Advanced Computer Architecture

COMP 5123

Fall 2016

Computer Science Department

Prairie View A&M University

Mary Heejin Kim (aka Heejin Lim), Ph.D.

Chapter 8

Operating System Support

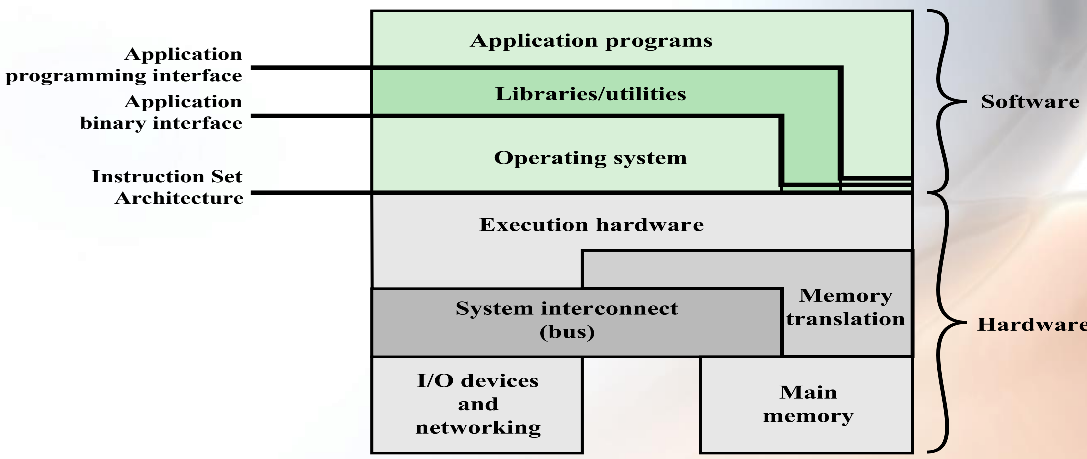


Figure 8.1 Computer Hardware and Software Structure

Operating System (OS) Services

- The most important system program
- Masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system
- The OS typically provides services in the following areas:
  - Program creation
  - Program execution
  - Access to I/O devices
  - Controlled access to files
  - System access
  - Error detection and response
  - Accounting

Interfaces

- Key interfaces in a typical computer system:

Instruction set architecture (ISA)
- Defines the machine language instructions that a computer can follow
- Boundary between hardware and software

Application binary interface (ABI)
- Defines a standard for binary portability across programs
- Defines the system call interface to the operating system and the hardware resources and services available in a system through the user ISA

Application programming interface (API)
- Gives a program access to the hardware resources and services available in a system through the user ISA supplemented with high-level language (HLL) library calls
- Using an API enables application software to be ported easily to other systems that support the same API

+

Operating System as Resource Manager

A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions

- **The OS is responsible for managing these resources**

The OS as a control mechanism is unusual in two respects:

- **The OS functions in the same way as ordinary computer software – it is a program executed by the processor**
- **The OS frequently relinquishes control and must depend on the processor to allow it to regain control**

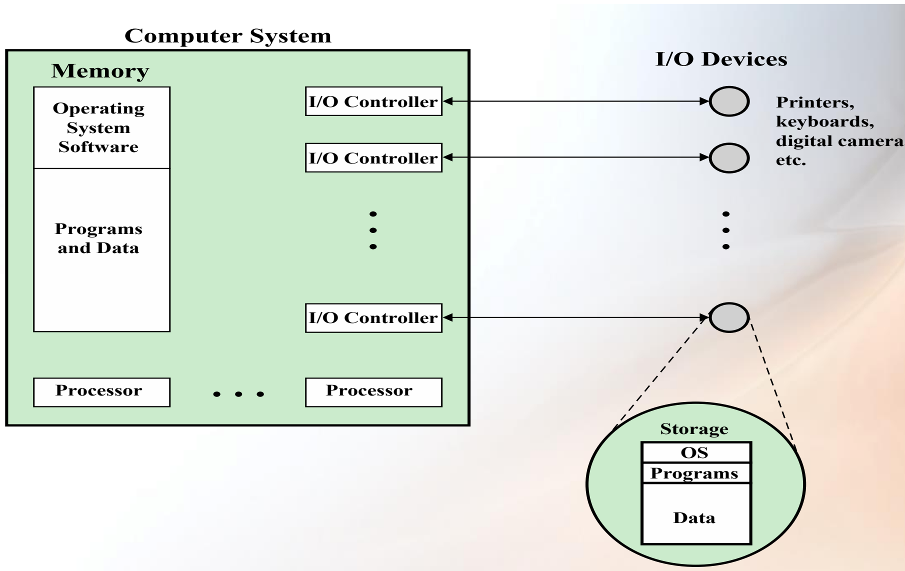
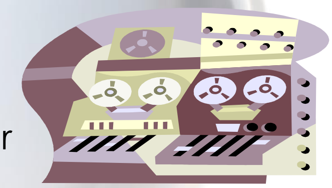


Figure 8.2 The Operating System as Resource Manager

Types of Operating Systems

- Interactive system
  - The user/programmer interacts directly with the computer to request the execution of a job or to perform a transaction
  - User may, depending on the nature of the application, communicate with the computer during the execution of the job
- Batch system
  - Opposite of interactive
  - The user's program is batched together with programs from other users and submitted by a computer operator
  - After the program is completed results are printed out for the user

Early Systems



- From the late 1940s to the mid-1950s the programmer interacted directly with the computer hardware – there was no OS
  - Processors were run from a console consisting of display lights, toggle switches, some form of input device and a printer
- Problems:
  - Scheduling
    - Sign-up sheets were used to reserve processor time
      - This could result in wasted computer idle time if the user finished early
      - If problems occurred the user could be forced to stop before resolving the problem
  - Setup time
    - A single program could involve
      - Loading the compiler plus the source program into memory
      - Saving the compiled program
      - Loading and linking together the object program and common functions

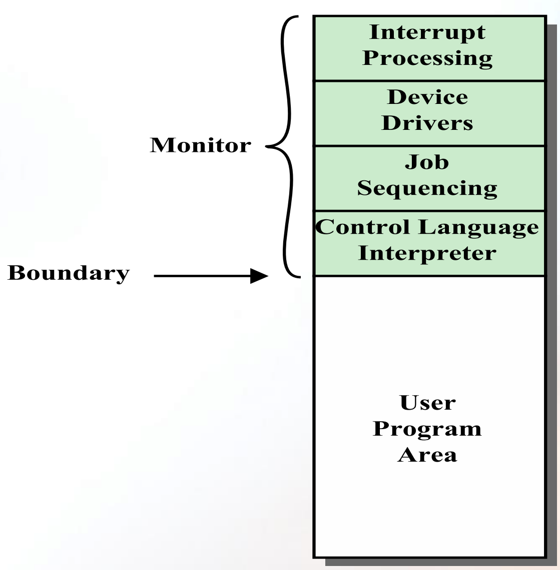


Figure 8.3 Memory Layout for a Resident Monitor

From the View of the Processor . . .

- Processor executes instructions from the portion of main memory containing the monitor
  - These instructions cause the next job to be read in another portion of main memory
  - The processor executes the instruction in the user's program until it encounters an ending or error condition
  - Either event causes the processor to fetch its next instruction from the monitor program
- The monitor handles setup and scheduling
  - A batch of jobs is queued up and executed as rapidly as possible with no idle time
- Job control language (JCL)
  - Special type of programming language used to provide instructions to the monitor
- Example:
  - $JOB
  - $FTN
  - ... Some Fortran instructions
  - $LOAD
  - $RUN
  - ... Some data
  - $END

**Each FORTRAN instruction and each item of data is on a separate punched card or a separate record on tape. In addition to FORTRAN and data lines, the job includes job control instructions, which are denoted by the beginning "$".

- Monitor, or batch OS, is simply a computer program
  - It relies on the ability of the processor to fetch instructions from various portions of main memory in order to seize and relinquish control alternately

+

Desirable Hardware Features

- Memory protection
  - User program must not alter the memory area containing the monitor
  - The processor hardware should detect an error and transfer control to the monitor
  - The monitor aborts the job, prints an error message, and loads the next job
- Timer
  - Used to prevent a job from monopolizing the system
  - If the timer expires an interrupt occurs and control returns to monitor
- Privileged instructions
  - Can only be executed by the monitor
  - If the processor encounters such an instruction while executing a user program an error interrupt occurs
  - I/O instructions are privileged so the monitor retains control of all I/O devices
- Interrupts
  - Gives the OS more flexibility in relinquishing control to and regaining control from user programs

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Figure 8.4 System Utilization Example

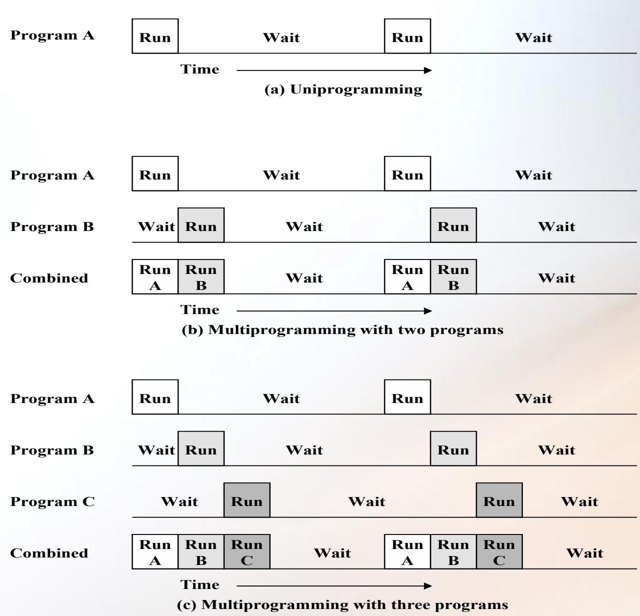


Figure 8.5 Multiprogramming Example

Table 8.1
Sample Program Execution Attributes

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	80 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Table 8.2
Effects of Multiprogramming on Resource Utilization

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput rate	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

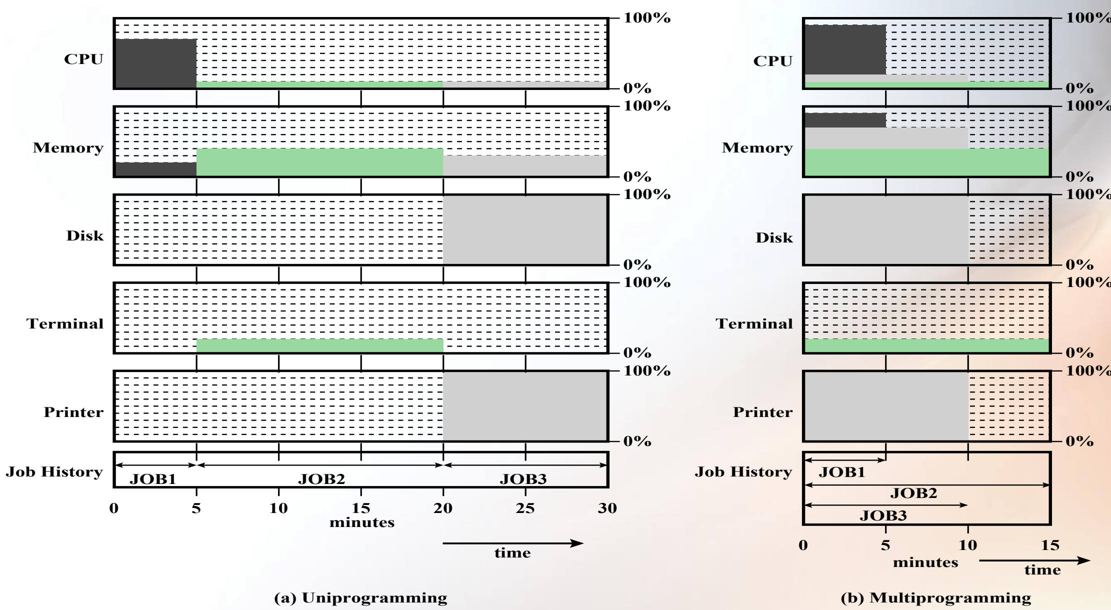
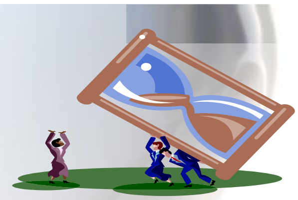


Figure 8.6 Utilization Histograms

Time Sharing Systems



- Used when the user interacts directly with the computer
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation
- Example:
  - If there are *n* users actively requesting service at one time, each user will only see on the average 1/*n* of the effective computer speed

Table 8.3
Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Table 8.4 Types of Scheduling

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device

Long Term Scheduling

Determines which programs are submitted for processing

Once submitted, a job becomes a process for the short term scheduler

In some systems a newly created process begins in a swapped-out condition, in which case it is added to a queue for the medium-term scheduler

Batch system
- Newly submitted jobs are routed to disk and held in a batch queue
- The long-term scheduler creates processes from the queue when it can

Time-sharing system
- A process request is generated when a user attempts to connect to the system
- OS will accept all authorized comers until the system is saturated
- At that point a connection request is met with a message indicating that the system is full and to try again later

Medium-Term Scheduling and Short-Term Scheduling

Medium-Term
- Part of the swapping function
- Swapping-in decision is based on the need to manage the degree of multiprogramming
- Swapping-in decision will consider the memory requirements of the swapped-out processes

Short-Term
- Also known as the dispatcher
- Executes frequently and makes the fine-grained decision of which job to execute next

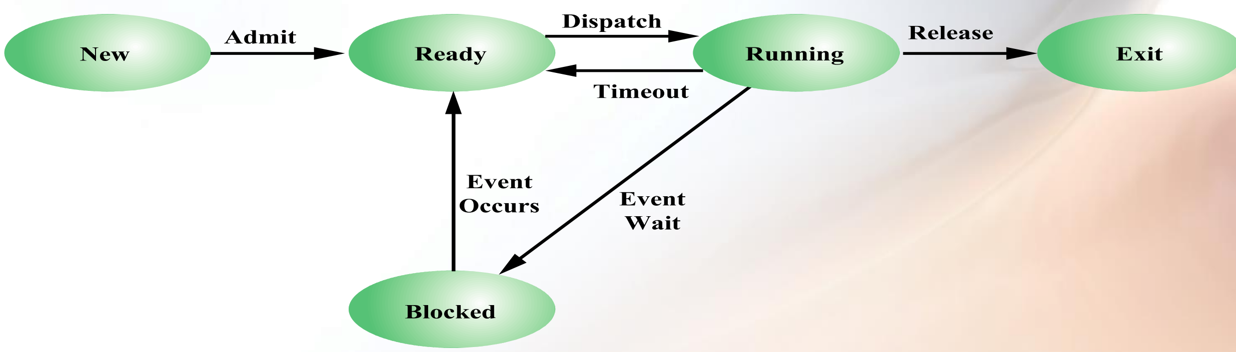


Figure 8.7 Five-State Process Model

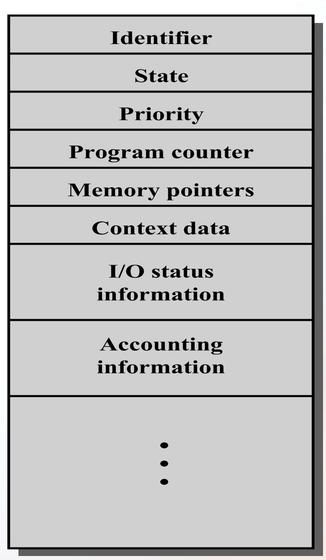


Figure 8.8 Process Control Block

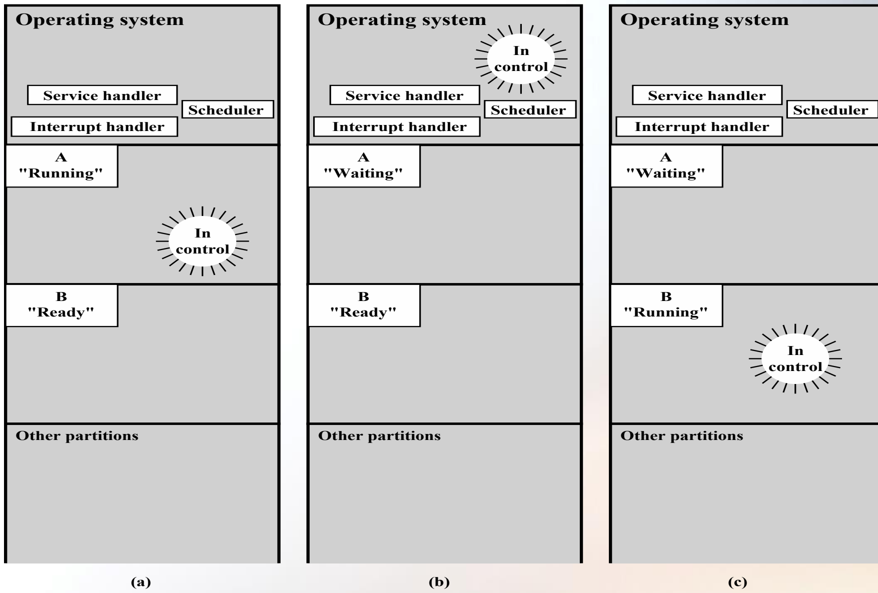


Figure 8.9 Scheduling Example

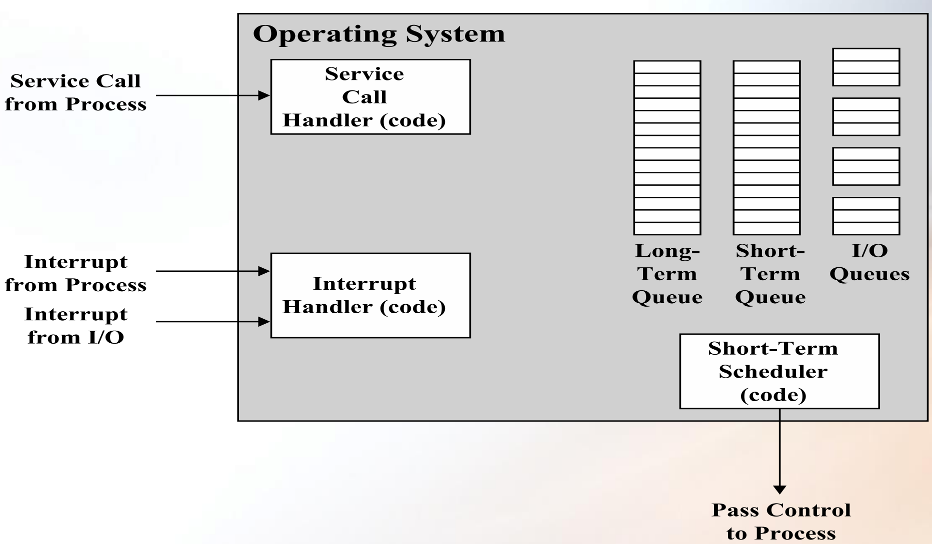


Figure 8.10 Key Elements of an Operating System for Multiprogramming

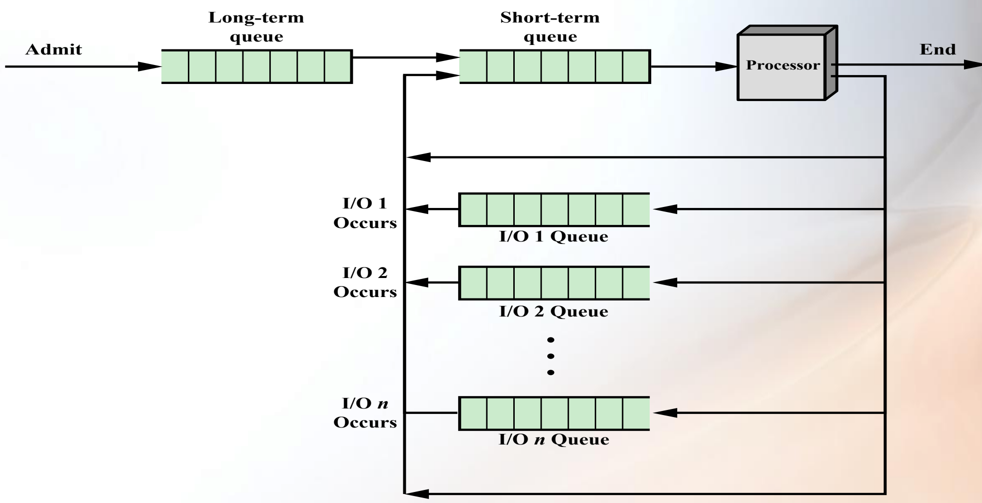


Figure 8.11 Queuing Diagram Representation of Processor Scheduling

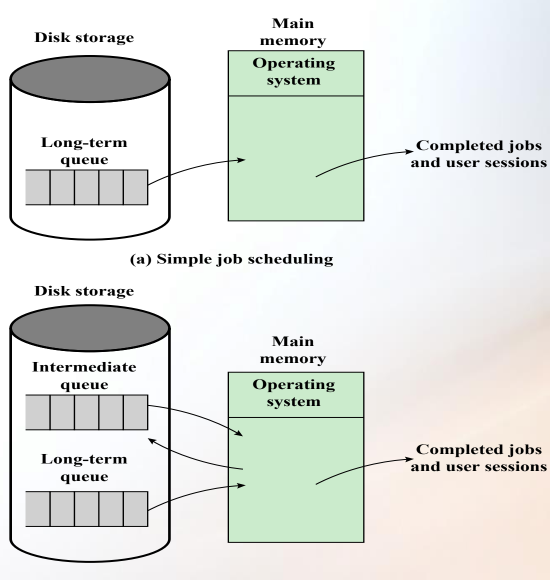
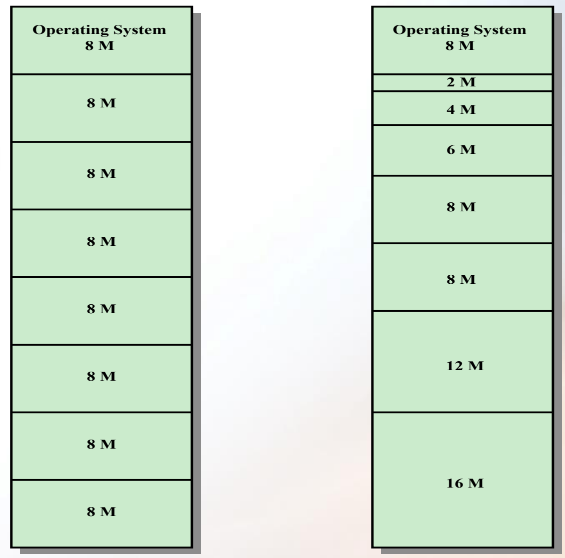


Figure 8.12 The Use of Swapping



(a) Equal-size partitions

(b) Unequal-size partitions

Figure 8.13 Example of Fixed Partitioning of a 64-Mbyte Memory

Logical address
- expressed as a location relative to the beginning of the program

Physical address
- an actual location in main memory

Base address
- current starting location of the process

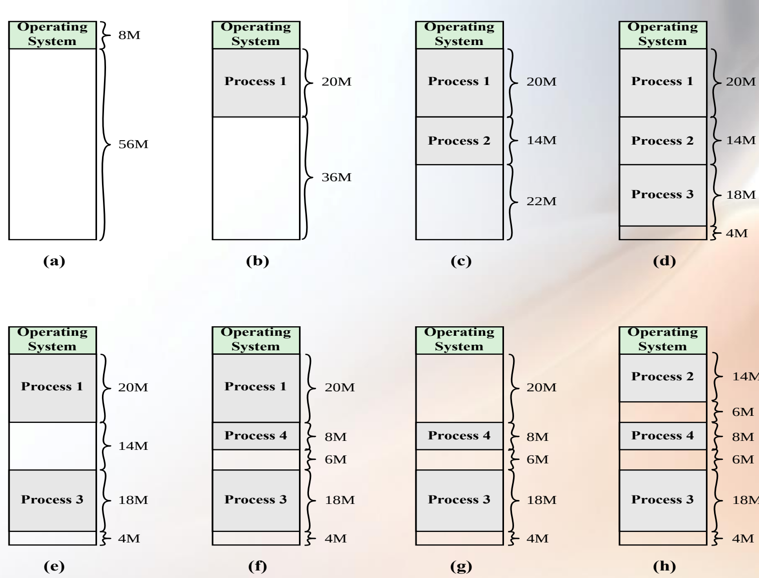


Figure 8.14 The Effect of Dynamic Partitioning

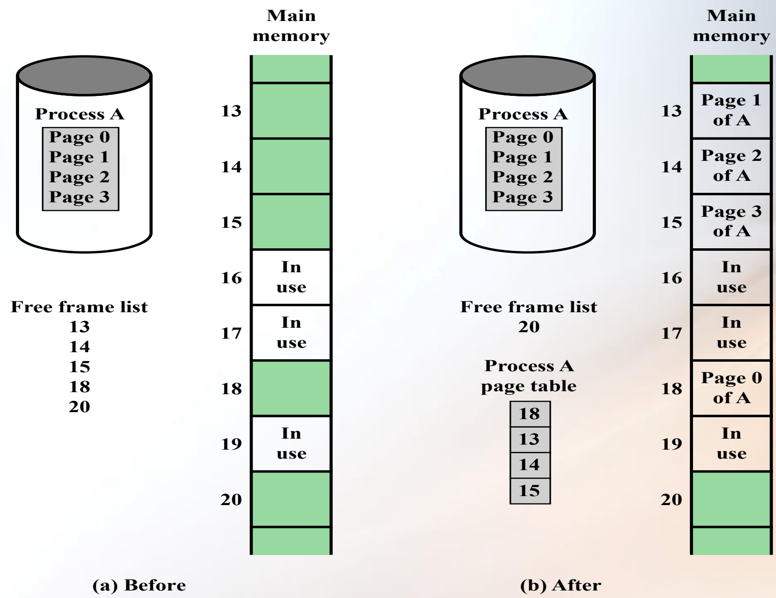


Figure 8.15 Allocation of Free Frames

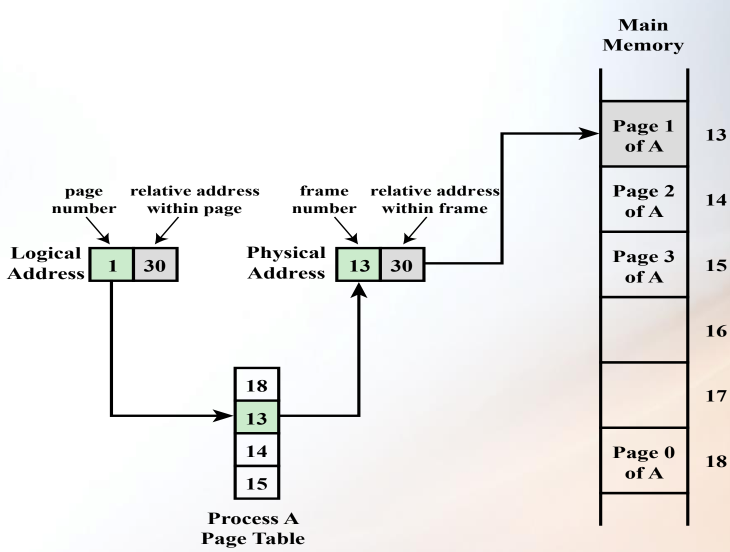


Figure 8.16 Logical and Physical Addresses

Virtual Memory

Demand Paging

- Each page of a process is brought in only when it is needed
- Principle of locality
  - When working with a large process execution may be confined to a small section of a program (subroutine)
  - It is better use of memory to load in just a few pages
  - If the program references data or branches to an instruction on a page not in main memory, a *page fault* is triggered which tells the OS to bring in the desired page
- Advantages:
  - More processes can be maintained in memory
  - Time is saved because unused pages are not swapped in and out of memory
- Disadvantages:
  - When one page is brought in, another page must be thrown out (*page replacement)*
  - If a page is thrown out just before it is about to be used the OS will have to go get the page again
  - *Thrashing*
    - When the processor spends most of its time swapping pages rather than executing instructions

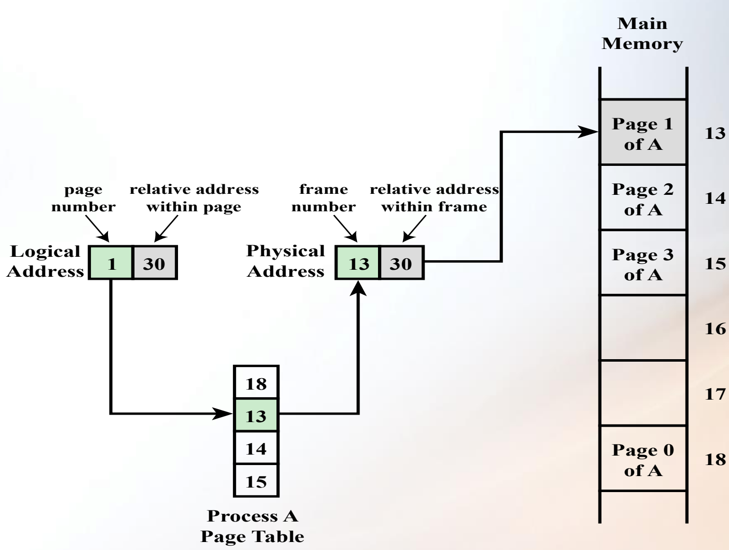


Figure 8.16 Logical and Physical Addresses

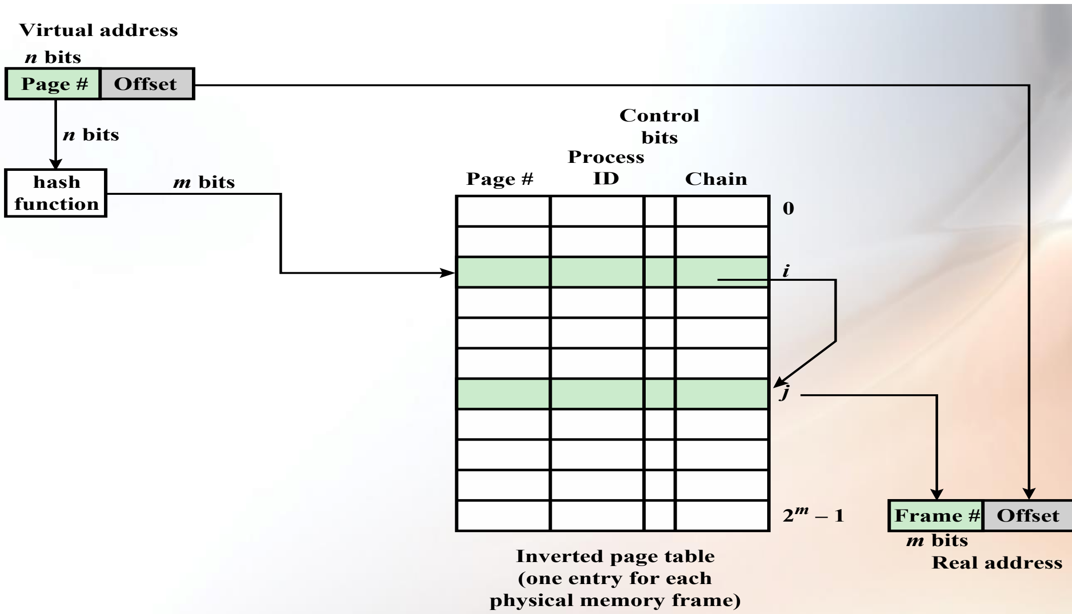


Figure 8.17 Inverted Page Table Structure

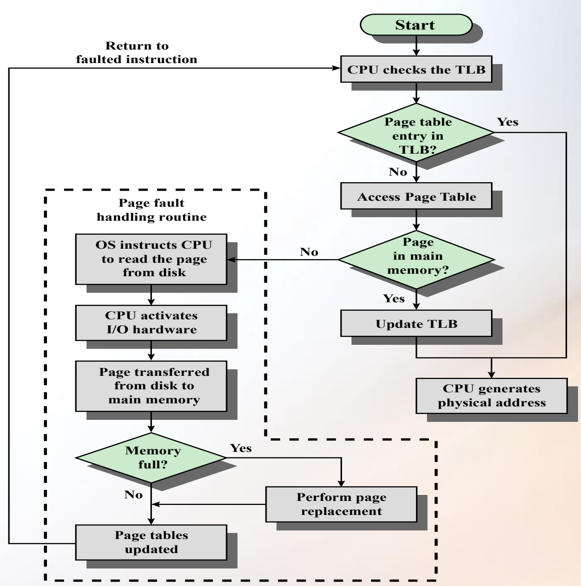


Figure 8.18 Operation of Paging and Translation Lookaside Buffer (TLB) [FURH87]

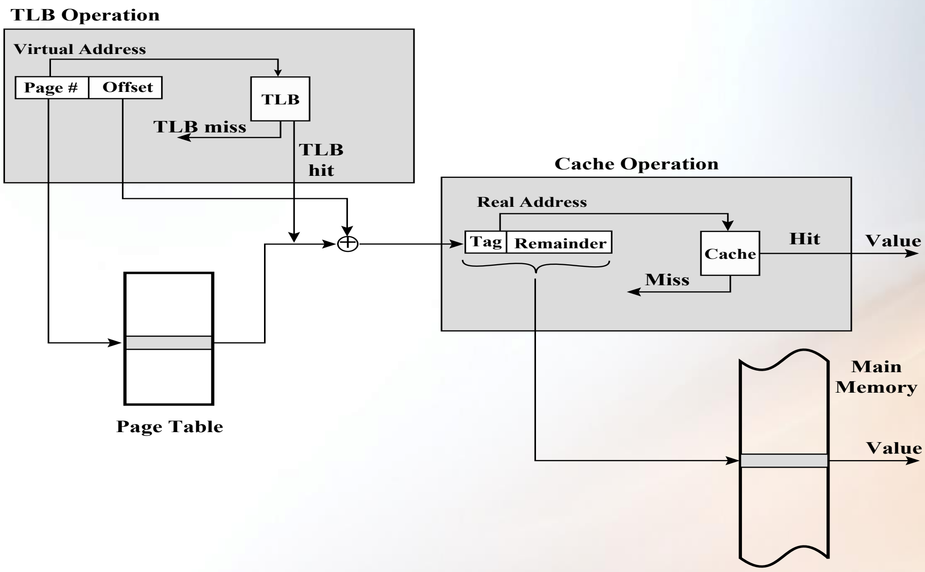


Figure 8.19 Translation Lookaside Buffer and Cache Operation