William Stallings Computer Organization and Architecture

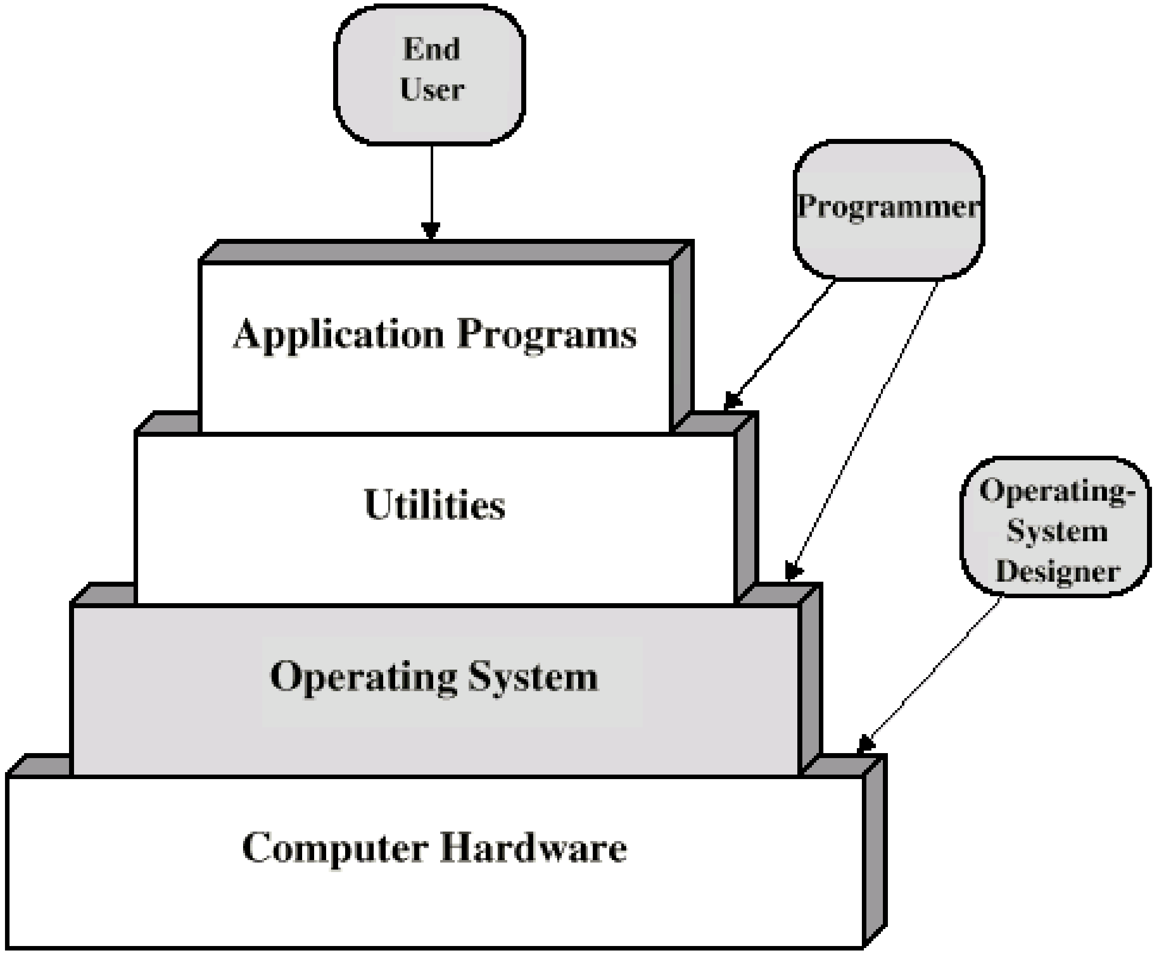
Chapter 7

Operating System Support

Objectives and Functions

- Convenience
  - Making the computer easier to use
- Efficiency
  - Allowing better use of computer resources

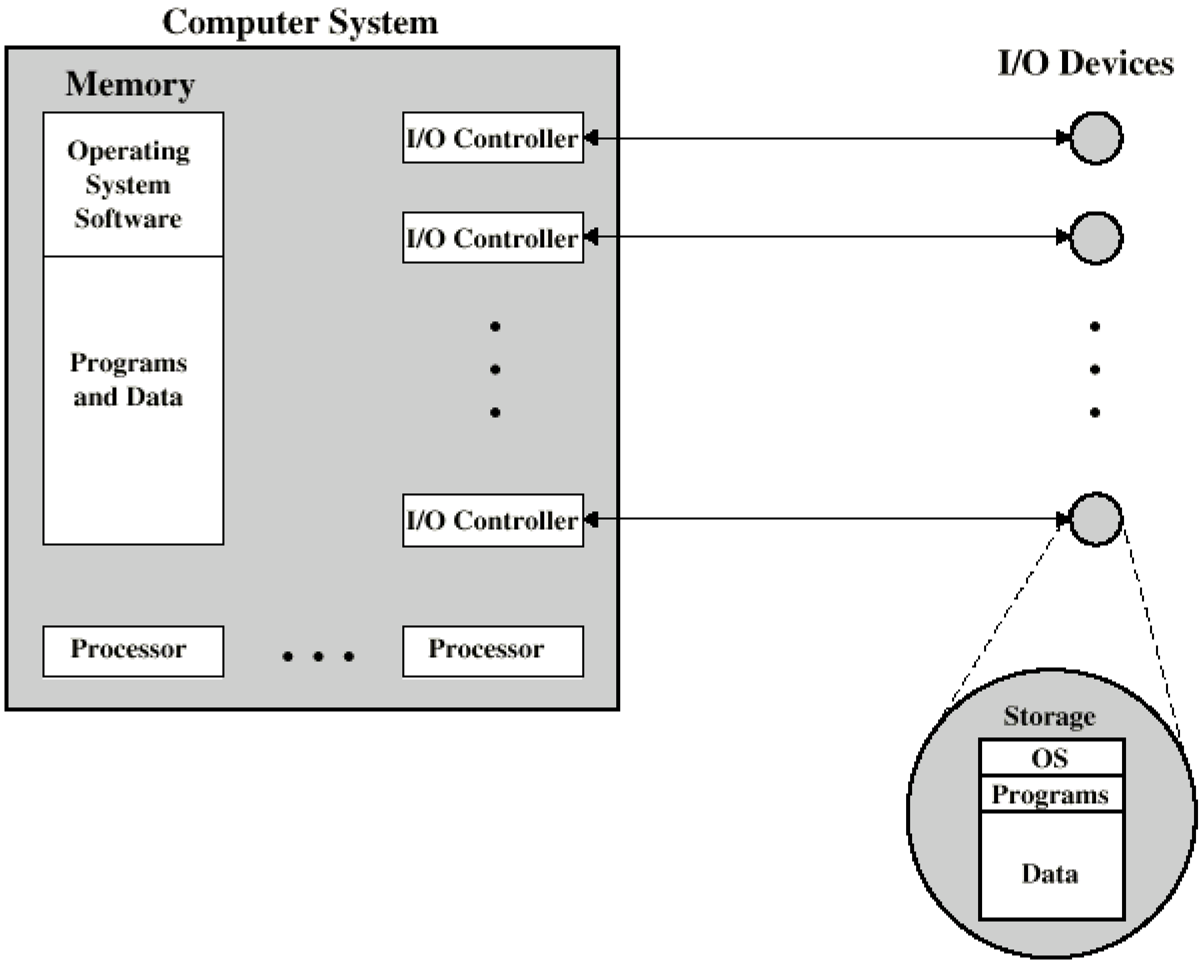
Layers and Views of a Computer System



Operating System Services

- Program creation
- Program execution
- Access to I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

O/S as a Resource Manager



Types of Operating System

- Batch
- Interactive

- Single program (Uni-programming)
- Multiple programs (Multi-tasking)

Early Systems

- Late 1940s to mid 1950s
- No Operating System
- Programs interact directly with hardware
- Two main problems:
  - Scheduling
  - Set-up time

Simple Batch Systems

- Resident Monitor program
- Users submit jobs to operator
- Operator batches jobs
- Monitor controls sequence of events to process batch
- When one job is finished, control returns to Monitor which reads next job
- Monitor handles scheduling

Job Control Language

- Instructions to Monitor
- Usually denoted by $
- e.g.
  - $JOB
  - $FTN
  - ...   Some Fortran instructions
  - $LOAD
  - $RUN
  - ...   Some data
  - $END

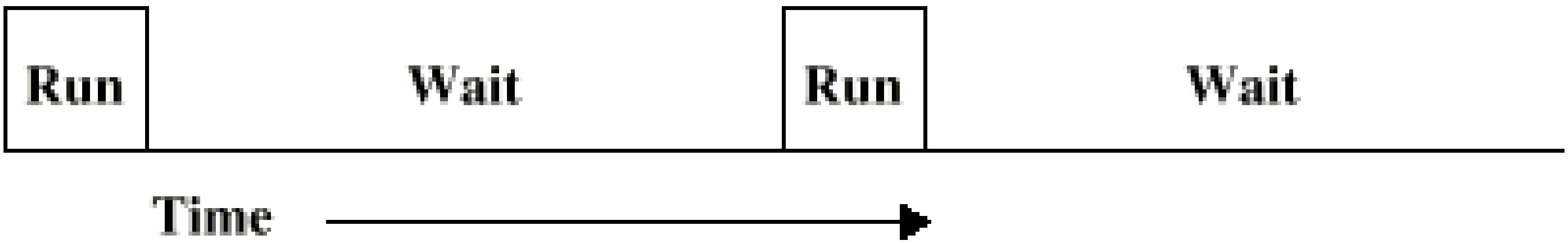
Desirable Hardware Features

- Memory protection
  - To protect the Monitor
- Timer
  - To prevent a job monopolizing the system
- Privileged instructions
  - Only executed by Monitor
  - e.g. I/O
- Interrupts
  - Allows for relinquishing and regaining control

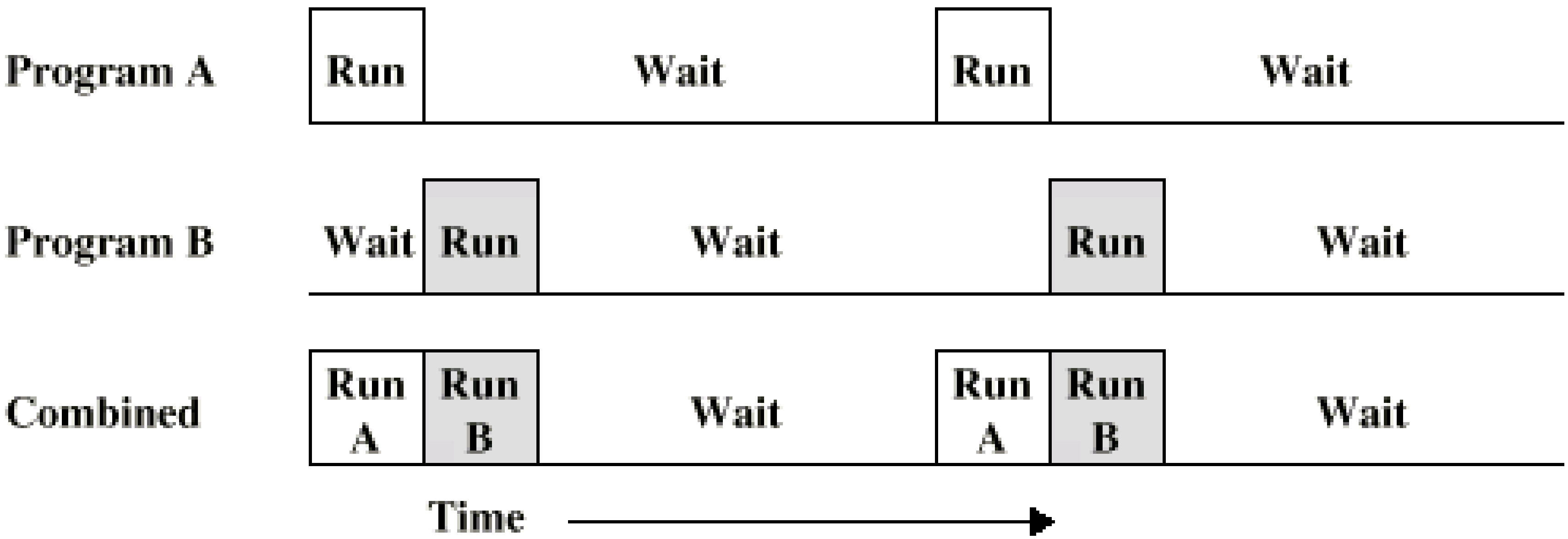
Multi-programmed Batch Systems

- I/O devices very slow
- When one program is waiting for I/O, another can use the CPU

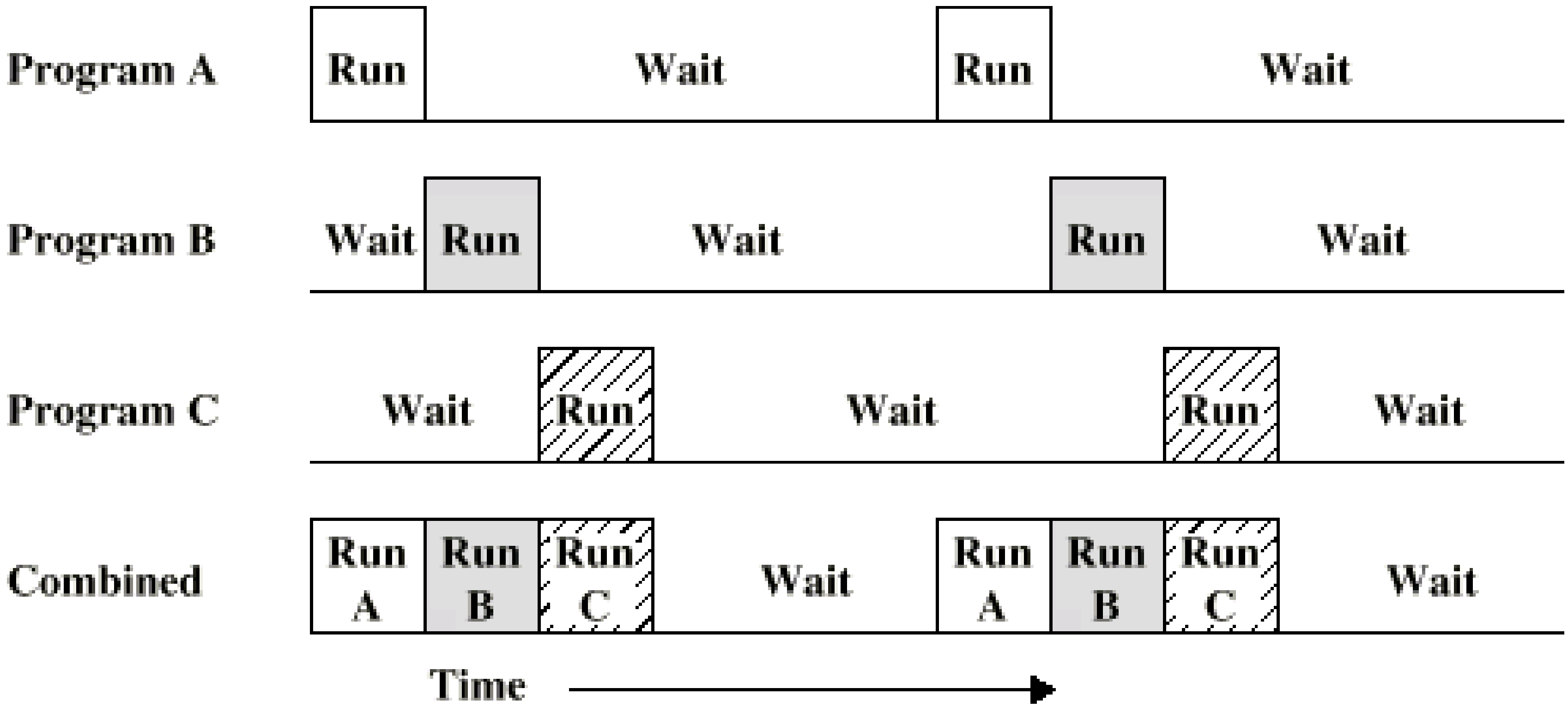
Single Program



Multi-Programming with Two Programs



Multi-Programming with Three Programs



Time Sharing Systems

- Allow users to interact directly with the computer
  - i.e. Interactive
- Multi-programming allows a number of users to interact with the computer

Scheduling

- Key to multi-programming
- Long term
- Medium term
- Short term
- I/O

Long Term Scheduling

- Determines which programs are accepted for processing
  - i.e. controls the degree of multi-programming
- Once accepted, a job becomes a process for the short term scheduler
- (or it becomes a swapped out job for the medium term scheduler)

Medium Term Scheduling

- Determines which process can be entered in the central memory (i.e., swapped in)
- Part of the swapping function (later…)
- Usually based on the need to manage multi-programming
- If no virtual memory, memory management is also an issue

Short Term Scheduler

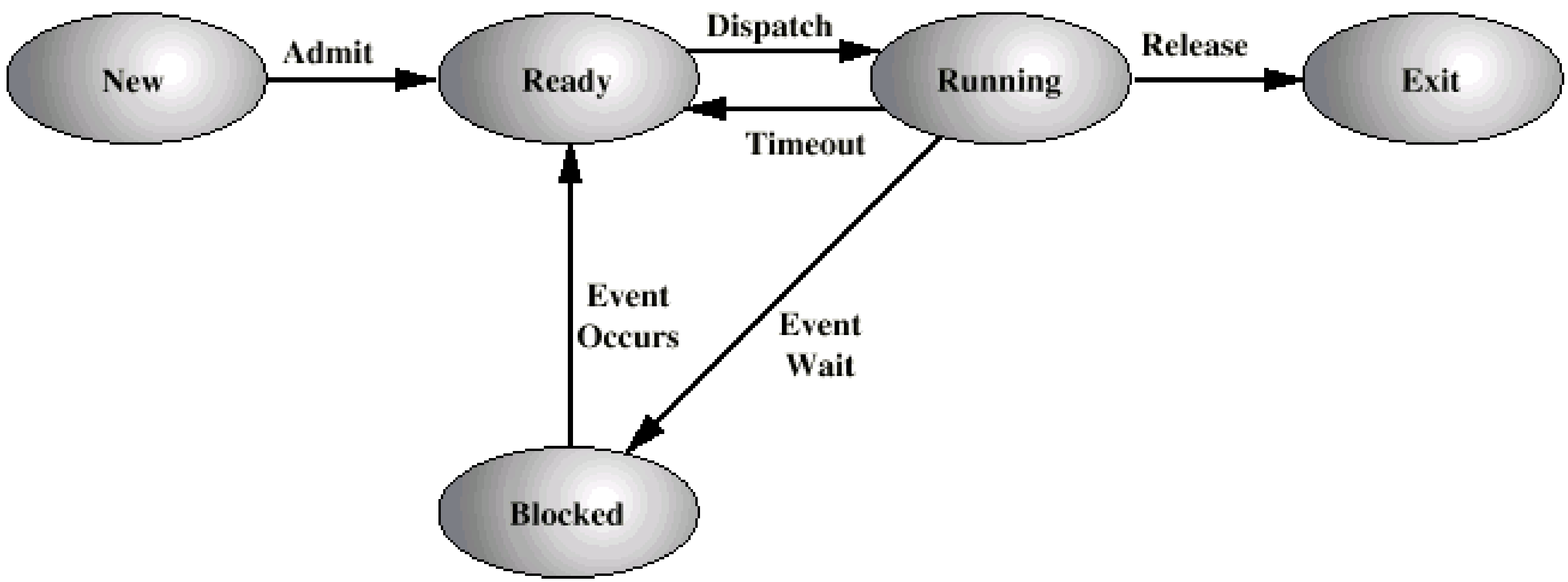
- Dispatcher
- Fine grained decisions of which job to execute next
- i.e. which job actually gets to use the processor in the next time slot

Process States

Long Term Sched.

Medium Term Sched.

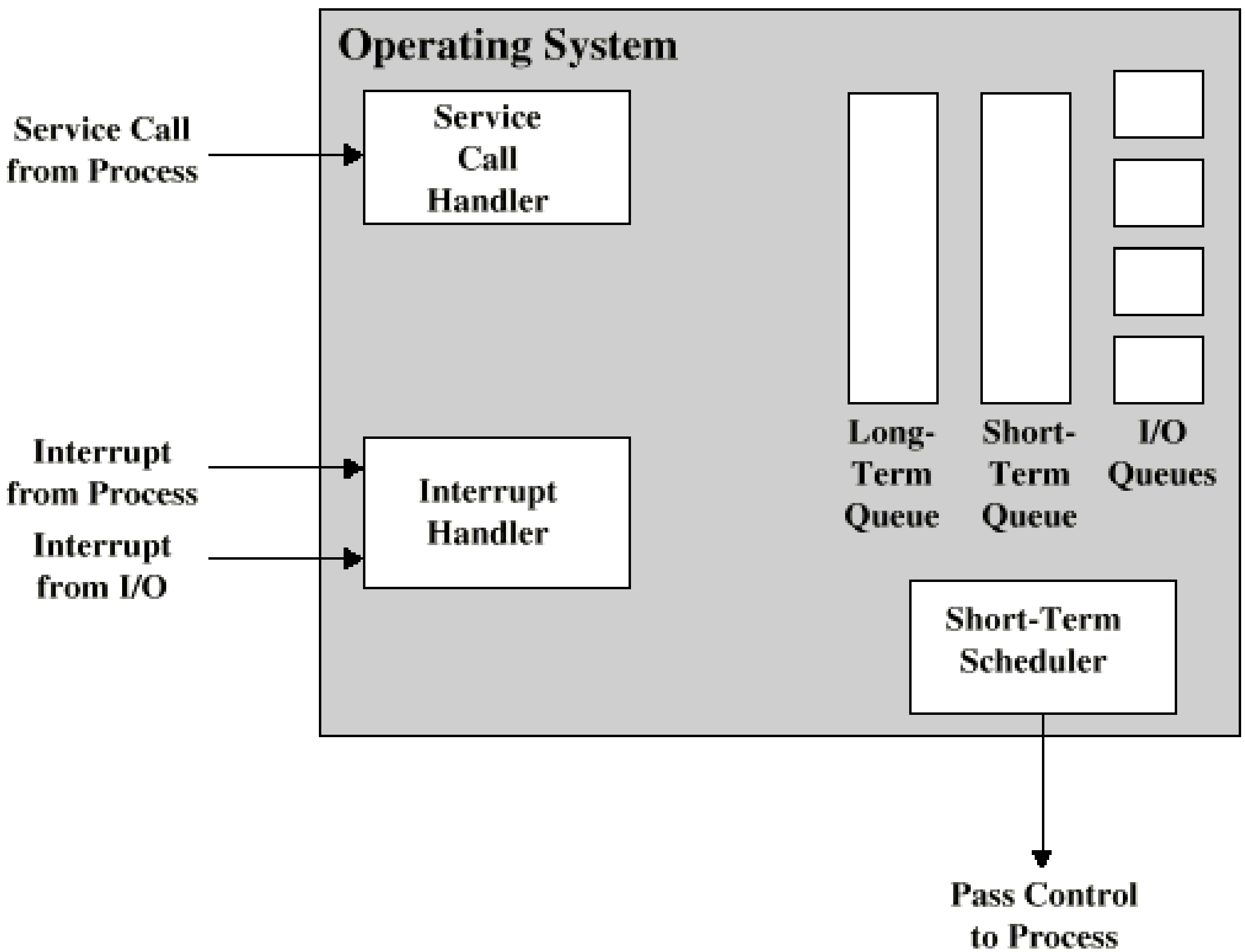
Short Term Sched.



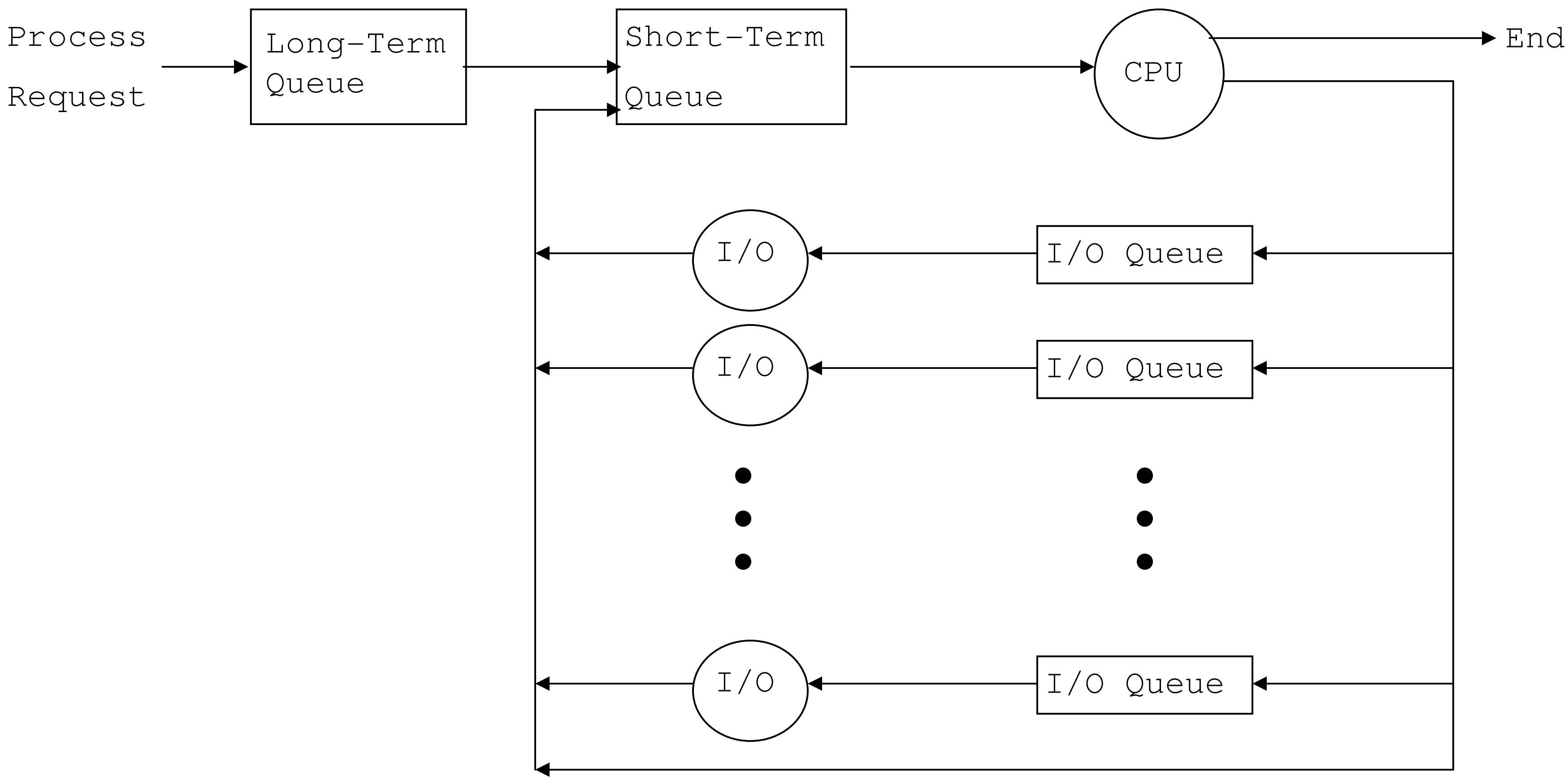
Process Control Block

- Identifier
- State
- Priority
- Program counter
- Process Memory pointers
- Context data
- I/O status
- Accounting information

Key Elements of O/S



Process Scheduling



Memory Management

- Uni-program
  - Memory split into two
  - One for Operating System (monitor)
  - One for currently executing program
- Multi-program
  - “User” part is sub-divided and shared among active processes
  - Requires memory management capabilities

Swapping

- Problem: I/O is so slow compared with CPU that even in multi-programming system, CPU can be idle most of the time
- Solutions:
  - Increase main memory
    - Expensive
    - Leads to larger programs
  - Swapping

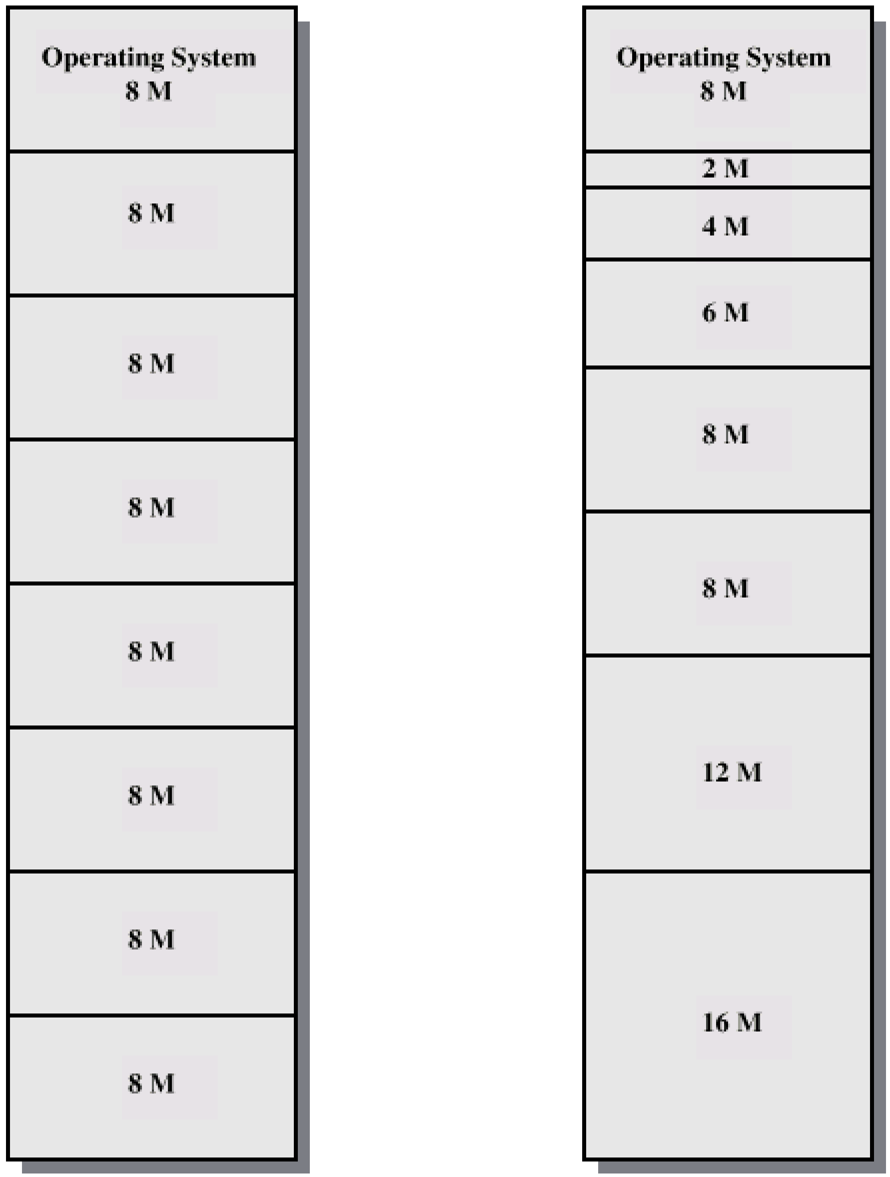
What is Swapping?

- Long term queue of processes stored on disk
- Processes "swapped" in as space becomes available
- As a process completes it is moved out of main memory
- If none of the processes in memory are ready (i.e. all I/O blocked)
  - Swap out a blocked process to intermediate queue
  - Swap in a ready process or a new process
  - But swapping is an I/O process...

Partitioning

- Splitting memory into sections to allocate to processes (including Operating System)
- Fixed-sized partitions
  - May not be equal size
  - Process is fitted into smallest hole that will take it (best fit)
  - Some wasted memory
  - Leads to variable sized partitions

Fixed Partitioning



(a) Equal-size partitions

(b) Unequal-size partitions

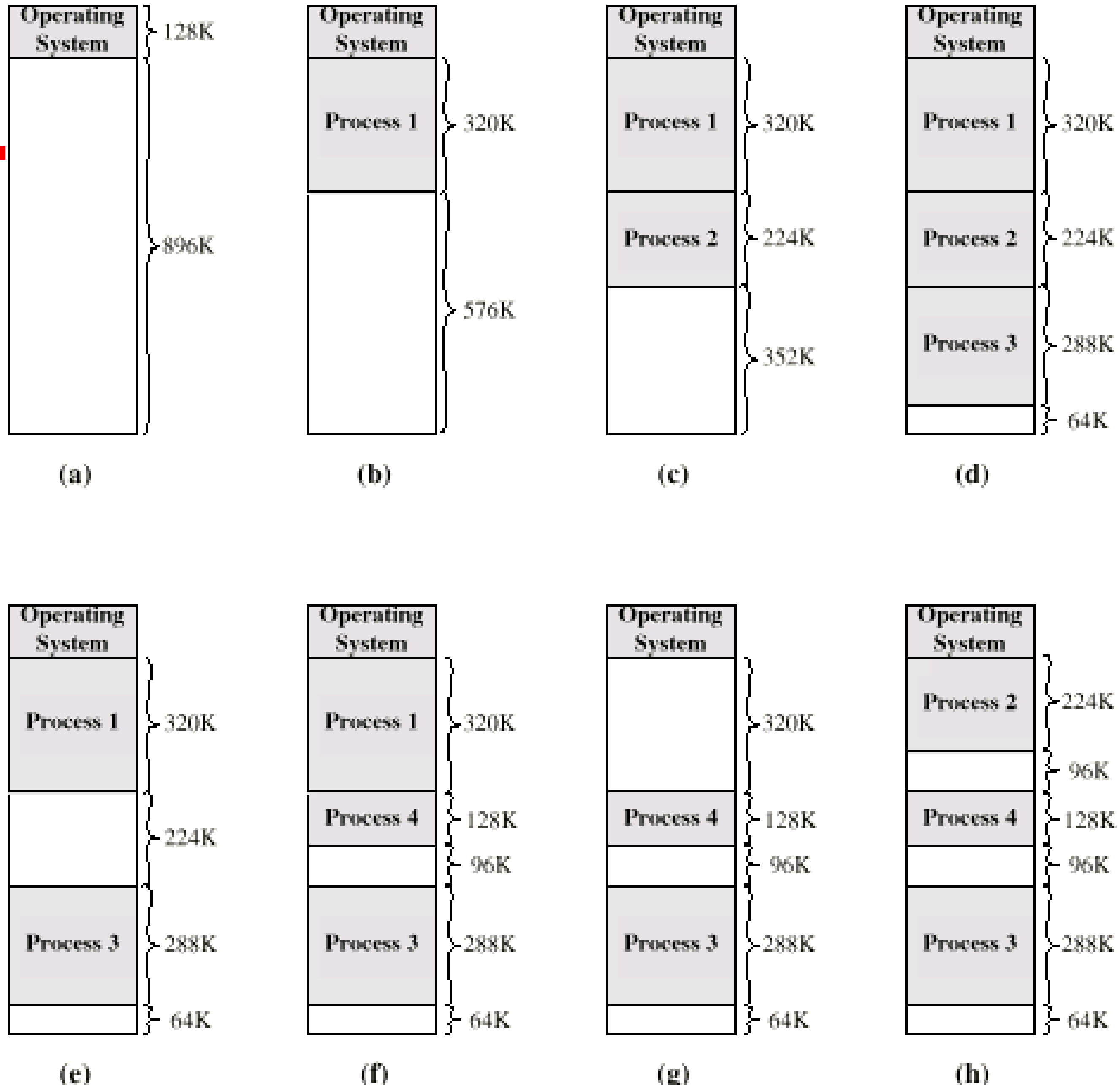
Variable Sized Partitions (1)

- Allocate exactly the required memory to a process
- This leads to a hole at the end of memory, too small to use
  - Only one small hole - less waste
- When all processes are blocked, swap out a process and bring in another
- New process may be smaller than swapped out process
- Another hole

Variable Sized Partitions (2)

- Eventually have lots of holes (fragmentation)
- Solutions:
  - Coalesce - Join adjacent holes into one large hole
  - Compaction - From time to time go through memory and move all hole into one free block (c.f. disk de-fragmentation)

Effect of Dynamic Partitioning



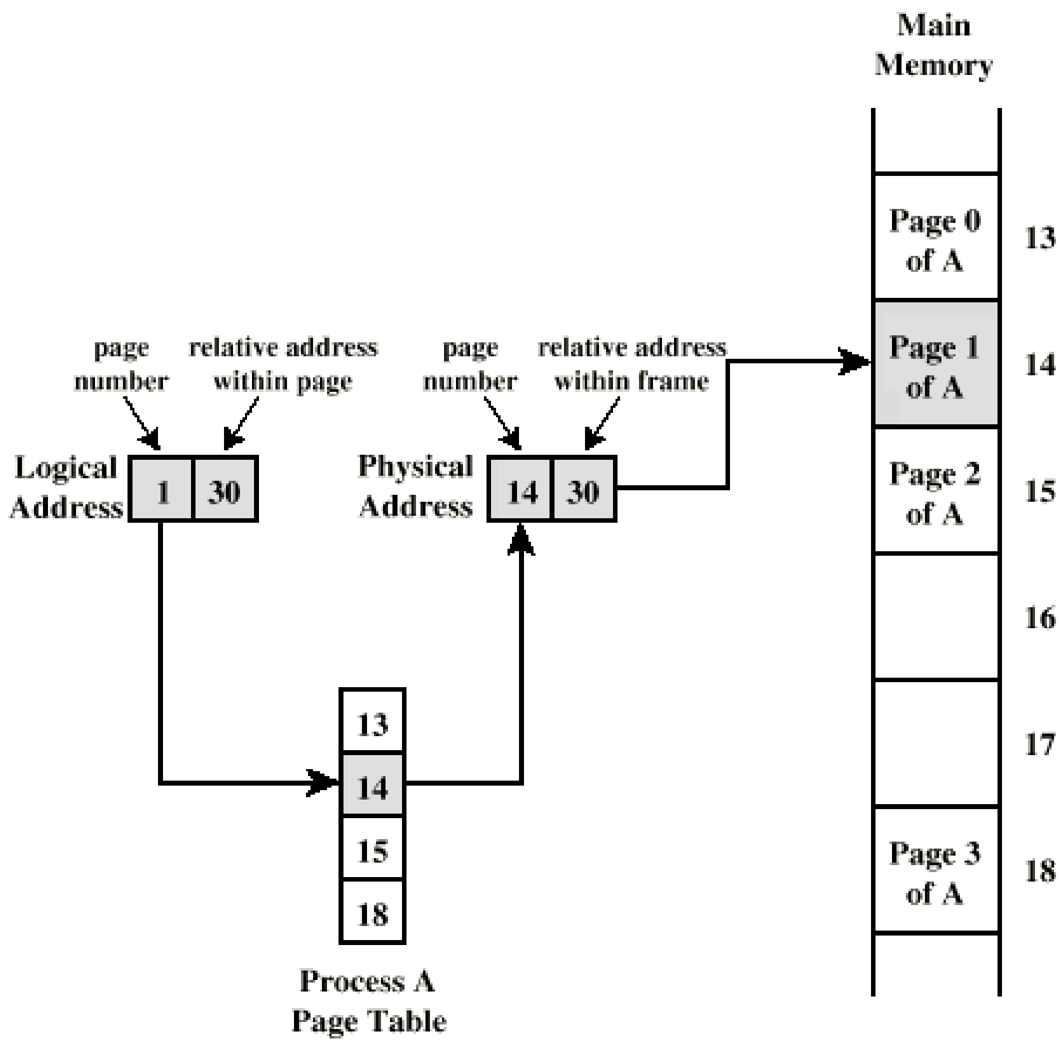
Relocation

- No guarantee that process will load into the same place in memory
- Instructions contain addresses
  - Locations of data
  - Addresses for instructions (branching)
- Logical address - relative to beginning of program
- Physical address - actual location in memory (this time)
- Automatic conversion using base address

Paging

- Split memory into equal sized, small chunks - page frames
- Split programs (processes) into equal sized small chunks - pages
- Allocate the required number page frames to a process
- Operating System maintains list of free frames
- A process does not require contiguous page frames
- Use page table to keep track

Logical and Physical Addresses - Paging



Virtual Memory

- Demand paging
  - Do not require all pages of a process in memory
  - Bring in pages as required
- Page fault
  - Required page is not in memory
  - Operating System must swap in required page
  - May need to swap out a page to make space
  - Select page to throw out based on recent history

Thrashing

- Too many processes in too little memory
- Operating System spends all its time swapping
- Little or no real work is done
- Disk light is on all the time

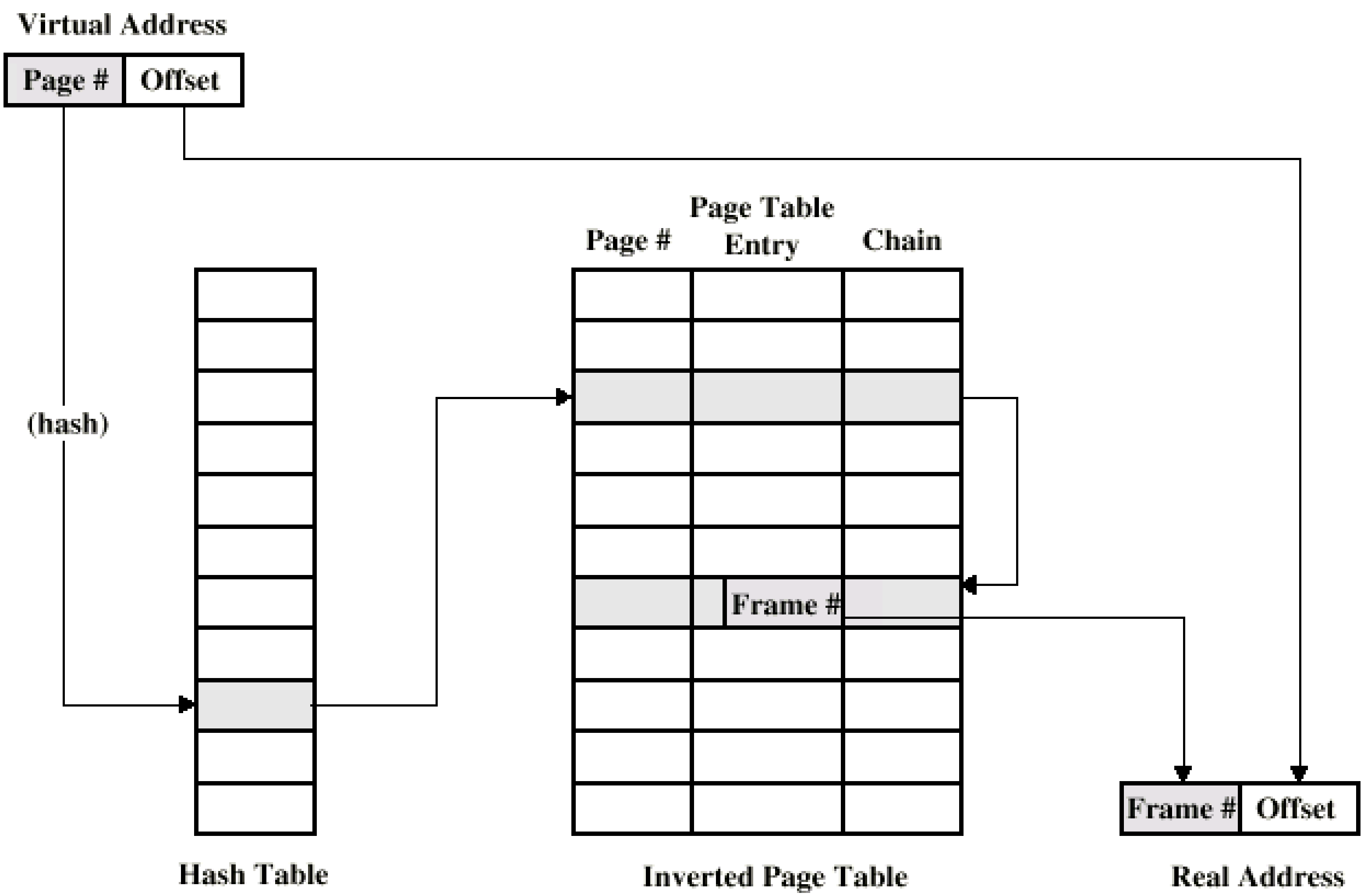
- Solutions
  - Good page replacement algorithms
  - Reduce number of processes running
  - Fit more memory

Bonus

- We do not need all of a process in memory for it to run
- We can swap in pages as required
- So - we can now run processes that are bigger than total memory available!

- Main memory is called real memory
- User/programmer sees much bigger memory - virtual memory

Page Table Structure



Segmentation

- Paging is not (usually) visible to the programmer
- Segmentation is visible to the programmer
- Usually different segments allocated to program and data
- May be a number of program and data segments

Advantages of Segmentation

- Simplifies handling of growing data structures
- Allows programs to be altered and recompiled independently, without re-linking and re-loading
- Lends itself to sharing among processes
- Lends itself to protection
- Some systems combine segmentation with paging