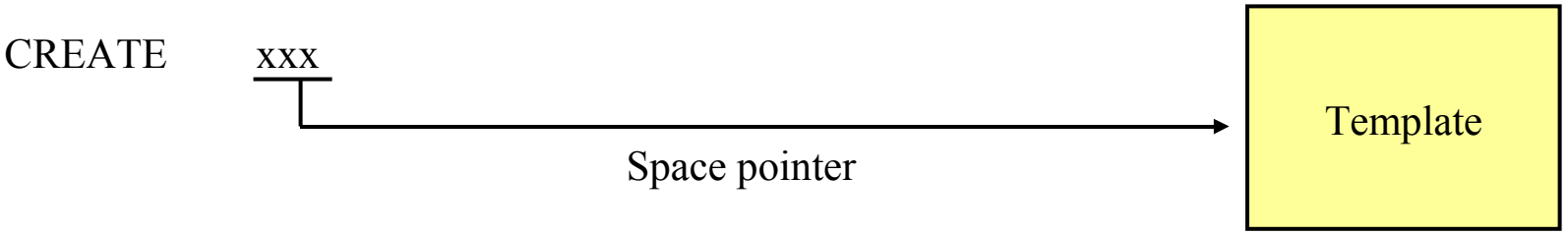
# Manipulating Data Inside an Object



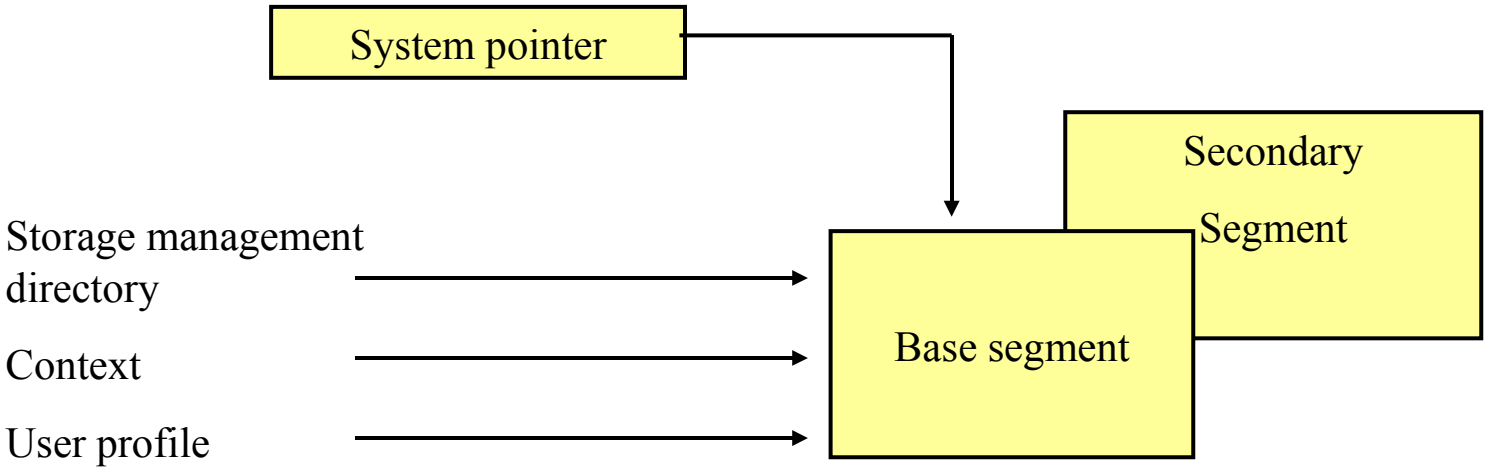

- System pointer can point only to the beginning of an object
- Space pointer points to a byte in the space portion
  - use space pointer to access and manipulate bytes in a space
- Space pointer can be modified by an MI program, system pointer cann‘t

# Object Creation

**INPUT**




**OUTPUT**




- System objects must be explicitly created with an CREATE instruction at MI
- A CREATE intructions references to an user-supplied template contained in a space object
- System pointers provides addressability of system objects

# Object Persistence

Object continues to exist in system memory

**forever**,

unless it‘s explicitly destroyed!

Sharing data between user means in conventional systems requires to store informations in a filesystem.

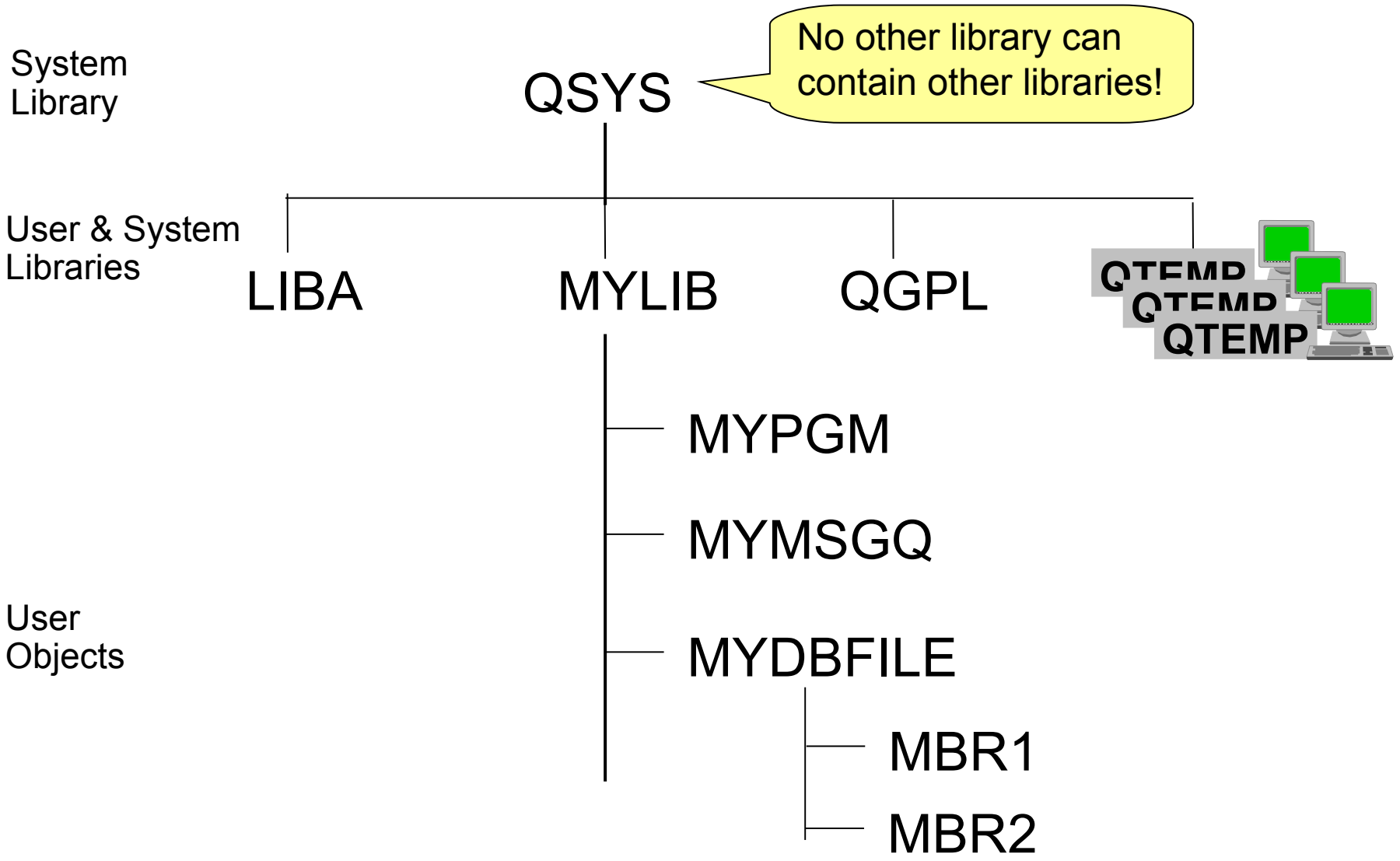
i5/OS have a single-level store!

**Security?**

**No filesystem?**

# 3.2 File Systems

# i5/OS Object File System



# Stream Files vs. Database Files

***Database files***

| | | | | |
|---|---|---|---|---|
| field-a | field-b | field-c | field-d | **record 1** |
| field-a | field-b | field-c | field-d | **record 2** |
| . . . | | | | |
| field-a | field-b | field-c | field-d | **record n** |

***Stream files***

abc1234567ABd8t4444X-+¶xxy2348RWY+?¶efg7654321XYZ~?¶a …

# Stream Files vs. Database Files

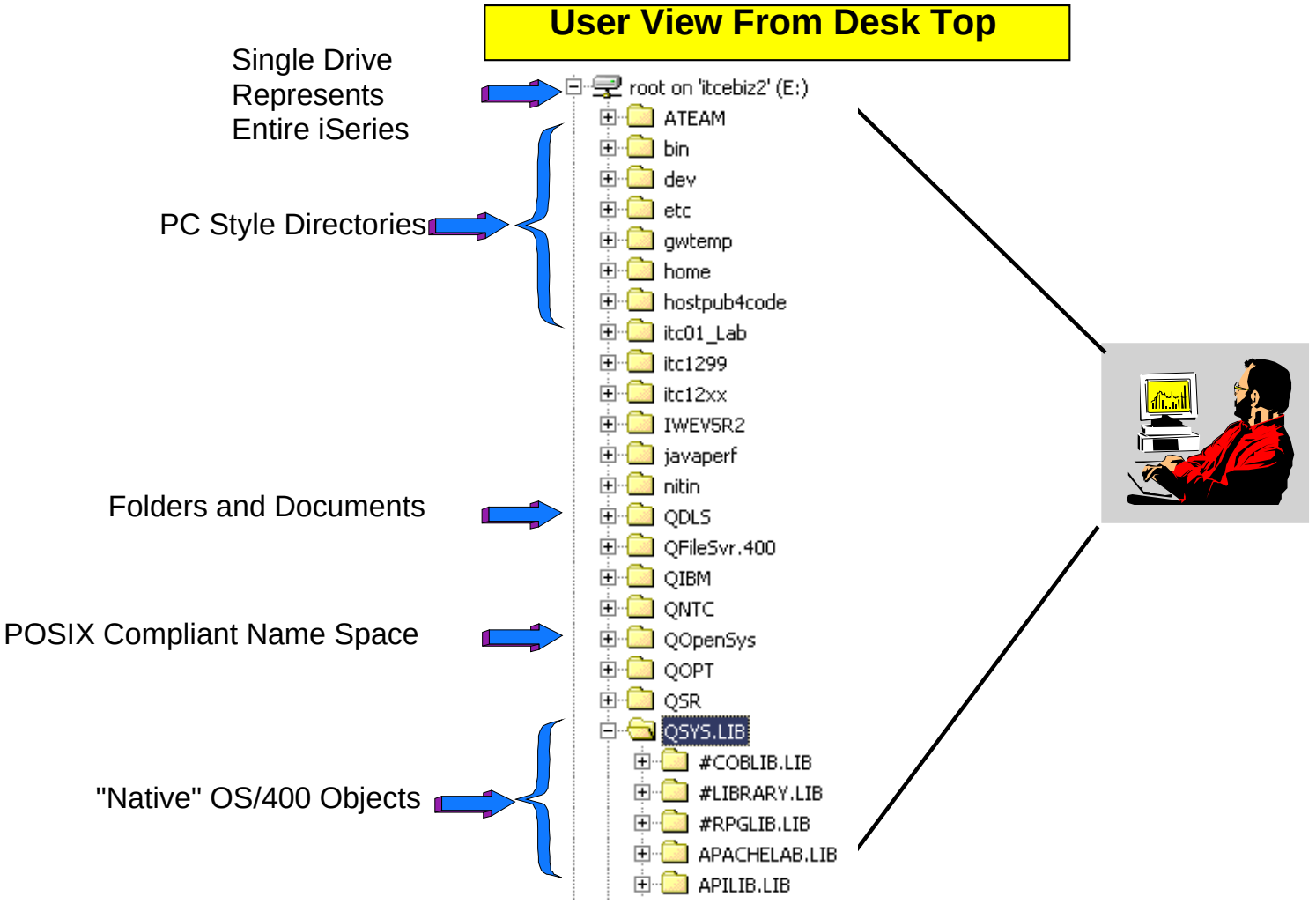
| | | | | | |
|---|---|---|---|---|---|
| 0001 | John | 23456 | 2345 | Highway 52N | Rochester MN |
| 0002 | James | 12345 | 25 | Frontage St | Greybull WY |
| 0003 | Douglas | 98765 | 1259 | Another St | Jackson WY |
| 0004 | Thomas | 56789 | 2345 | By Lane | Rapid City SD |

Native Database Files

| |
|---|
| 0001;John;23456;2345;Highway 52N;Rochester MN:: |
| 0002;James;12345;25;Frontage St;Greybull WY:: |
| 0003;Douglas;98765;Another St;Jackson WY:: |
| 0004;Thomas;56789;By Lane;Rapid City SD:: |

Byte-Stream Files

# Integrated File System

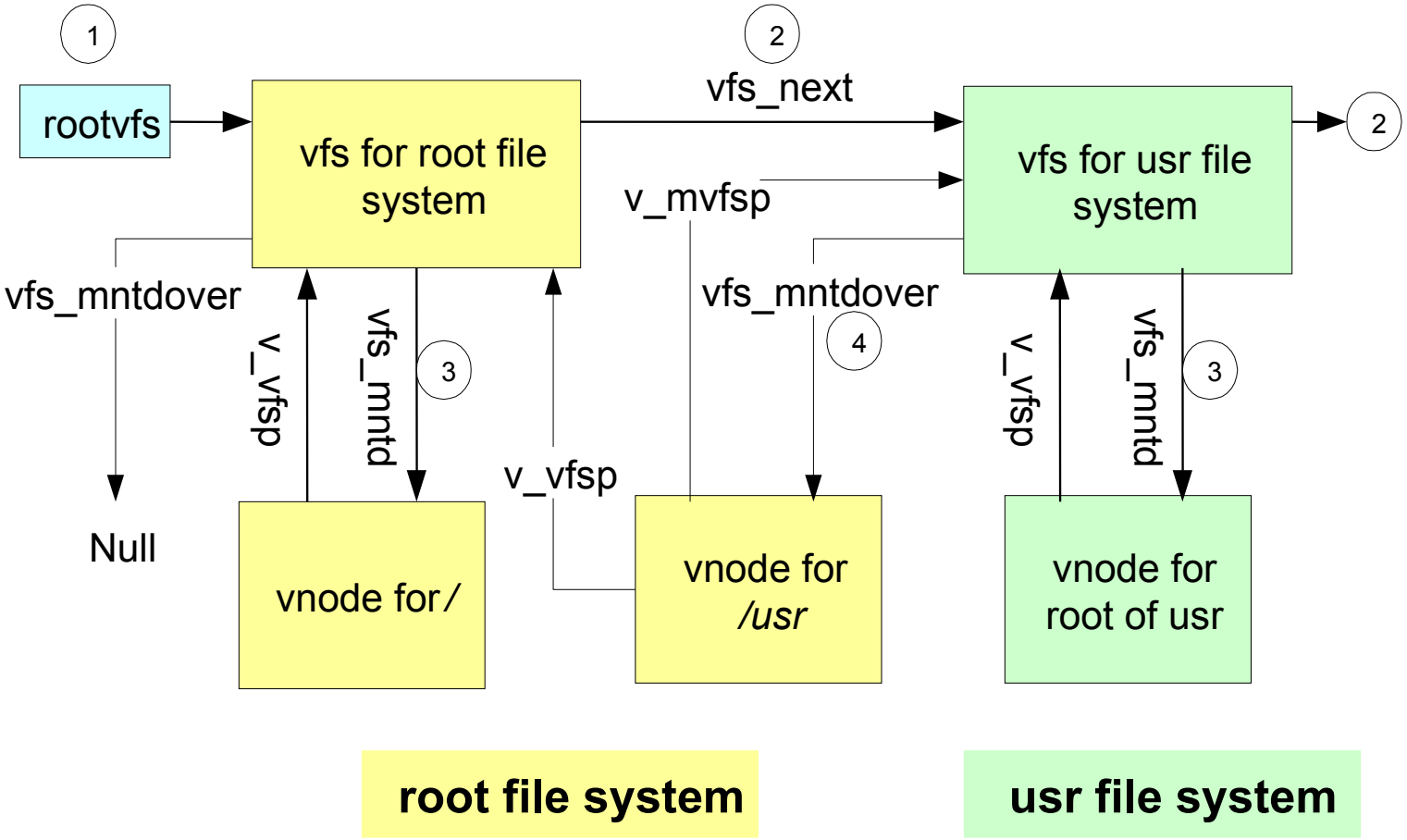


# Virtual File System (VFS)

- **Virtual file system** (VFS) is a standard and abstract software layer that allows the operating system kernel (e.g. the Linux kernel) to call file system functions without having to know the type of file system being used.
- A **vnode** is an object in kernel memory that speaks the UNIX file interface (open, read, write, close, readdir, etc.). Vnodes can represent files, directories, FIFOs, domain sockets, block devices, character devices.

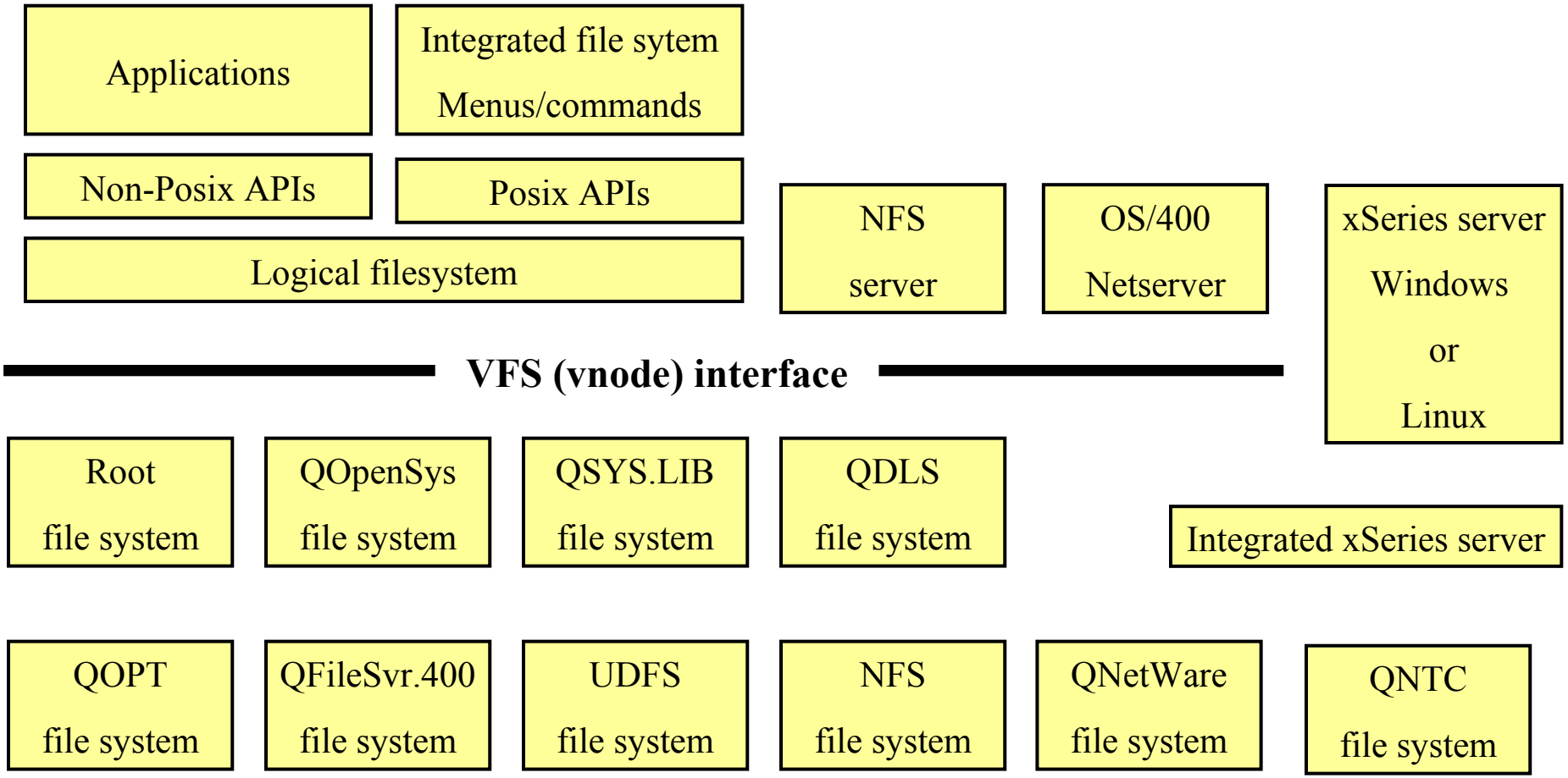
- Objectives:
  - **Both different types of Unix file systems and non-Unix file systems should be supported at the same time**. Different disk partitions may contain different types of file systems, but they should be mountable on each other to form a single directory tree.
  - Files belonging to different file systems should be **easily sharable** over a network.

# Relationship between the *vfs* and *vnode* structures




1. The rootvfs points to the vfs root file system
2. The vfs_next pointers create a linked list of mounted filesystems
3. The vfs_mntd points to the vnode representing the root in the filesystem
4. The vfs_mntdover points to the vnode of the directory the file system is mount over.

# Integrated File System Structure



# 3.3 Single-Level Store

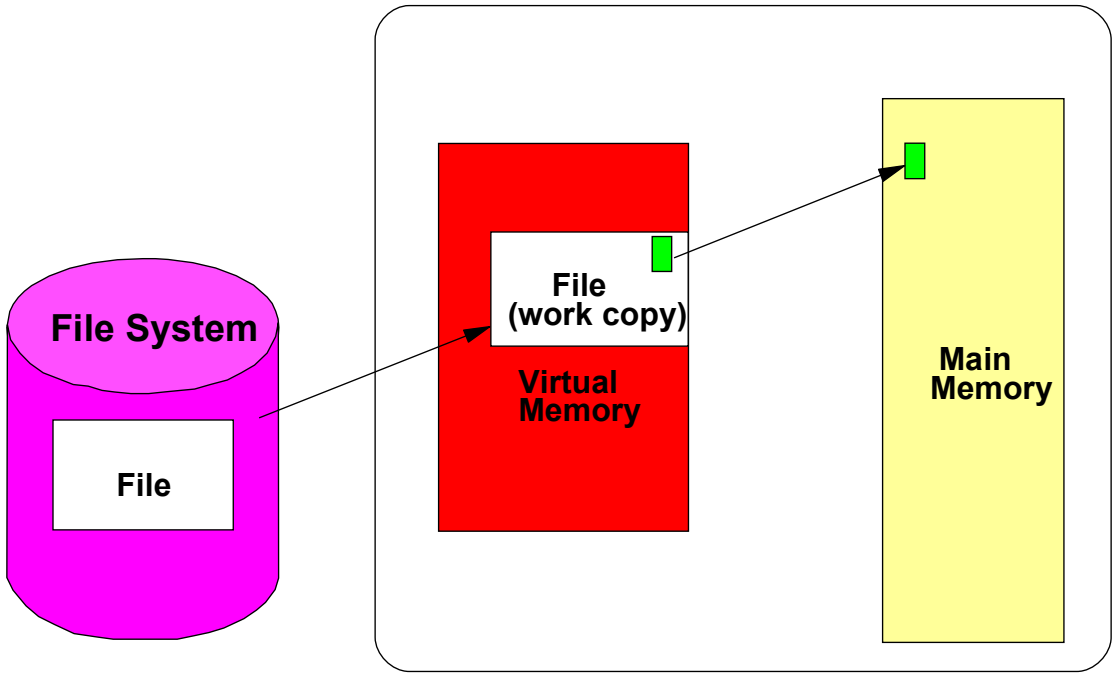
… is not about large address space; it‘s about sharing.

# Evolution of Virtual Memory

- Programs need more main memory than available
  - Overlay structure of programs
  - Programmer as memory manager?
- Multi-user environments
  - Many pieces of programs must exist in memory at the same time
  - Time-sharing: processor should be busy all the time even in the case that one program is waiting for an I/O

# Conventional Approach

- Non-shared address space model
  - Each user gets a separate address space
- a file system outside of virtual memory (two-level store)
  - Anything have to be moved into virtual memory before it can be used or changed



# Disadvantage of this Approach

On a task switch, it takes a processor hundreds or even a thousand instructions to put away the address space of one user and load up the address space of a new user.

# An Observation

- For heavy interactive workloads:
  - System i switch to a new task after executing an average of about 1200 instructions

But, if nearly 1000 intructions are needed during a task switch to load the new address space, how System i get any work done?

"… A system has been devised to make the core drum combination appear to the programmer as a single-level store, the requisite transfers taking place automatically." *

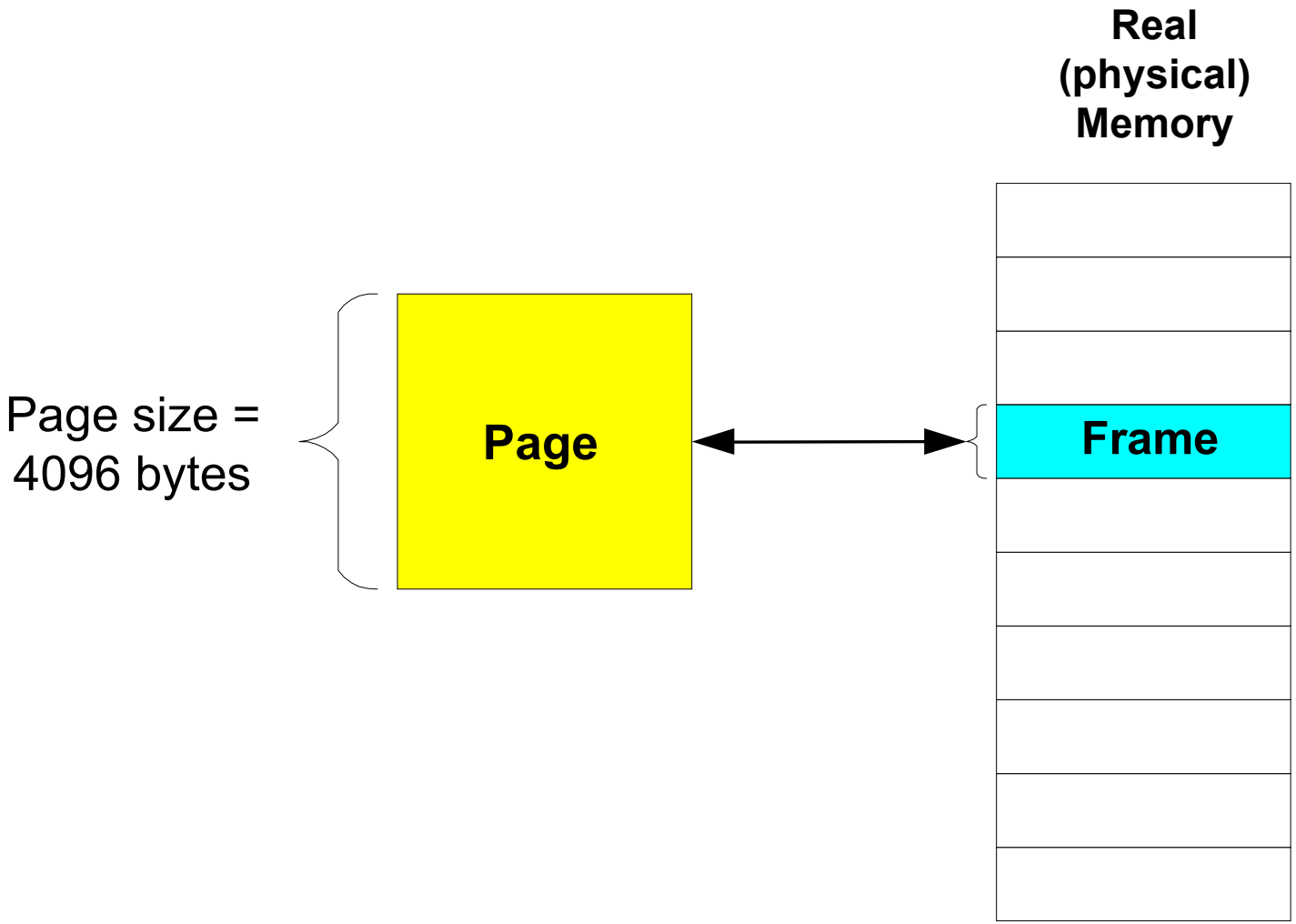
* Source: Kilborn, T.D., at al. „One-Level Storage System", IRE Transactions on Electronic Computers, April 1962

# Alternative Approach

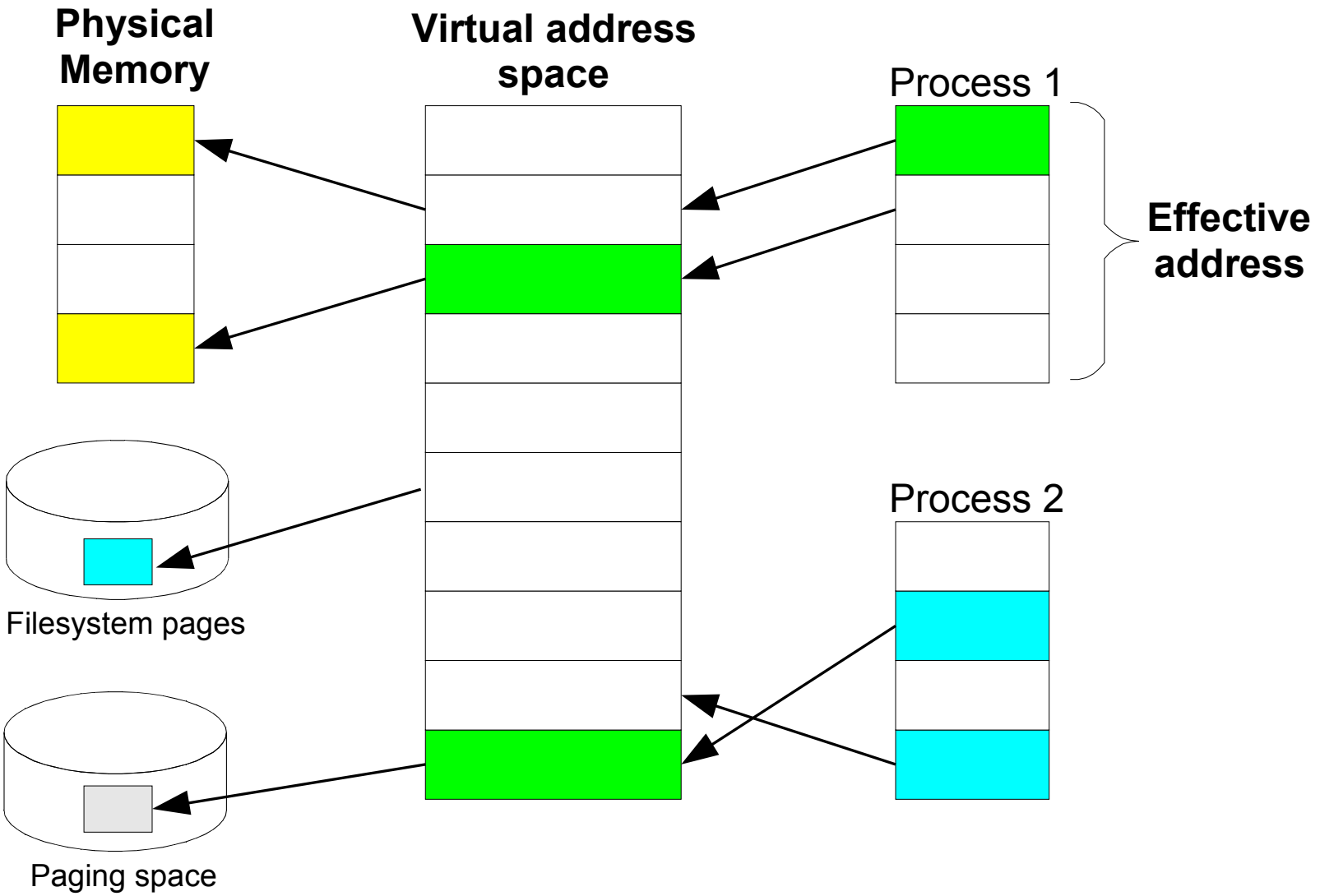
- Single-level store
  - Entire file system is part of virtual memory
    - Memory is now cache for all the disk storage
  - Requires a large address space
    - An address has to cover all the disk space storage

# 3.3.1 The Address Space

# Pages and Frames



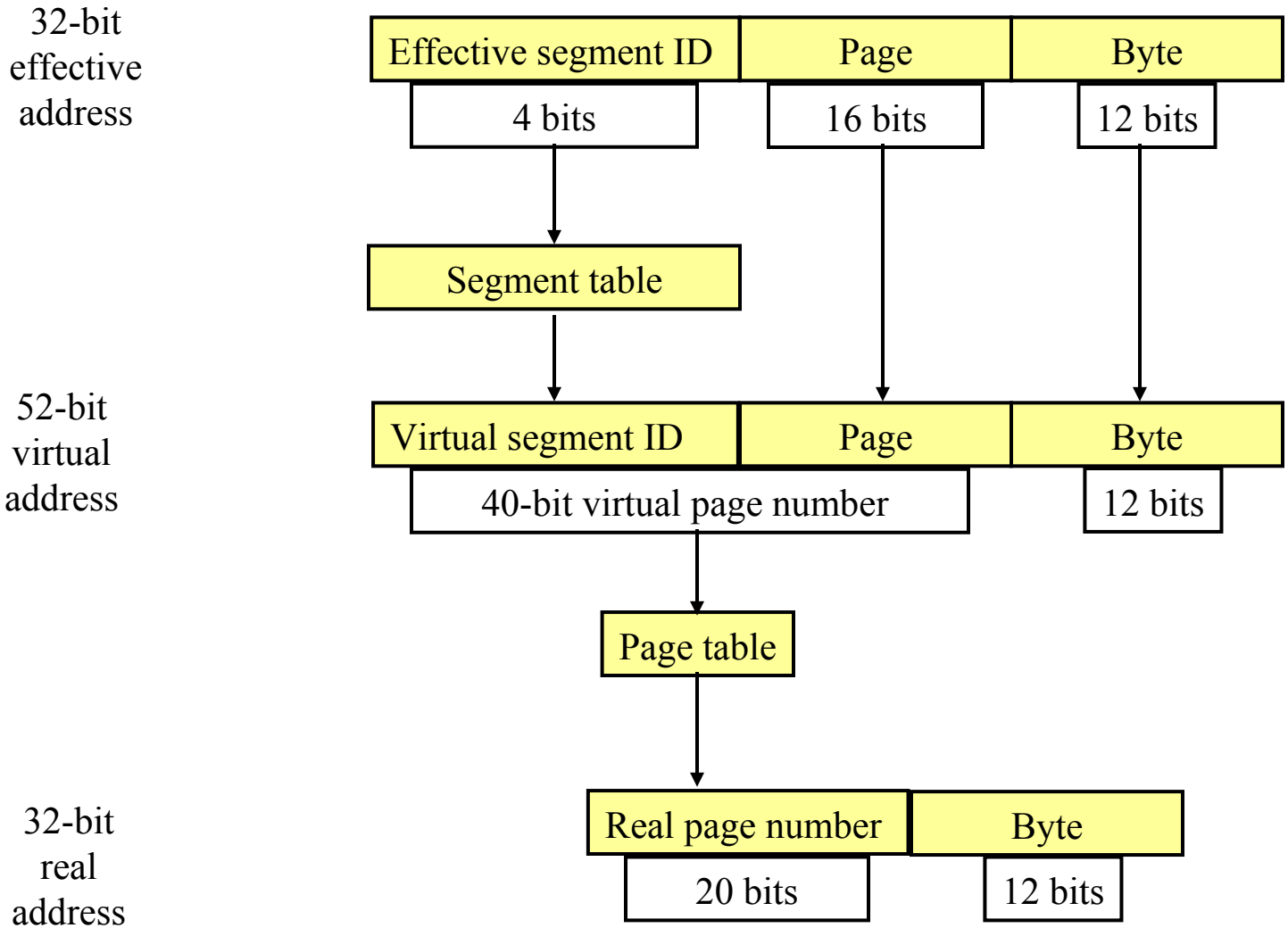
# Address Space of Conventional Systems



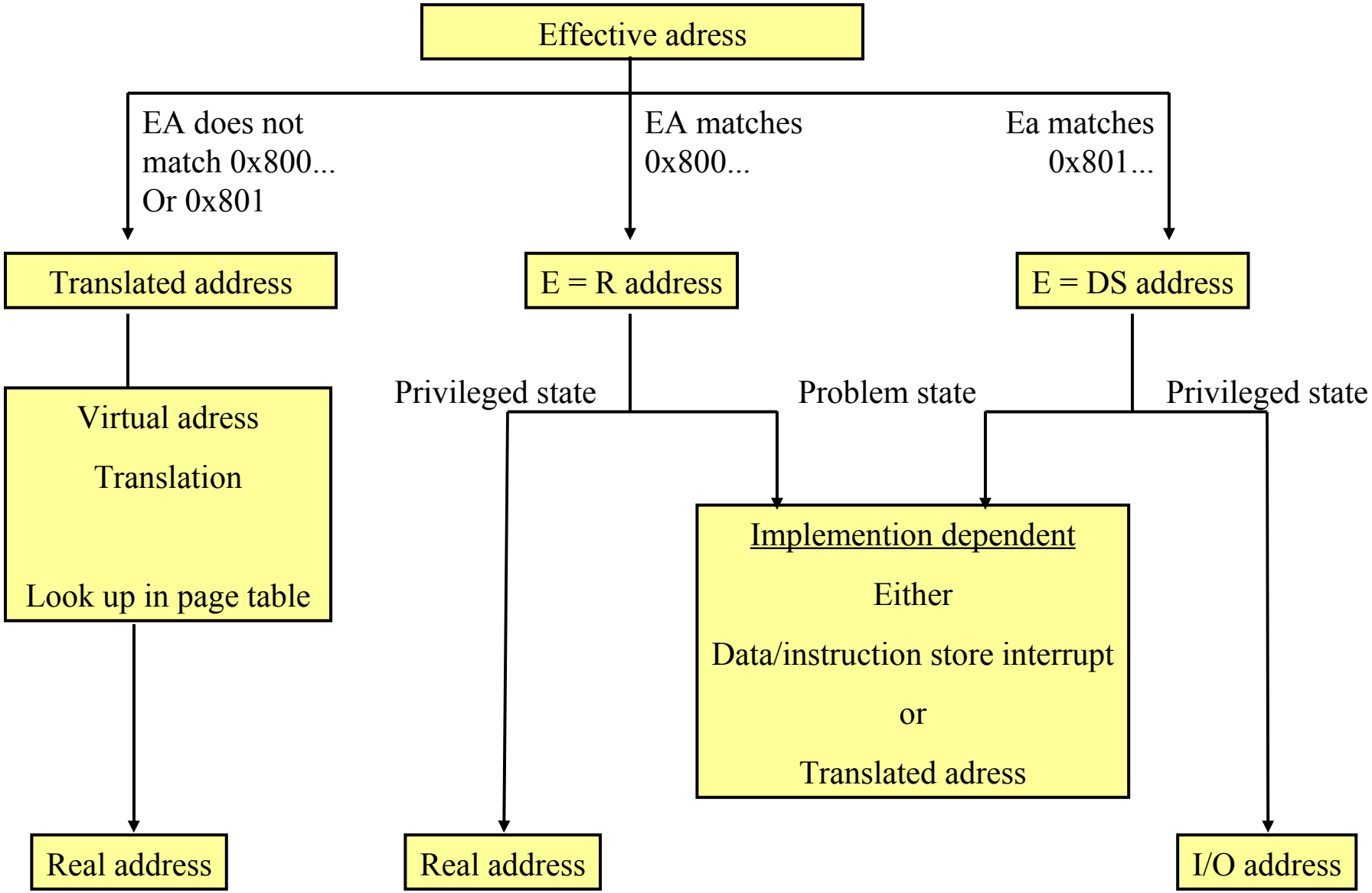
# Translating Addresses

| Step | Action |
|---|---|
| 1 | An effective address is referenced by a process or by the kernel. |
| 2 | The hardware translates the address into a system wide virtual address. |
| 3 | The page containing the virtual address is located in physical memory or on disk. |
| 4 | If the page is currently located on disk, a free frame is found in physical memory, and the page is loaded into this frame. |
| 5 | The memory operation requested by the process or kernel is completed on the physical memory. |

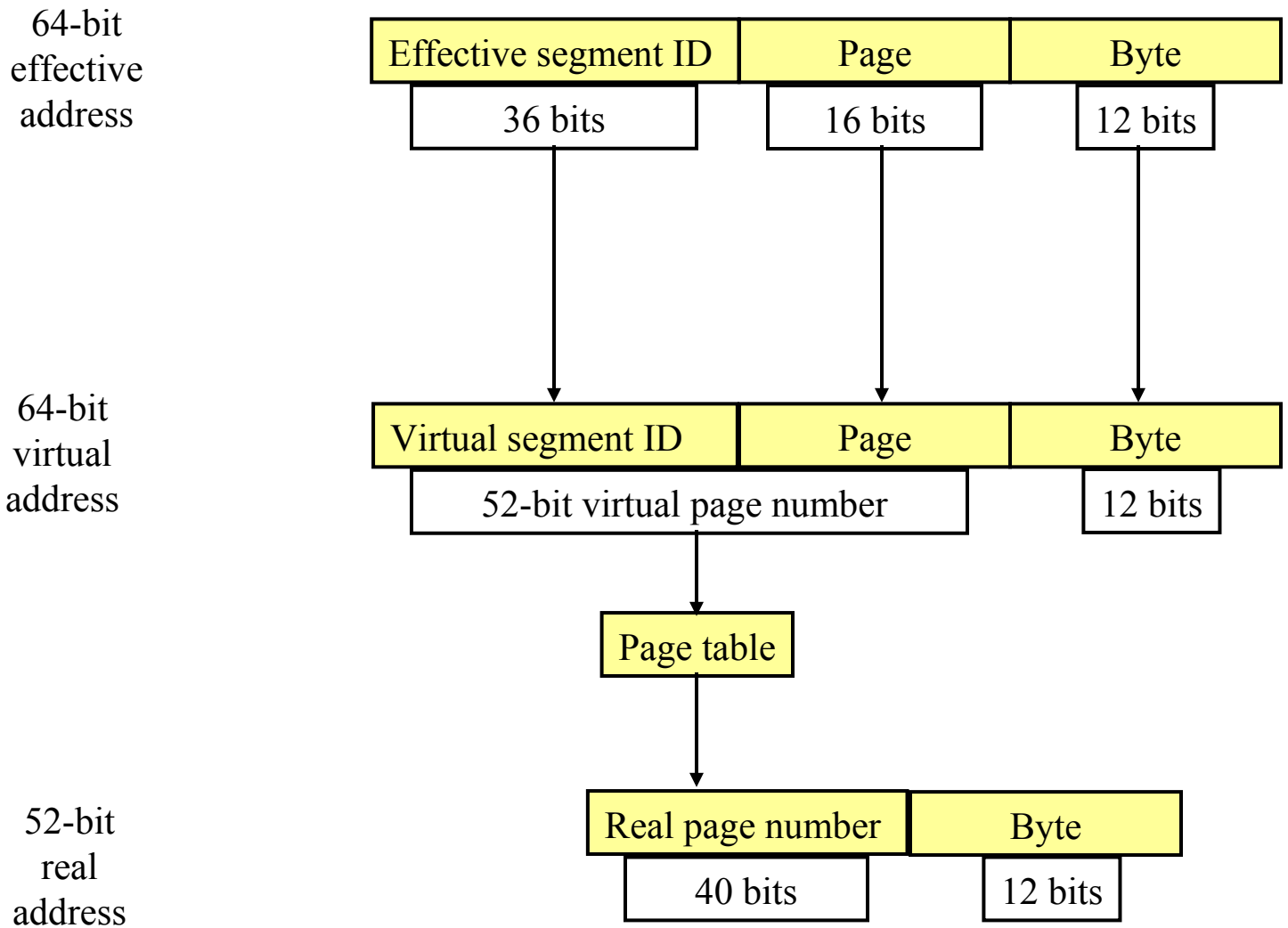
# Segment-Relative Addressing



# Address Translation



# Steps Involved in Address Translation

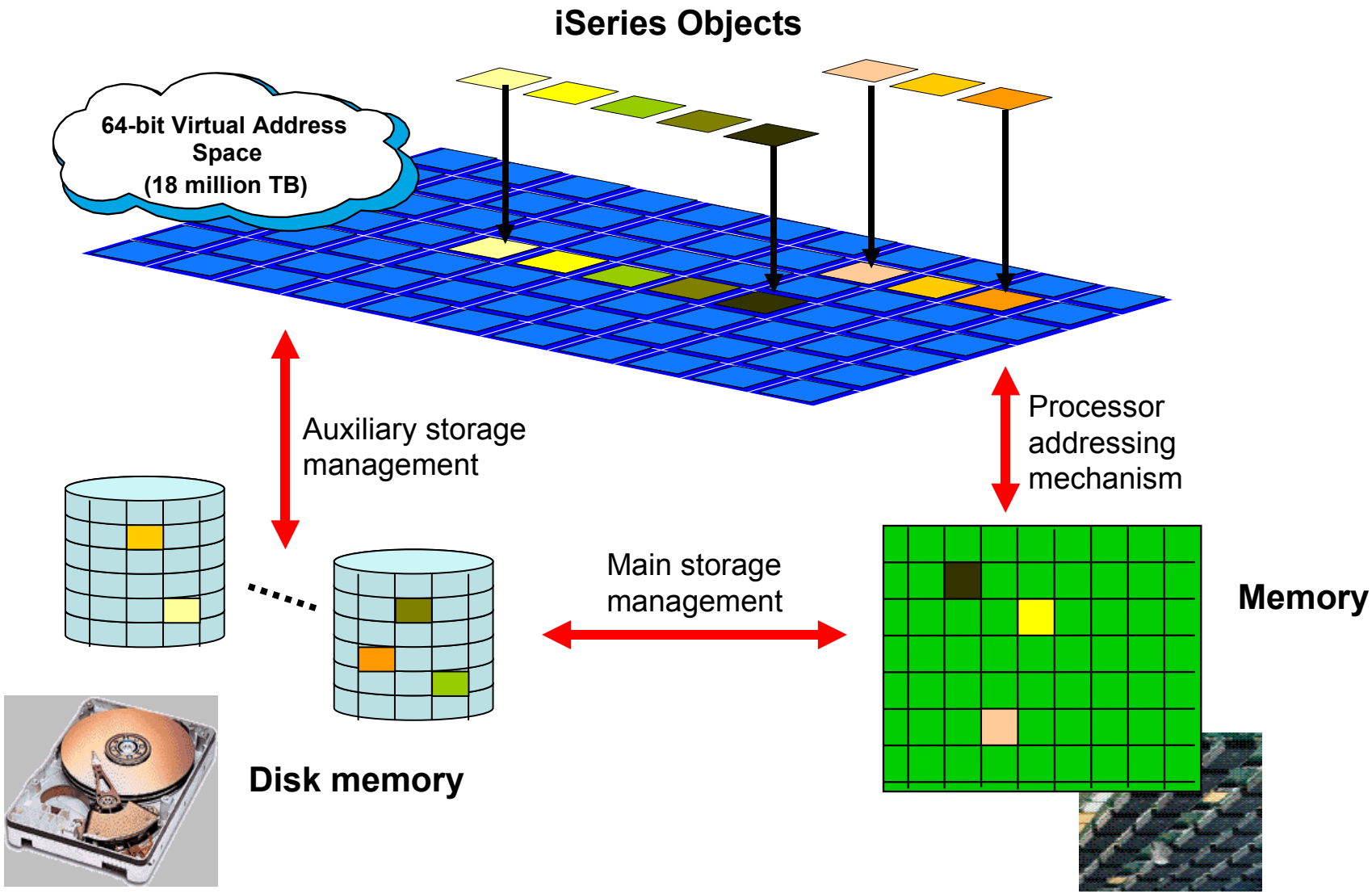


# iSeries Memory Model Characteristic

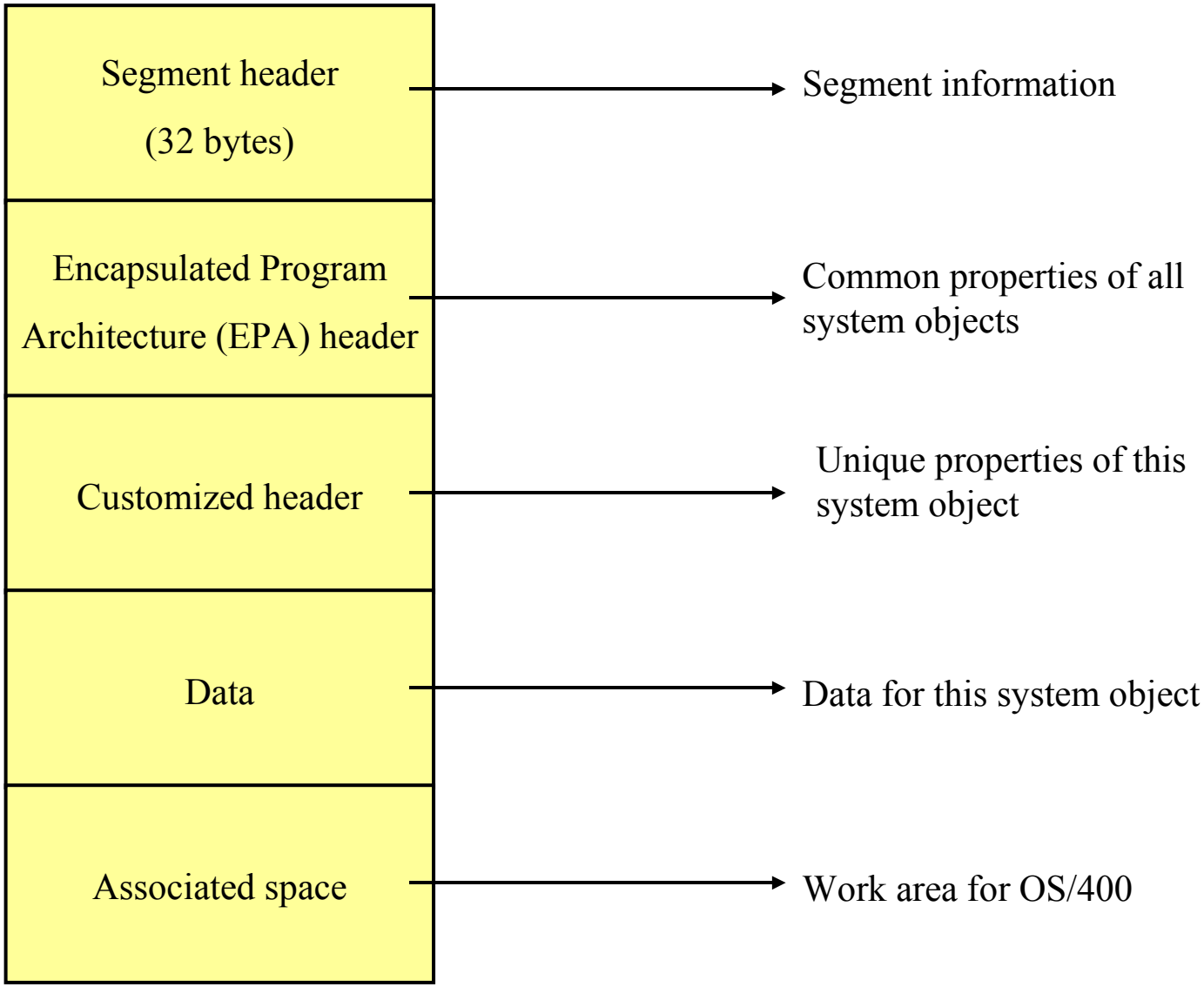
- The page size is $2^{12}$ bytes (4 KB)
- The effective address range is $2^{64}$ bytes
  - The effective segment size is $2^{24}$ bytes (16 MB)
  - The number of effective segments is $2^{40}$
- There are two special types of effective addresses
  - 0x800 Effective = Real addresses map all real memory
    (–> The real address range is $2^{52}$ bit)
  - 0x801 Effective = Direct-Store addresses map I/O space
- The virtual address range is $2^{64}$ bytes
  - The number of virtual segments is $2^{40}$-$2^{29}$
  - The size of virtual segments is $2^{24}$ bytes (16 MB)

**No need for effective-to-virtual translation!**

# Objects in the Single Level Store



# System Object Structure



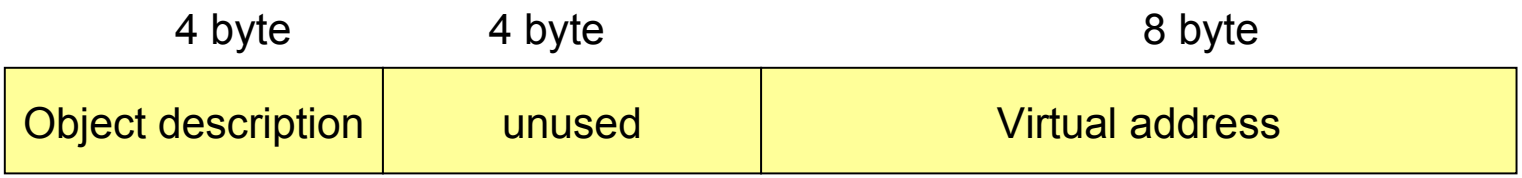
# 3.3.2 Pointer Protection

Aside from performance, the biggest advantage of a single-level store is that everything can be shared; it‘s biggest disadvantage is that everything can be shared.

# Tag-bit

- Pointers should not be modified by MI programs
  - MI programs can use object names and should not modify resolved pointers containing a virtual address of an object
- Usage of a memory protection bit (tag-bit)
  - only privileged instructions can modify the contents of a pointer
  - Tag-bits are used to detect modifications

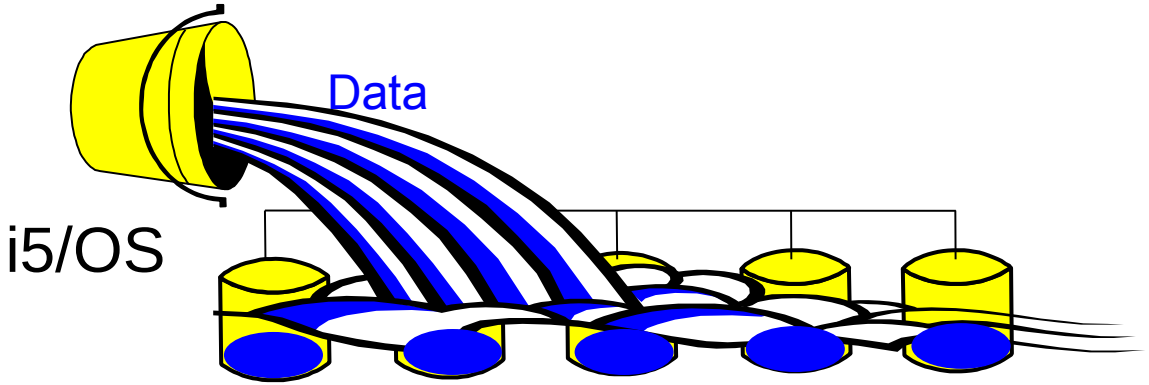
# MI Pointer in Memory




- Representation in memory:
  - 2 64-bit words
  - 8 ECC*-bits/word
  - one tag-bit/word
- Every write to memory means also
  - create and store the ECC bits
  - turns off the tag-bits
- Only privileged instructions can set the tag-bits
  - MI translator does not generate tag instructions

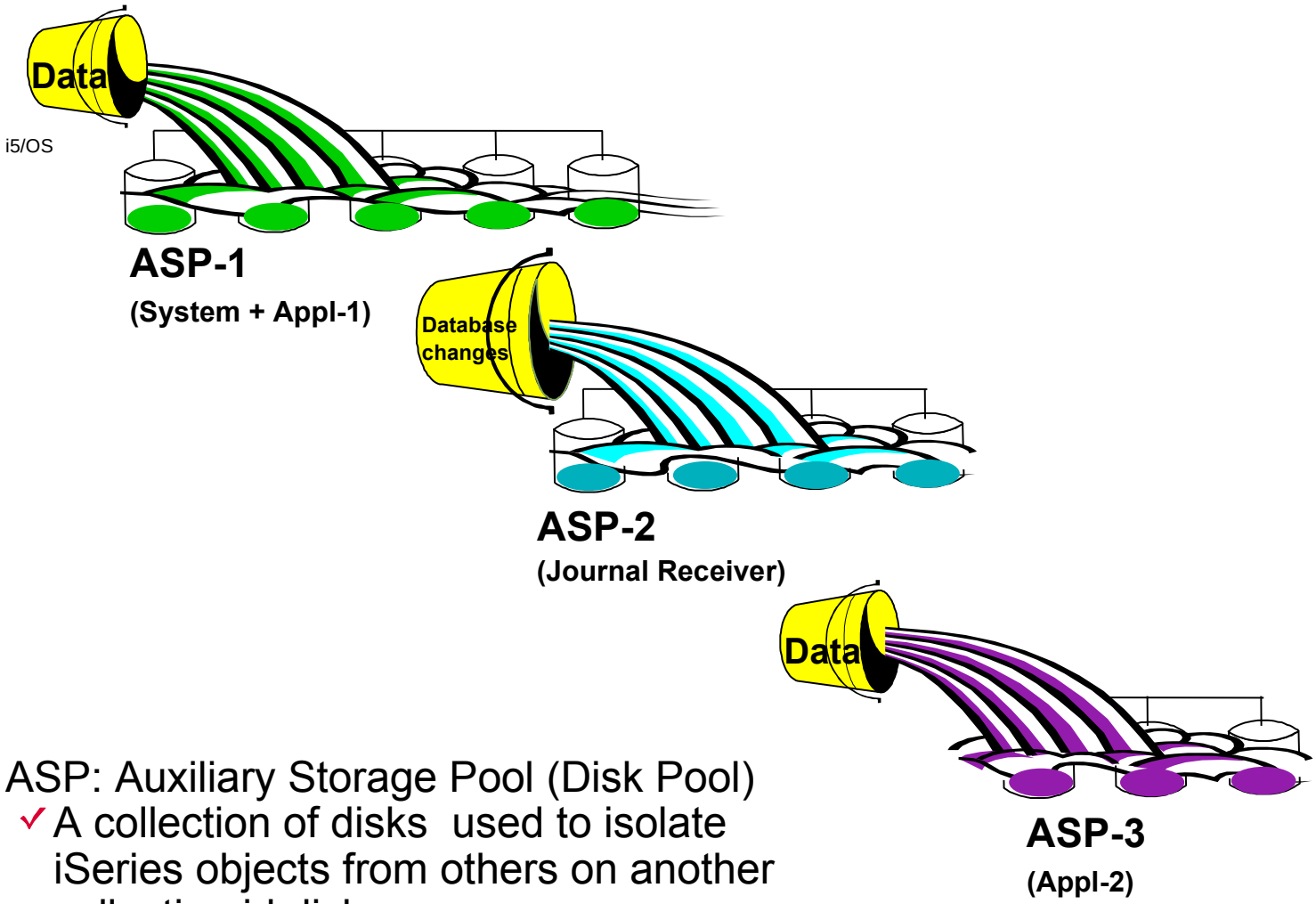
* Error Correction Code

# Single-level Store and Disk Management




- All data is spread across available disk arms
  - Optimum performance - automatically
- Not all information is necessarily contiguous (1MB)
  - Improved performance
    - Balanced disk arm utilization
- Optional rebalancing
  - space/arm utilization
- Minimal Database Administration
  - Information accessed by name not hardware address

# Auxiliary Storage Pool (ASP)

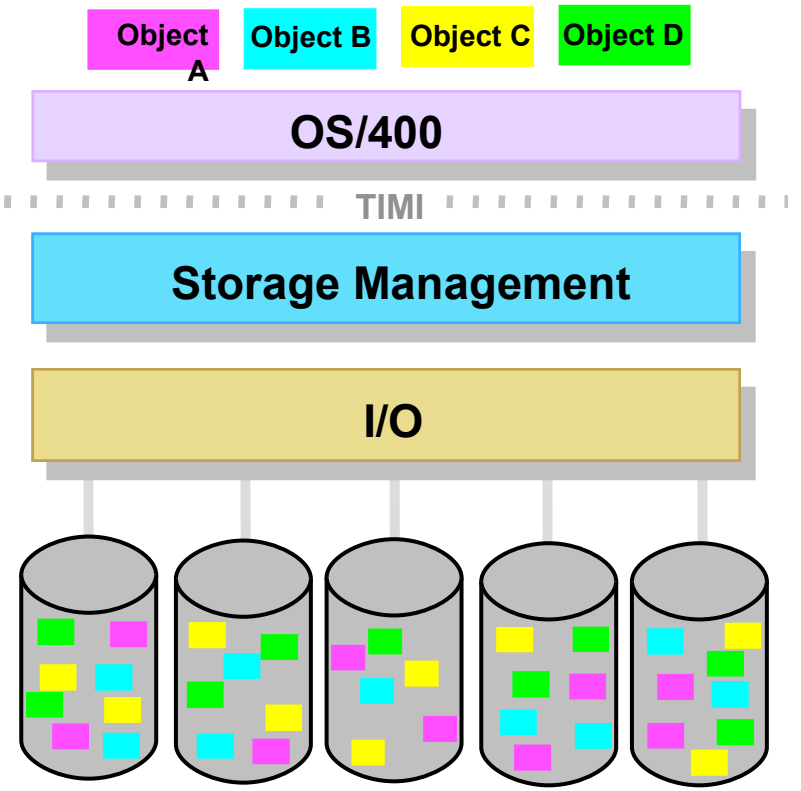



- ASP: Auxiliary Storage Pool (Disk Pool)
  - ✓ A collection of disks used to isolate iSeries objects from others on another collection id disks.

# iSeries Storage Architecture

- Data is scattered across all disks in a disk pool
- Good performance due to Parallel I/O
- Disks fill evenly
  - No manual data placement
  - No individual "disk full" conditions to handle
- Newly added disk capacity is utilized automatically
- No continuous disk performance monitoring
- Automatic disk operations eliminating DBA needs

## Single Level Store



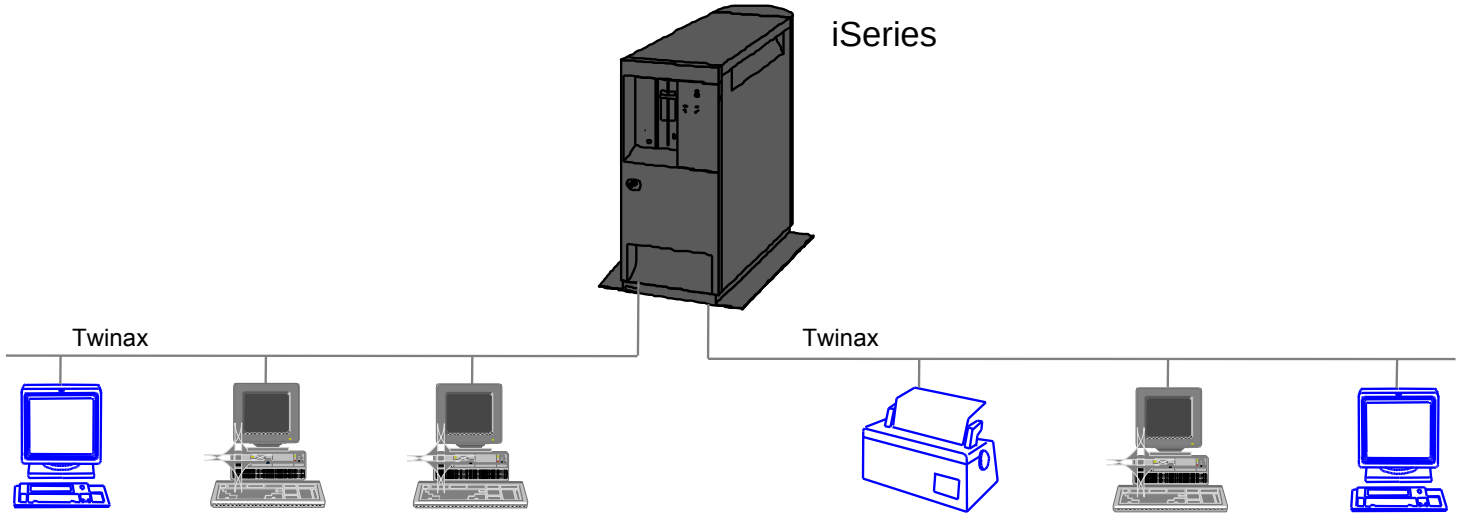
# 3.4 Hardware Integration

# I/O Requirements

- I/O performance depends on workload
  - long data streams (e.g. multimedia)
    - response time depends on I/O bandwidth (MB/s)
  - small I/O‘s (e.g. order entry application)
    - reponse time depends on latency of I/O devices
- Commercial environment:
  - Response time isn‘t the only performance measurement
  - High throughput is also required
    - How many tasks can be processed in a unit of time?
    - How much data can be moved in a certain time?
  - Scalability, Reliability, Costs ($/GB), ...

# 3.4.1 Hierachy of Micro Processors

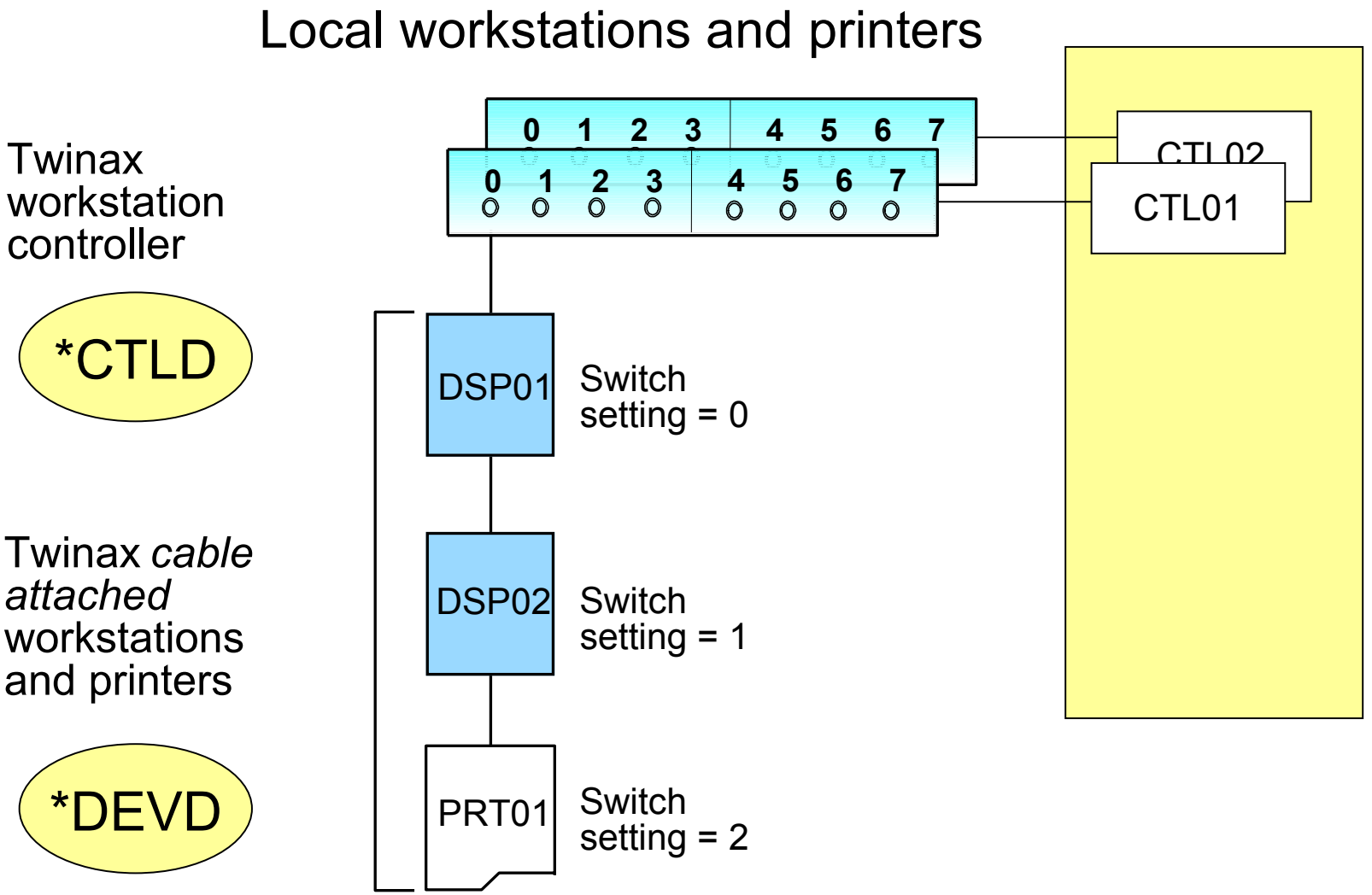
# Local Attached Devices




iSeries Terminals are connected via Twinax cabling topology to a central system

- Display and printer devices
- PCs with Twinax adapter and emulation software
- Seven Twinax station addresses per iSeries 5250 workstation controller port
- Twinax cable, up to 1500 m, UTP cable shorter distance

# Locally Attached: Workstation Controller (Twinaxial)



# IDE vs. SCSI

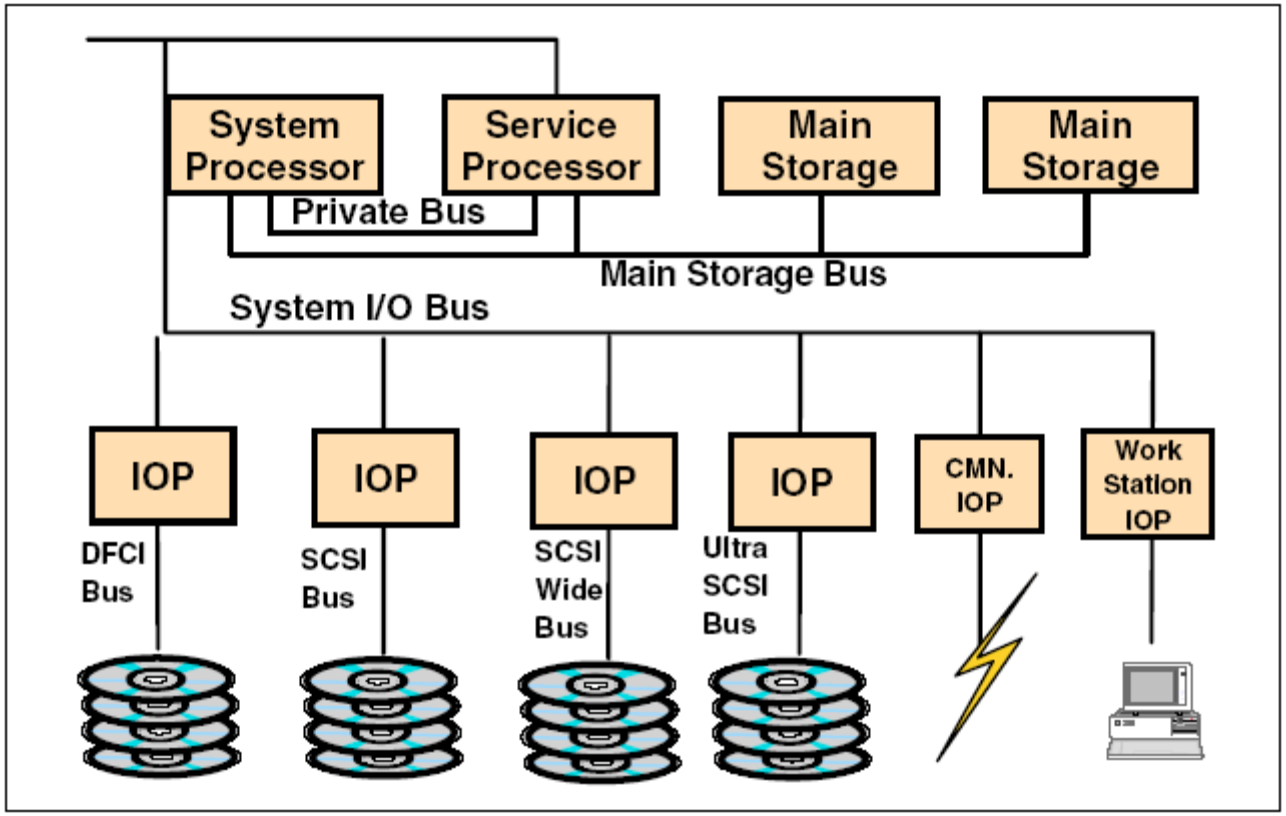
- **SCSI** stands for “Small Computer System Interface”
- host bus requires a SCSI host adapter which controls the data transfer on the SCSI bus
- SCSI interface generally includes the ability to handle multiple requests, up to a certain number
  - called **command queuing and reordering** or multiple command queuing
  - very useful for servers and other systems being used by multiple people

- **IDE** stands for “Integrated Drive Electronics”
  - an IDE interface is generally limited to a single transaction
  - is generally not a problem for most single-user PC’s

# Special IOP‘s Could Improve Performance

1. Main processor encounters a data request (e.g. read from disk)
2. Delegates the request to IOP
3. Main processor is working on other programs
4. Returns if data becomes available

# Hierachy of Micro Processors



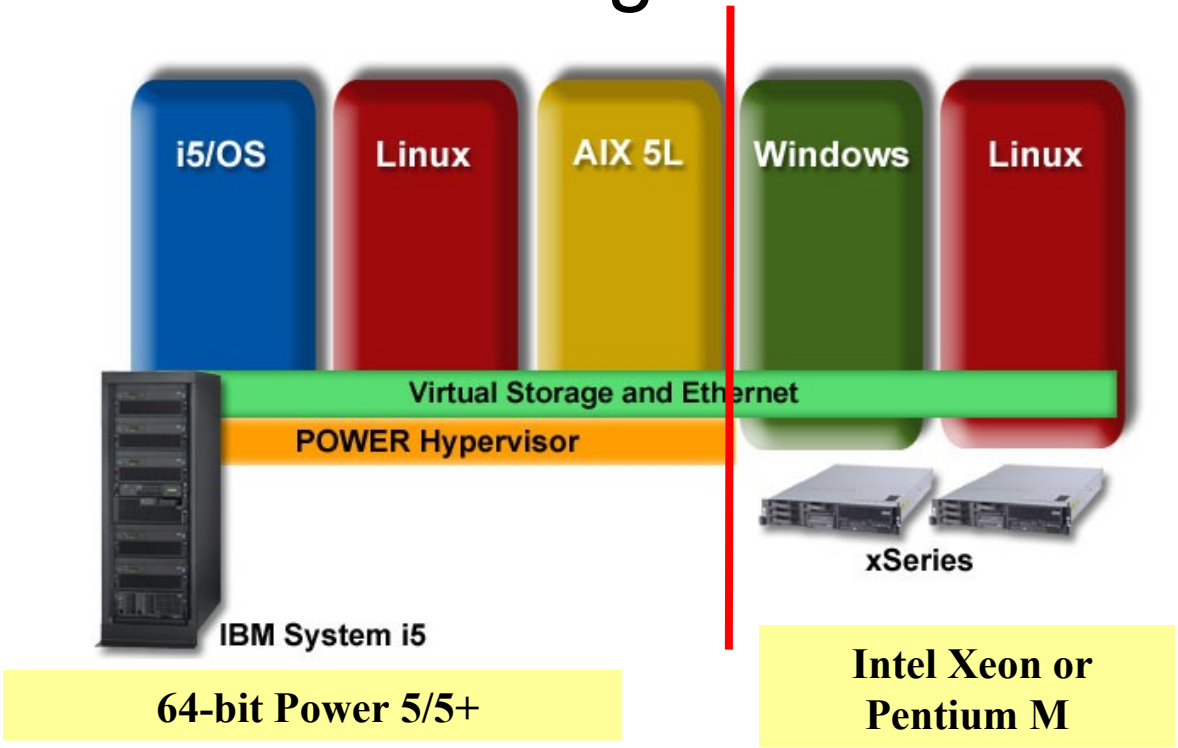

Advantages:

- Performance
- I/O technology can easily used and updated any time without disrupting the rest of the systen

# CPW – Commercial Processing Workload

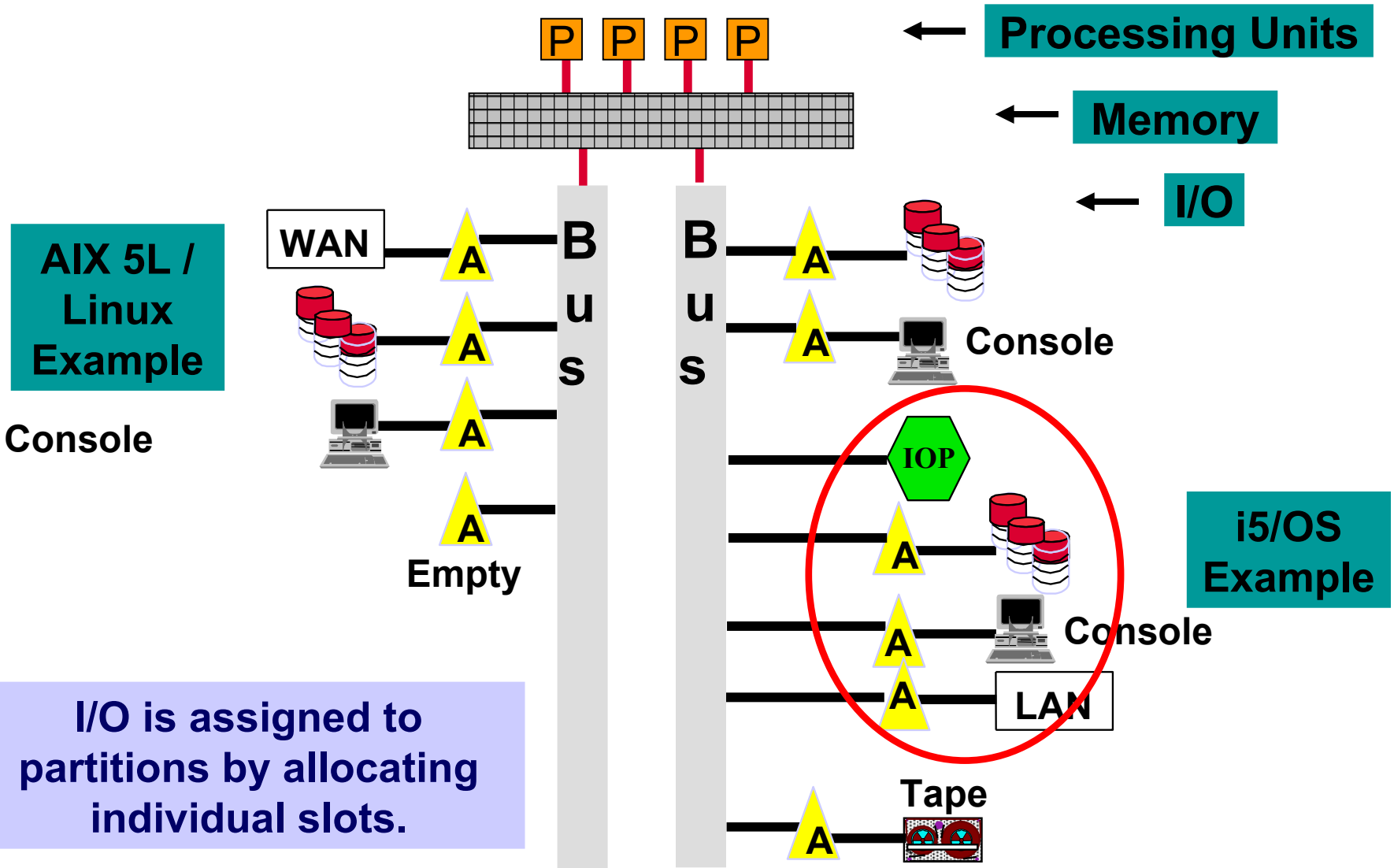
- CPW: Based on TPC-C
  - iSeries performance, because it is not a uni-processor, is not in direct relation to MHz. Rather, it is measured with a relative, commercial benchmark
- Users use four distinct interactive applications
- Complex Transactions
- Reads, updates, inserts, deletes, block inserts, index changes
- Journaling and commitment control
- Includes daytime batch

# Remember: i5 Heterogeneouse OS Support



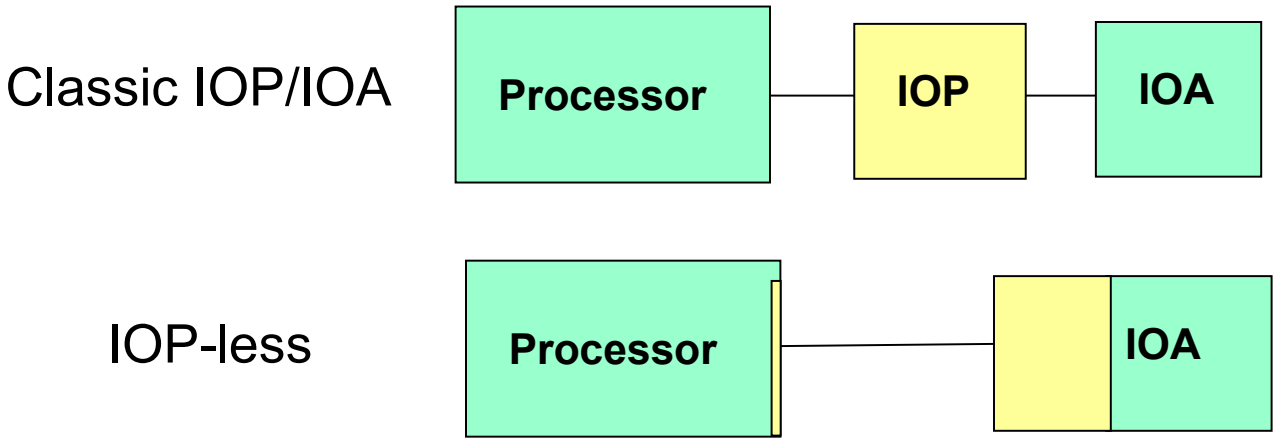

- Multiple logical systems within one physical systems enabling flexibility to run multiple application types
- Dynamically share resources
- Support heterogeneous environments

# I/O Resources in a LPAR Environment




**I/O is assigned to partitions by allocating individual slots.**

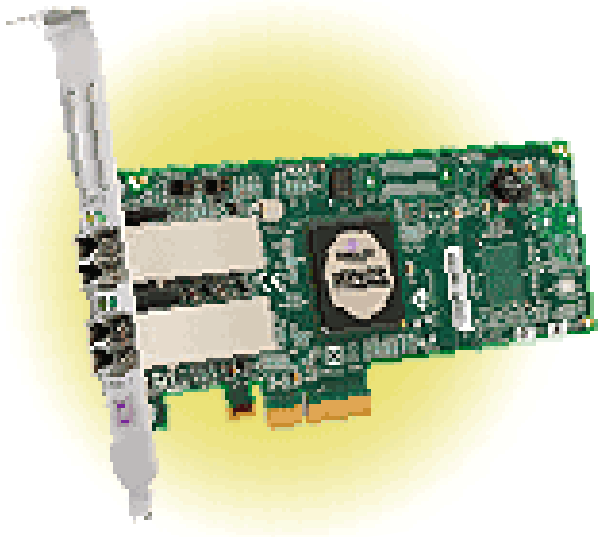
# IOP-less Direction



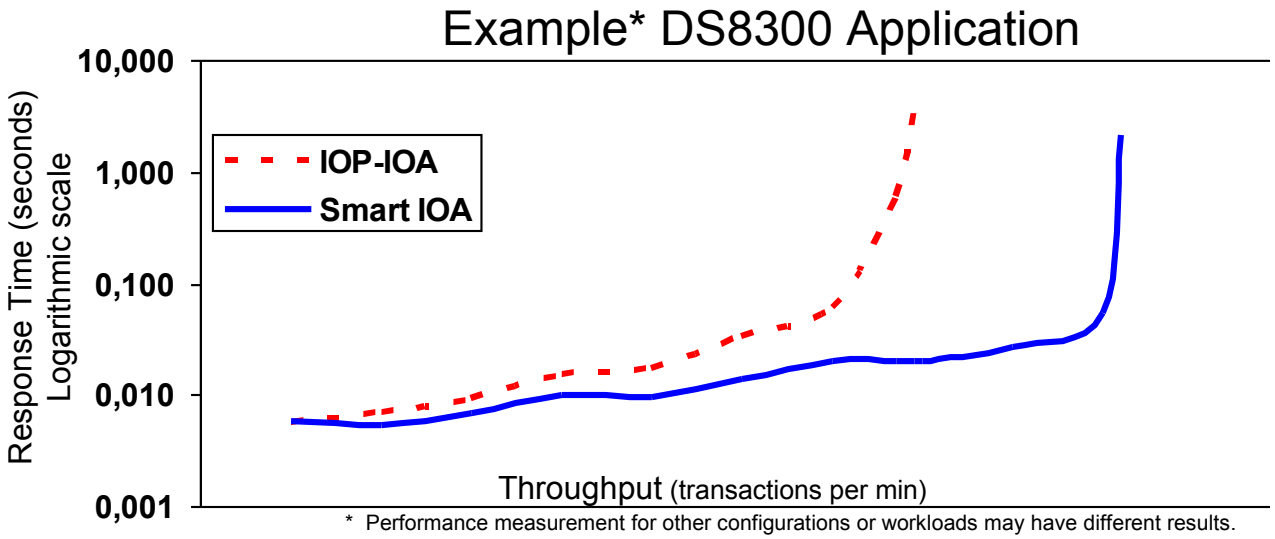

- IOA technology has evolved to the point where an IOA can take over the IOP as well as the IOA functionality.
- Benefits include
  - Avoiding cost of IOP and PCI slot to hold IOP
  - Configuration flexibility including simpler LPAR I/O
- POWER5 or later server required

# Example: Smart Fibre Channel Adapters

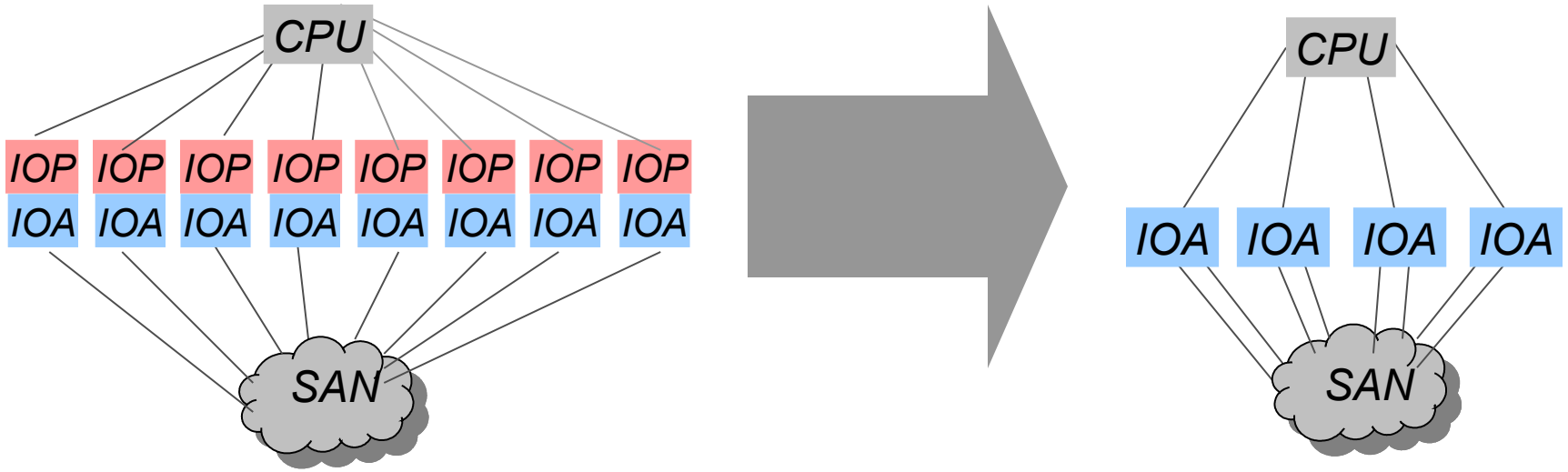
- Saves PCI slots
  - Smart IOA (no IOP)
  - Dual 4Gb ports
- Greatly enhanced SAN disk performance
- Enhanced flexibility
  - Dual ports
  - No IOP-IOA placement rules
  - Alternate tape IPL
- POWER6 & IBM i 6.1.(aka i5/OS V6R1) required



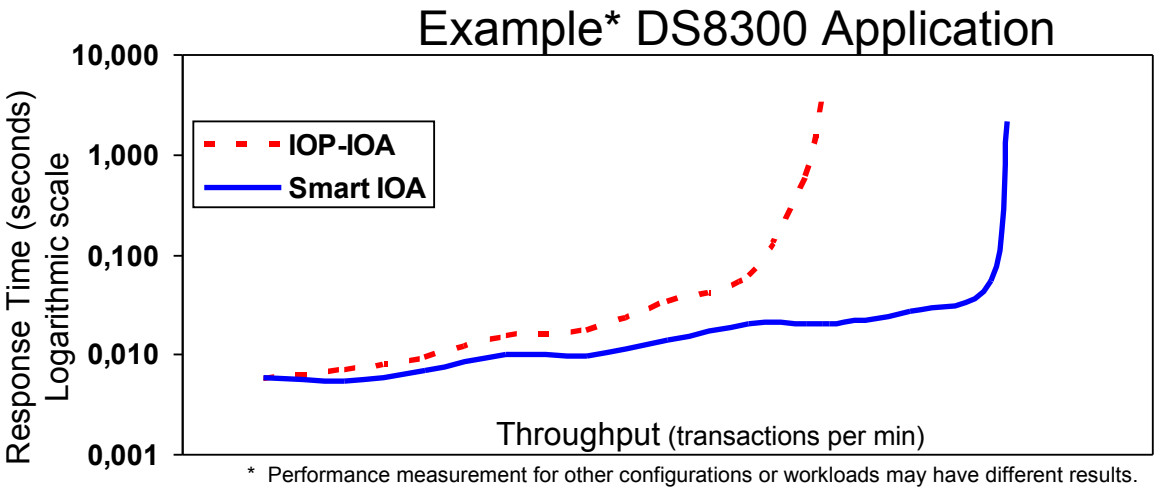
# Greatly Enhanced SAN Disk Performance for i5/OS

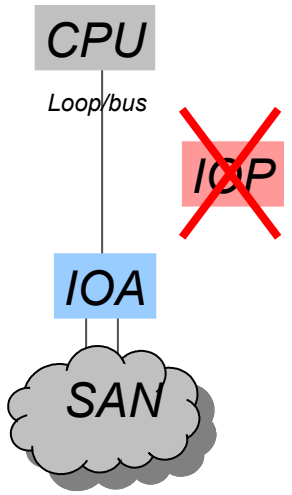



- Everything the same except the Fibre Channel cards.
- Eight IOP-IOA pairs replaced by 4 smart IOAs.
- Smart IOA performance wins.



# Greatly Enhanced SAN Disk Performance for i5/OS

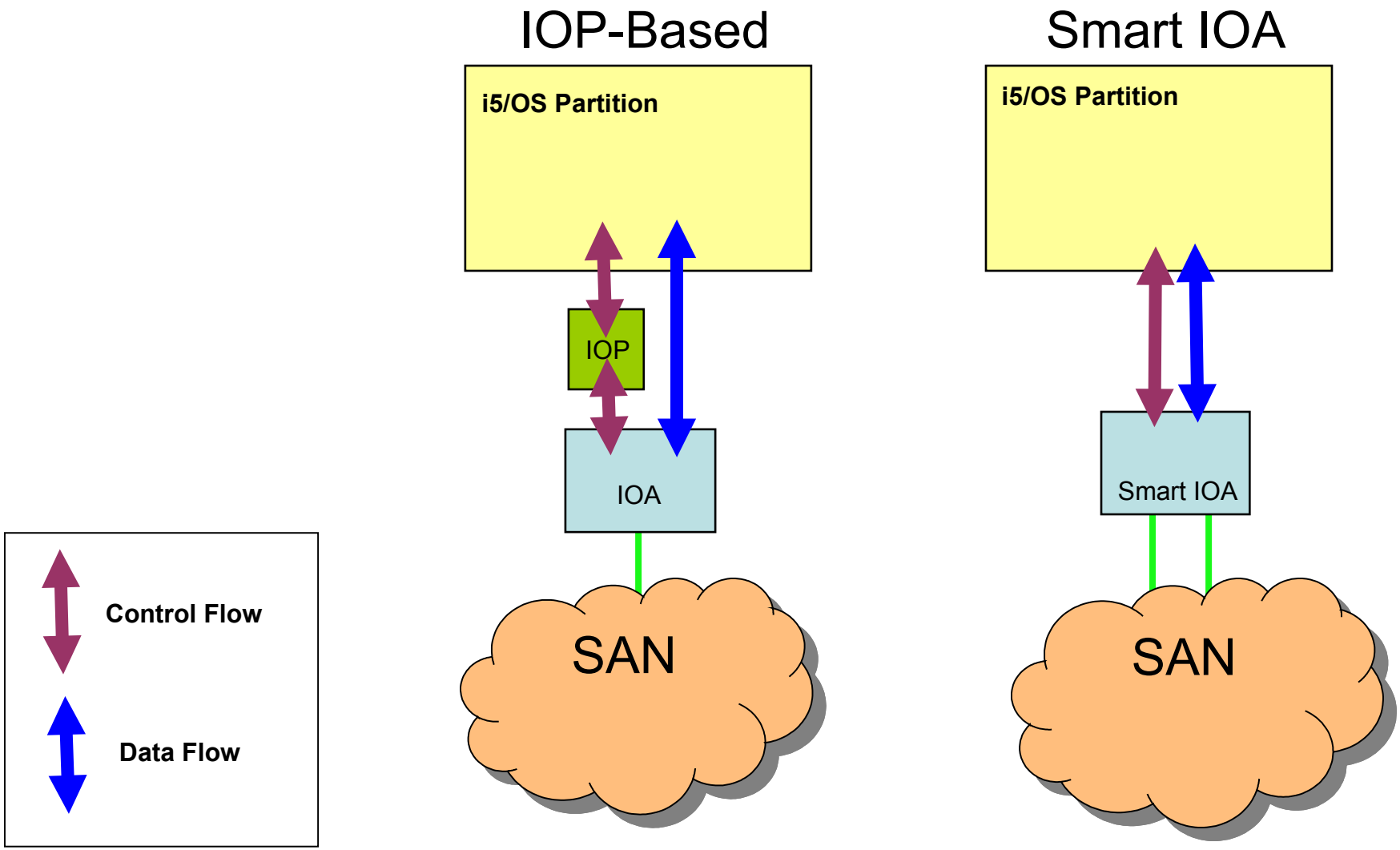


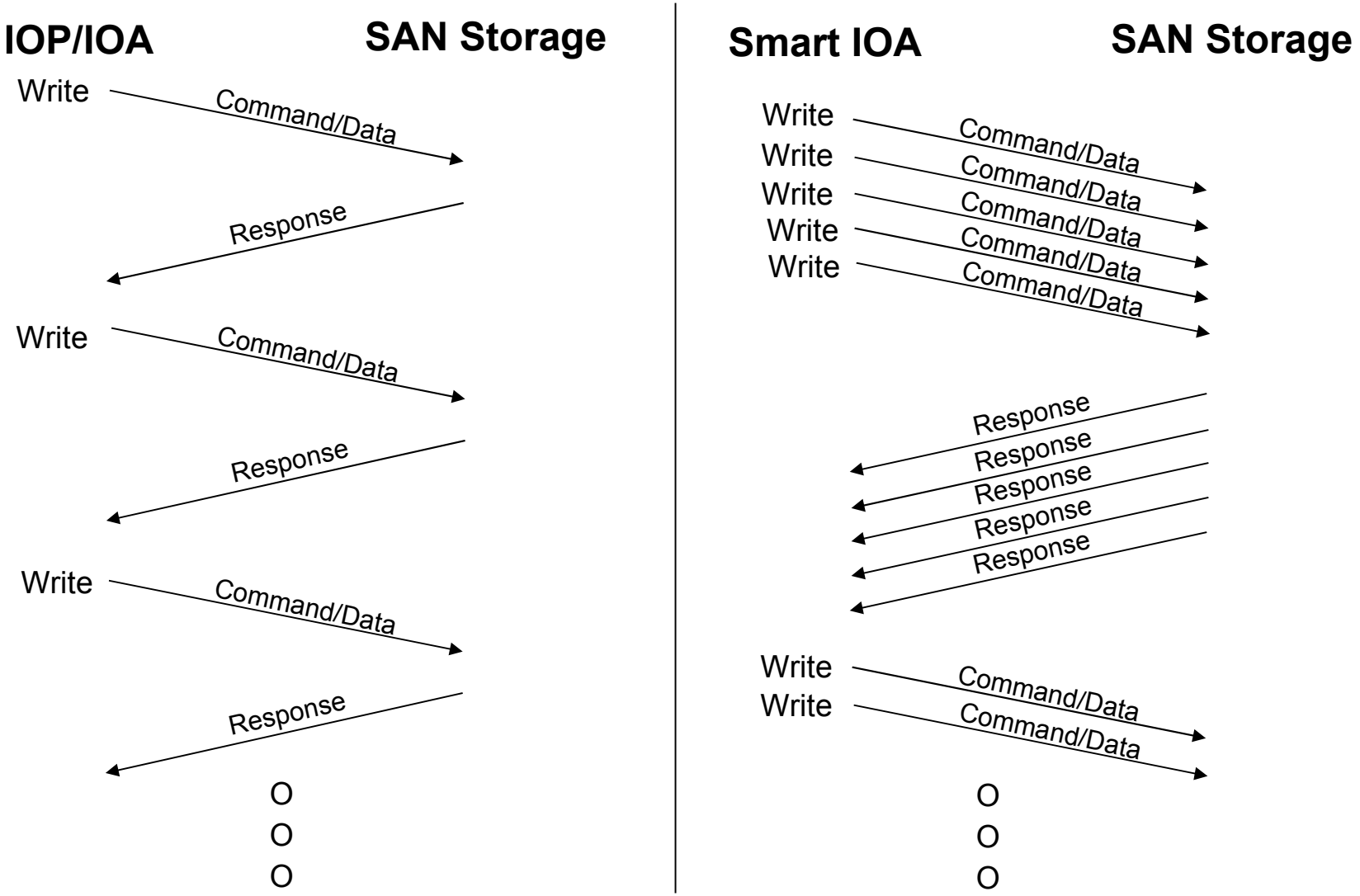

Key enhancements of the re-architected data/command paths

- Tagged Command Queuing
  - Multiple ops to each LUN-connection (more multi-threading), leveraging existing DS8000 efficiencies
- Header Strip/Merge Improvements
  - Moved function into IOA reducing traffic on HSL loop and PCI X bus, reducing latency
- No IOP in the command path
  - Leverage today's faster, more powerful IOA technology to eliminate one link in the chain

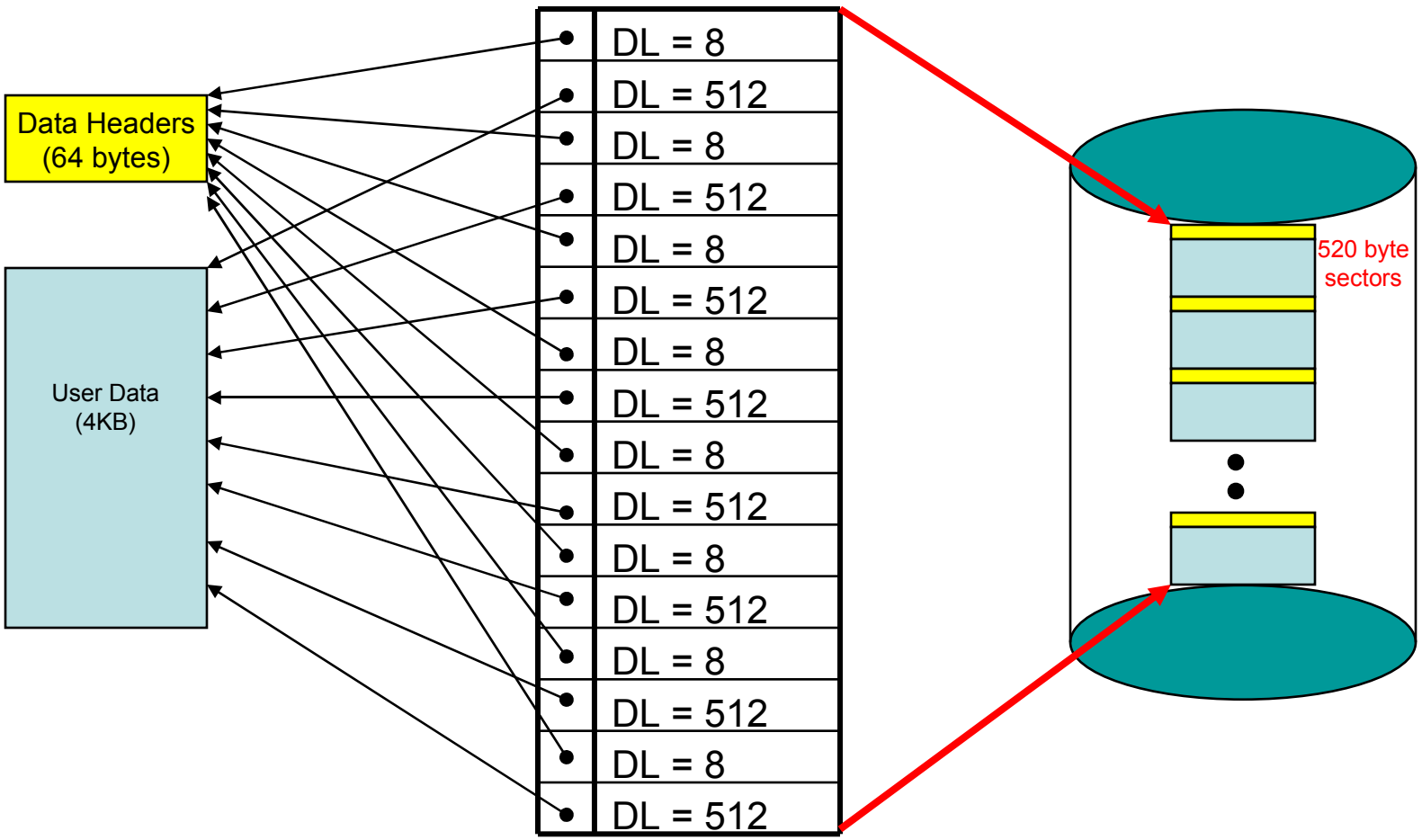
# Smart IOA Architecture - Data & Control Flow



# Smart IOA Architecture – Tagged Command Queuing



# Header Strip/Merge




SLI-2 BDL for 1 Page,
repeated for every page of the I/O

# Tagged Command Queuing

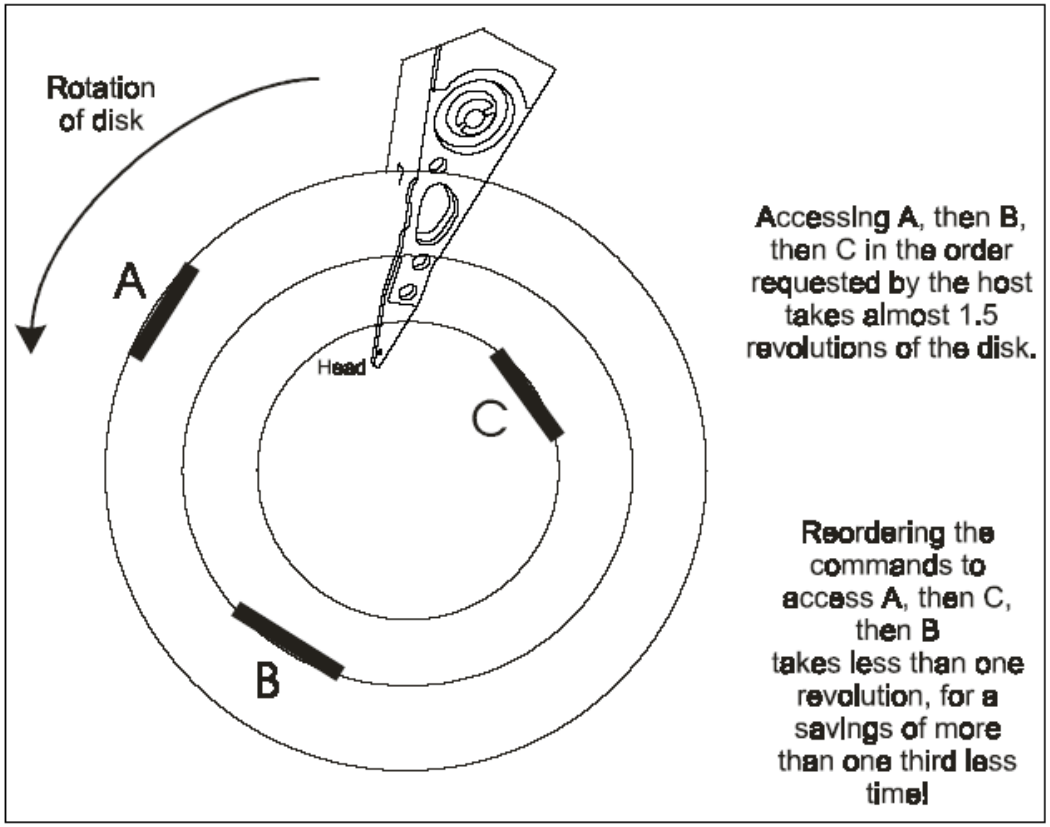


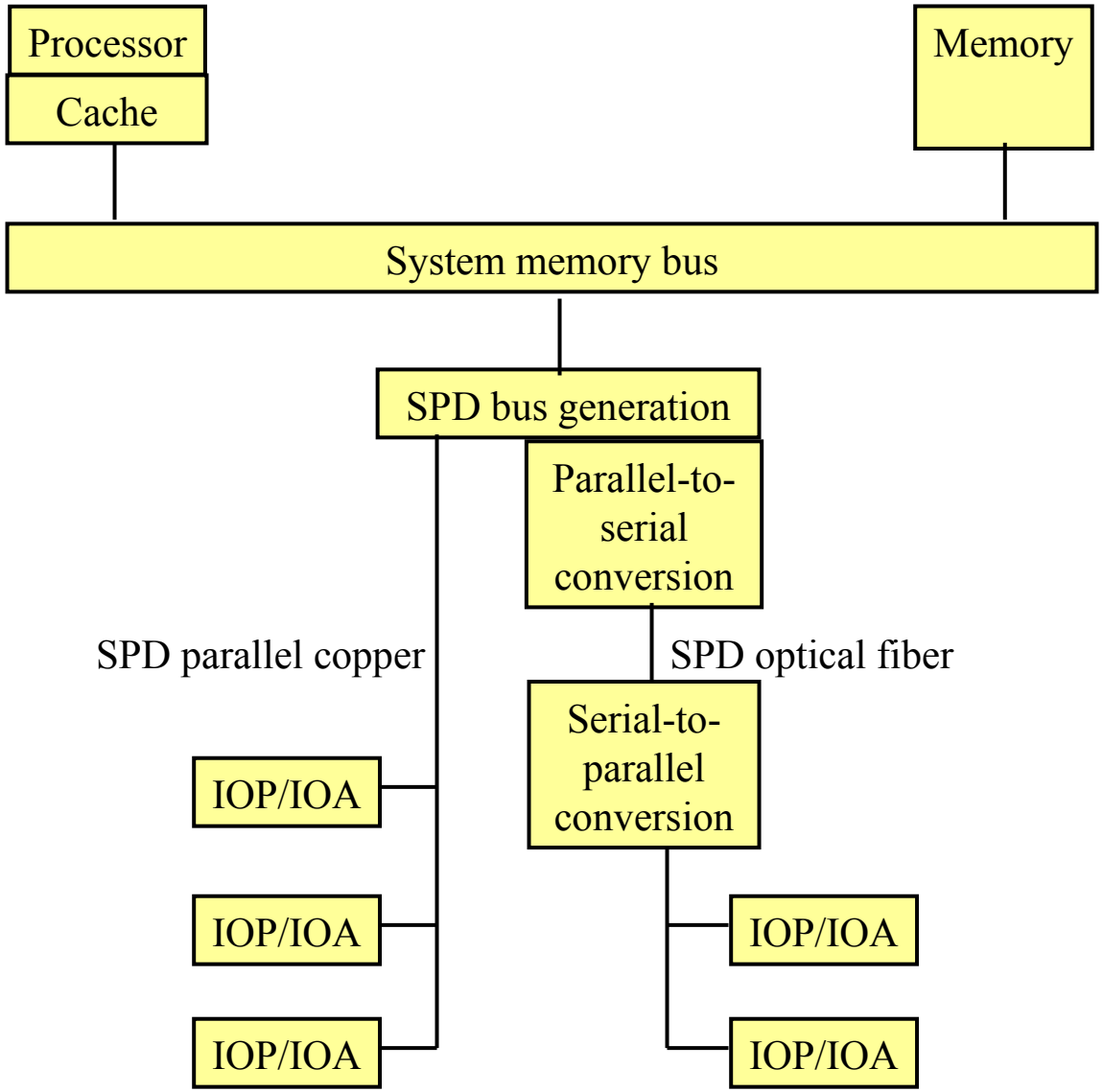

*Figure 1. Command Ordering in Tagged Command Queuing*

The adapter adds tags to the individual commands. The hard drive reorders the commands based on seek distance and rotation. The drive has dedicated buffers that will accept up to 32 commands at one time. The drive can then internally change the order of processing the commands to optimize the seeks ...

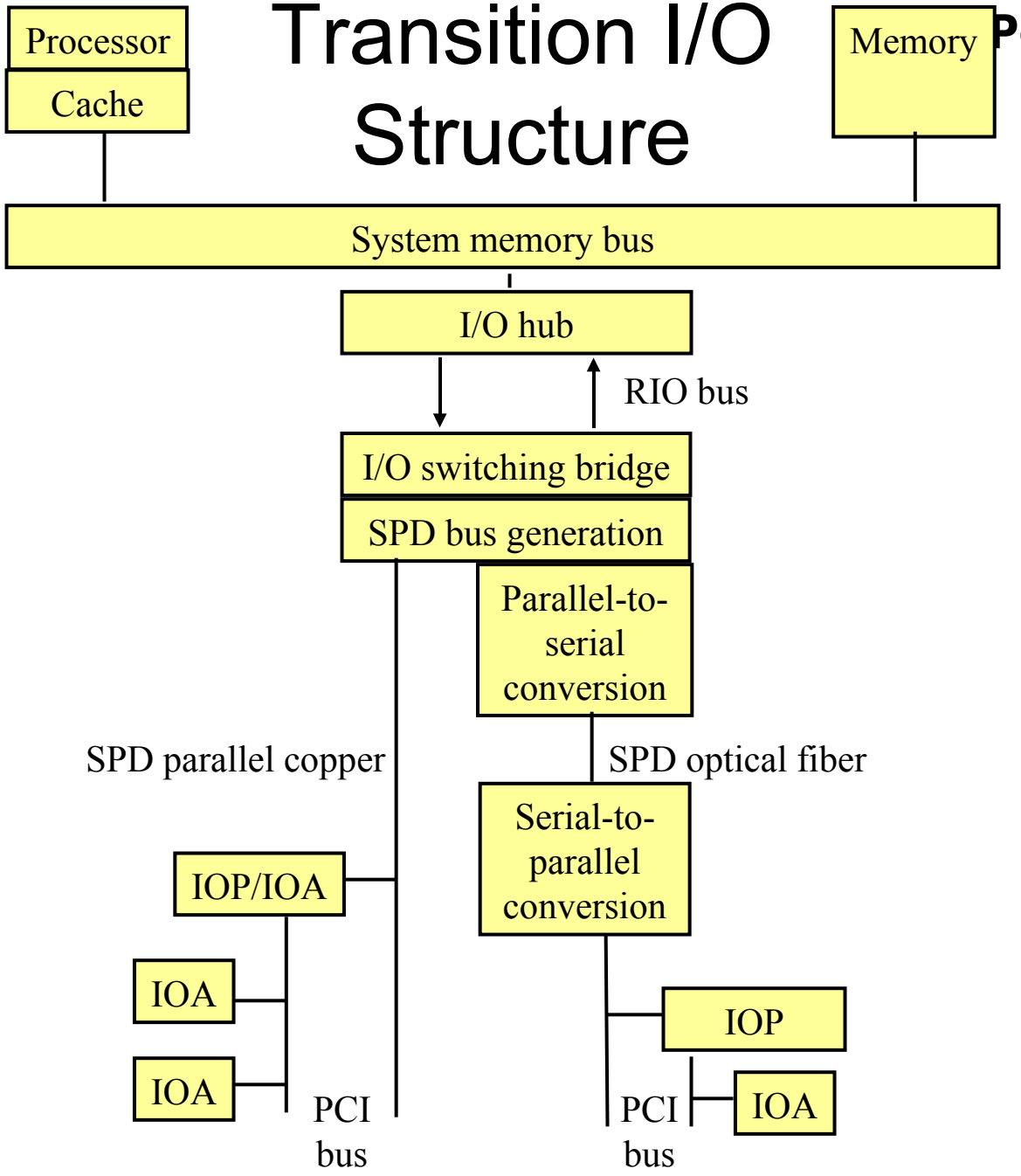
**Source: http://www.wdc.com/en/library/sata/2579-001076.pdf**

# 3.4.2 System i I/O Structure

# Original SPD I/O Structures



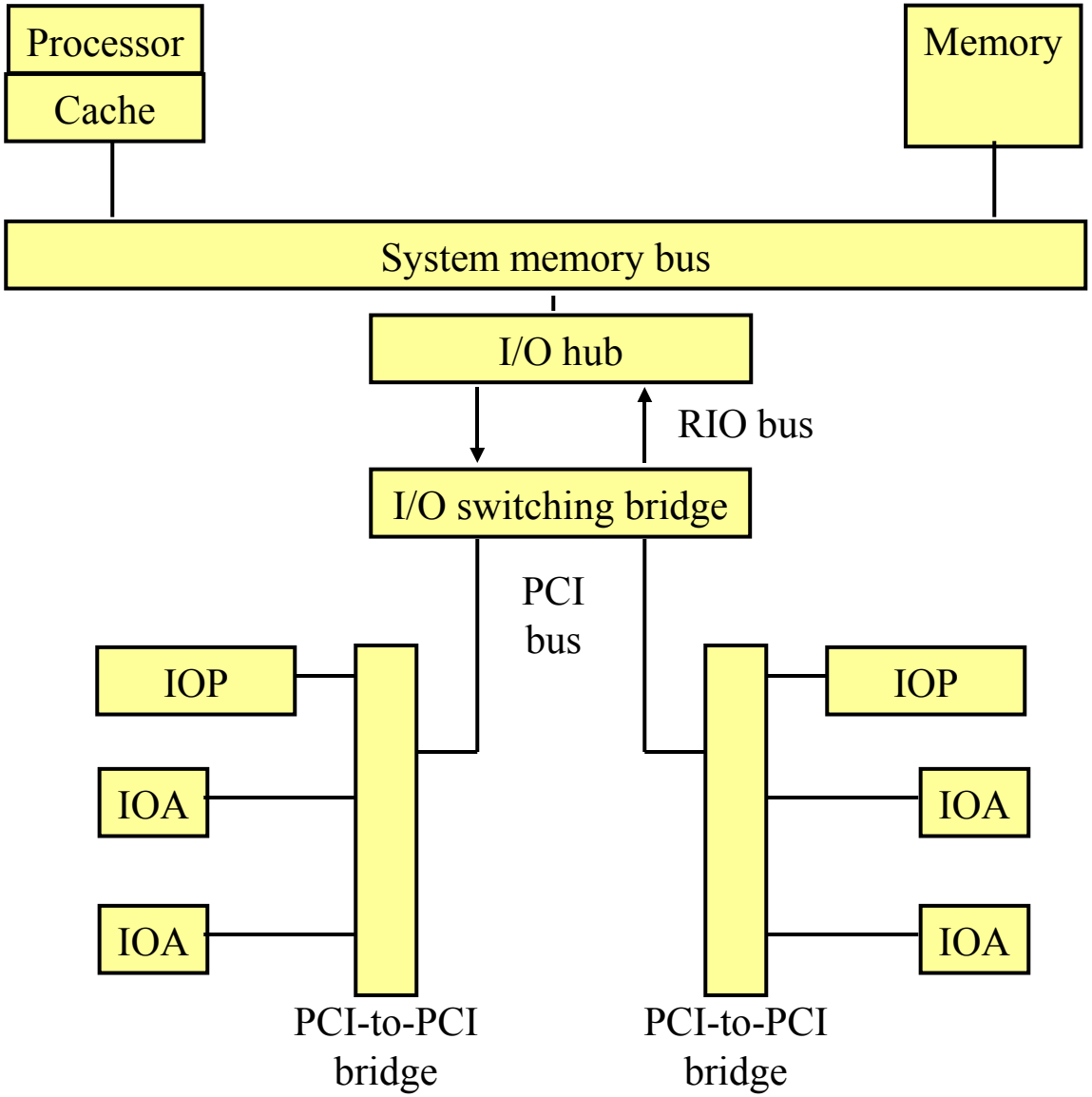
# Transition I/O Structure



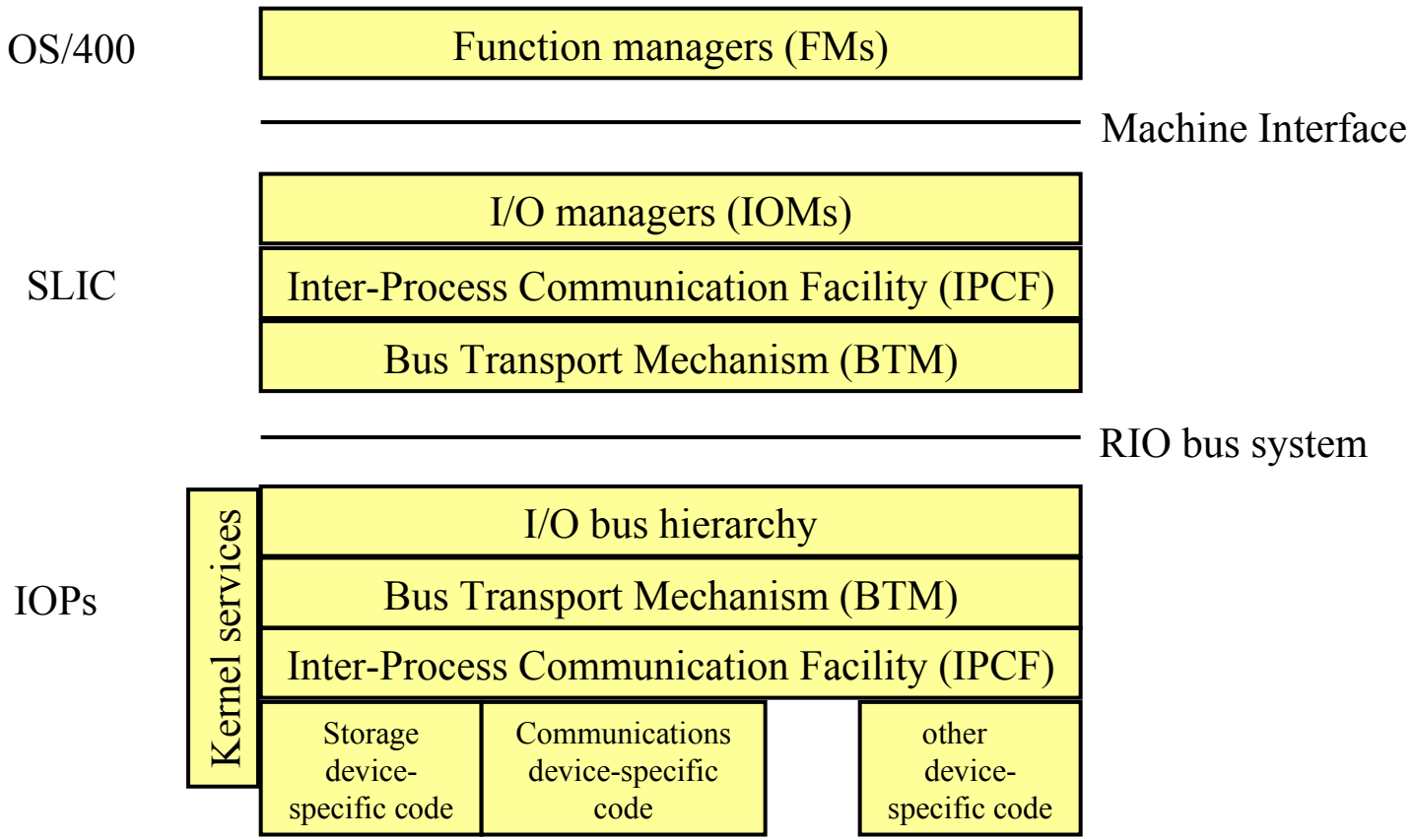
# SPD-PCI Transition

- SPD – System Product Devision
- RIO – Remote I/O
- HSL – High Speed Link
  - RIO outside the box
- PCI – Peripheral Component Interface

# RIO/PCI I/O Structure



# I/O Software Structure



# 3.5 Software Integration

# 3.5.1 Integrated Middleware

# Software Integration or Integrated OS

Designed to reduce total cost of ownership and speed time to deploy

Advanced autonomic operating Systems functions Integrated DB2 UDB (object-relational database)

TCP/IP protocols and application

OLTP

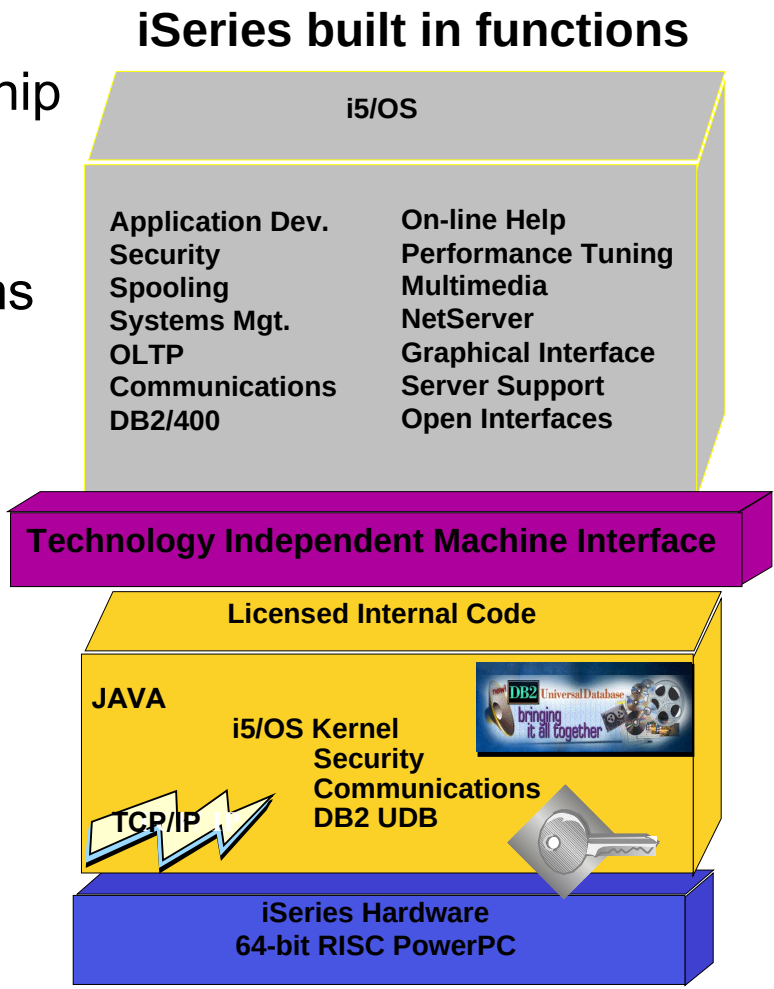
Security

Reduces costs of

Procurement

Integration of components

Maintenance of components



# 3.5.2 User Interfaces

# Human-Computer Interaction

- Human-computer interaction (HCI) is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use in a social context, and with the study of major phenomena surrounding them. *

* ACM Special Interest Group on Computer-Human Interaction Curriculum Development

# User Interfaces

- CLI – Command line interfaces
  - Text-and-keyboard oriented
  - Widely used outside desktop computing
  - Examples: DOS, Unix Shells
- GUI – graphical user interface
  - mouse-oriented: windows, pull-down menus, buttons, scroll bars, iconic images, wizards, …
  - Examples: Windows Desktop, KDE

# System i User Interfaces

- CLI
  - 5250 terminal (emulation)
- GUI
  - System i Navigator
  - System i Access for Web
  - System i Access for Wireless
  - IBM Systems Director Navigator for i5/OS
  - ...