Computer Architecture

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr
nizamettinaydin@gmail.com

http://www.yildiz.edu.tr/~naydin

Computer Architecture

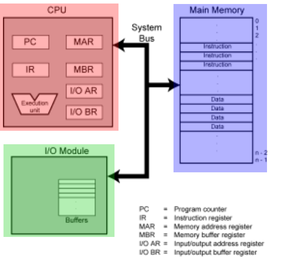
Operating System Support

Outline

- Objectives and Functions of OS
- Operating System Services
- Types of Operating System
  - Interactive
  - Batch
  - Single program
  - Multi-programming
- Scheduling
  - Long Term Scheduling
  - Medium Term Scheduling
  - Short Term Scheduler
- Swapping
- Partitioning
- Relocation
- Paging
- Virtual Memory
- Segmentation

Operating System

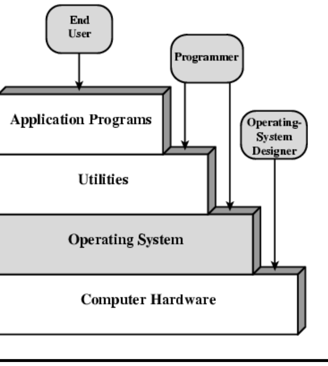
- A computer consists of modules of three basic types that communicate with each other.
  - CPU
  - Memory
  - Input/Output
- Management of these modules is done by OS.



Objectives and Functions of OS

- Convenience
  - An operating system makes a computer easier to use
- Efficiency
  - An operating system allows better use of computer resources

Layers and Views of a Computer System



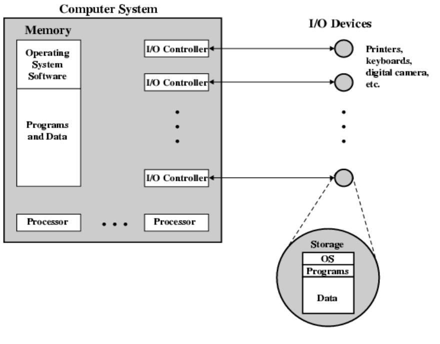
Operating System Services

- 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

- A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions
- The O/S is responsible for managing these resources
- O/S is a program executed by the processor
- The O/S frequently relinquishes control and must depend on the processor to allow it to regain control

Main Resources managed by the O/S



Types of Operating System

- Interactive
  - User/programmer interacts directly with the computer through a keyboard/display terminal
- Batch
  - Opposite of interactive. Rare
- Single program (Uni-programming)
  - Works only one program at atime
- Multi-programming (Multi-tasking)
  - Processor works on more than one program at a time

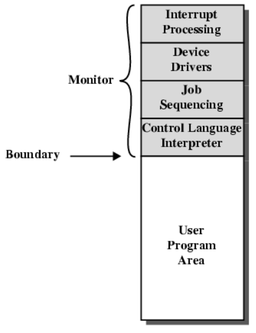
Early Systems

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

Memory Layout for Resident Monitor



13

Job Control Language

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

14

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

15

Multi-programmed Batch Systems

- I/O devices very slow
- When one program is waiting for I/O, another can use the CPU
- Following illustrates the problem:
  - the calculation concerns a program that processes a file of records and performs, on average, 100 processor instructions per record.
    - In this example the computer spends over 96% of its time waiting for I/O devices to finish transferring data.

System utilization example

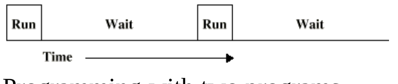
Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

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

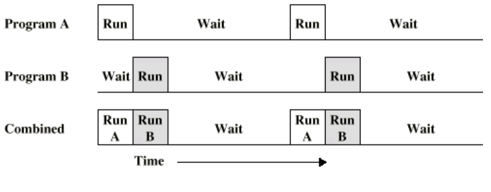
16

Single vs Multi-Programming

- Single program



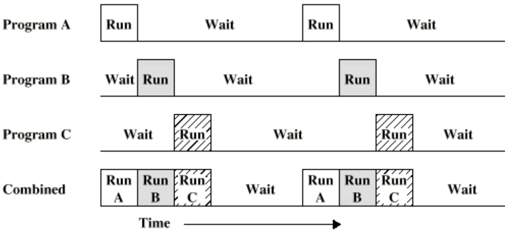
- Multi-Programming with two programs



17

Single vs Multi-Programming

- Multi-Programming with Three Programs



18

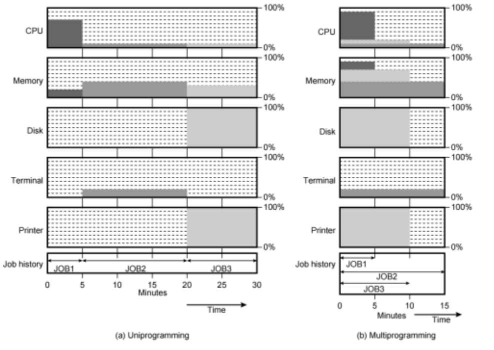
Example- benefits of mutiprogramming

- Consider a computer with 250 MBytes of memory, a disk, a terminal, and a printer.
- The programs JOB1, JOB2, and JOB3 are submitted for execution at the same time with the following 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

- We assume minimal processor requirements for JOB2 and JOB3 and continuous disk and printer use by JOB3.
- For a simple batch environment, these jobs will be executed in sequence

Utilization histograms



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

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

- Scheduling is key to multi-programming
- A process is:
  - A program in execution
  - The “animated spirit” of a program
  - That entity to which a processor is assigned
- 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
- i.e. controls the degree of multi-programming
- Once submitted, 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

- 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

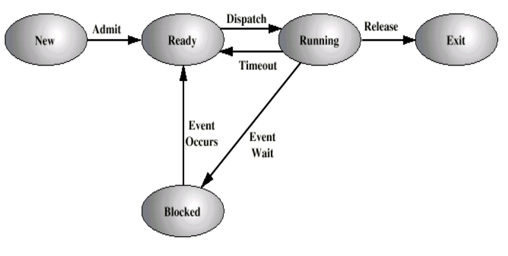
25

Short Term Scheduler

- Also known as Dispatcher, executes frequently and makes the fine grained decisions of which job to execute next
  - i.e. which job actually gets to use the processor in the next time slot
- 5 define states in a process state:
  - New:
    - A program is admitted by the high-level schedular but is not yet ready to execute
  - Ready:
    - The process is ready to execute
  - Running:
    - The prcess is being executed
  - Waiting:
    - The process is suspended, waiting for some system resources
  - Halted:
    - The process has terminated and will be destroyed by the O/S.

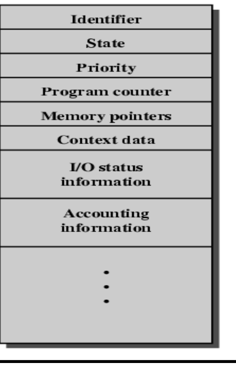
26

Five State Process Model



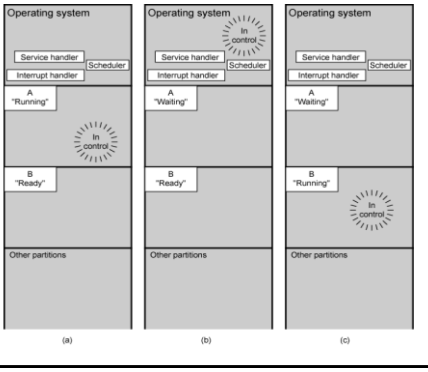
27

Process Control Block



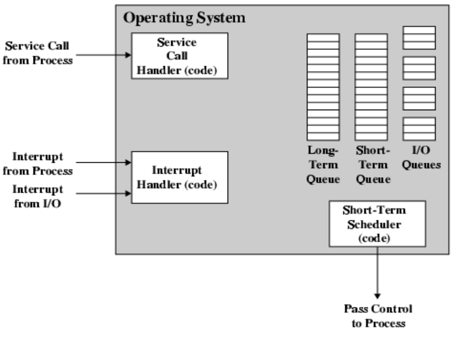
28

Scheduling Example



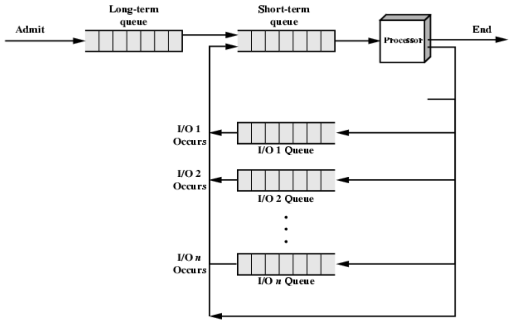
29

Key Elements of O/S for Multiprogramming



30

Queuing Diagram Representation of Process Scheduling



31

Memory Management

- Task of dynamically subdivison of memory
- Effective memory management is vital in a multiprogramming system
- 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

32

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

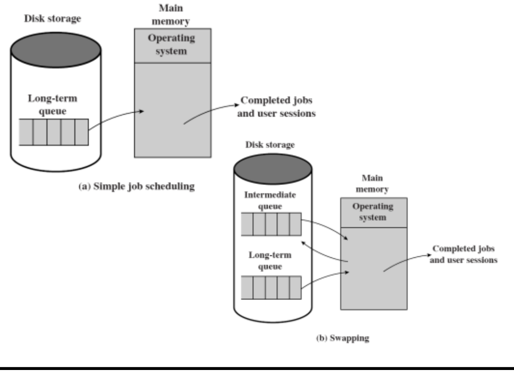
33

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…

34

Use of Swapping



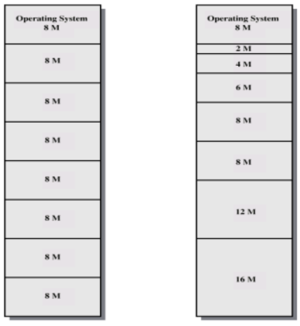
35

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

36

Fixed Partitioning



(a) Equal-size partitions (b) Unequal-size partitions

37

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

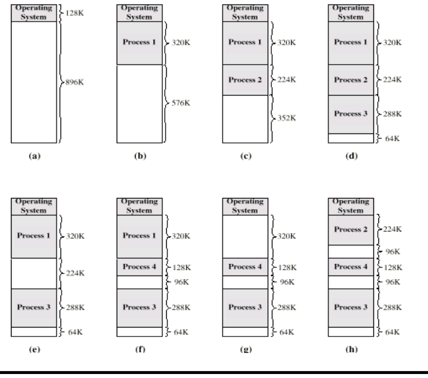
38

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)

39

Effect of Dynamic Partitioning



40

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

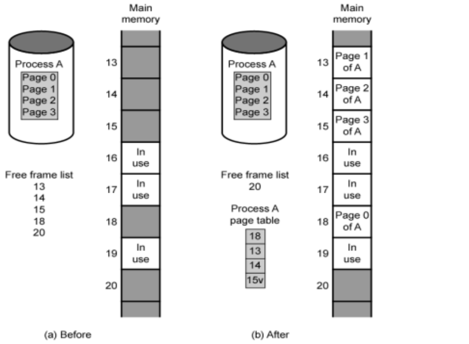
41

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

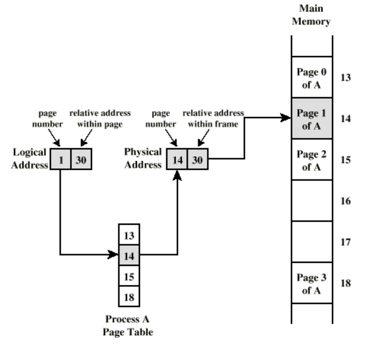
42

Allocation of Free Frames



(a) Before (b) After

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

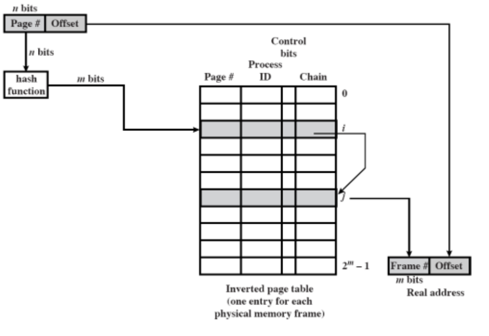
- Solutions
  - Good page replacement algorithms
  - Reduce number of processes running
  - Fit more memory

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

Inverted Page Table Structure

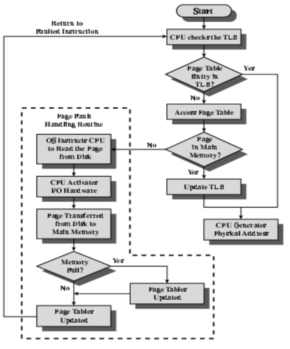


Translation Lookaside Buffer

- Every virtual memory reference causes two physical memory access
  - Fetch page table entry
  - Fetch data
- Use special cache for page table
  - TLB

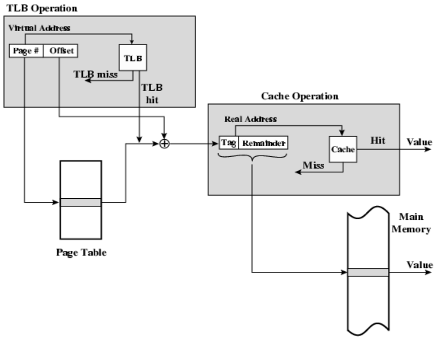
49

TLB Operation



50

TLB and Cache Operation



51

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

52

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

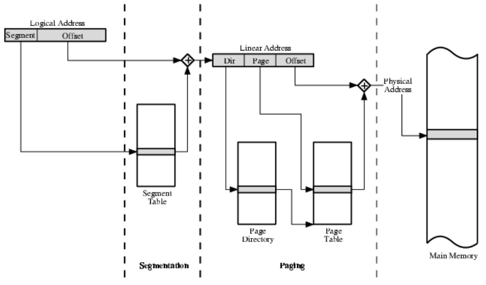
53

Pentium II

- Hardware for segmentation and paging
- Unsegmented unpaged
  - virtual address = physical address
  - Low complexity
  - High performance
- Unsegmented paged
  - Memory viewed as paged linear address space
  - Protection and management via paging
  - Berkeley UNIX
- Segmented unpaged
  - Collection of local address spaces
  - Protection to single byte level
  - Translation table needed is on chip when segment is in memory
- Segmented paged
  - Segmentation used to define logical memory partitions subject to access control
  - Paging manages allocation of memory within partitions
  - Unix System V

54

Pentium II Address Translation Mechanism



55

Pentium II Segmentation

- Each virtual address is 16-bit segment and 32-bit offset
- 2 bits of segment are protection mechanism
- 14 bits specify segment
- Unsegmented virtual memory 2^{32} = 4 Gbytes
- Segmented 2^{46} = 64 terabytes
  - Can be larger – depends on which process is active
  - Half (8K segments of 4 Gbytes) is global
  - Half is local and distinct for each process

56

Pentium II Protection

- Protection bits give 4 levels of privilege
  - 0 most protected, 3 least
  - Use of levels software dependent
  - Usually level 3 for applications, level 1 for O/S and level 0 for kernel (level 2 not used)
  - Level 2 may be used for apps that have internal security e.g. database
  - Some instructions only work in level 0

57

Pentium II Paging

- Segmentation may be disabled
  - In which case linear address space is used
- Two level page table lookup
  - First, page directory
    - 1024 entries max
    - Splits 4G linear memory into 1024 page groups of 4Mbyte
    - Each page table has 1024 entries corresponding to 4Kbyte pages
    - Can use one page directory for all processes, one per process or mixture
    - Page directory for current process always in memory
  - Use TLB holding 32 page table entries
  - Two page sizes available 4k or 4M

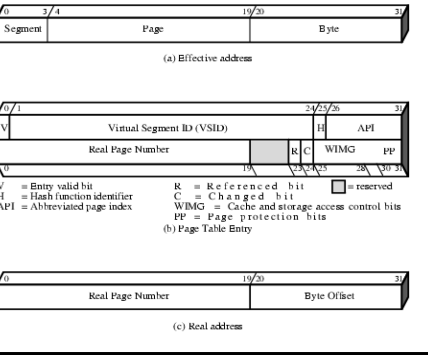
58

PowerPC Memory Management Hardware

- 32 bit – paging with simple segmentation
  - 64 bit paging with more powerful segmentation
- Or, both do block address translation
  - Map 4 large blocks of instructions & 4 of memory to bypass paging
  - e.g. OS tables or graphics frame buffers
- 32 bit effective address
  - 12 bit byte selector
    - =4kbyte pages
  - 16 bit page id
    - 64k pages per segment
  - 4 bits indicate one of 16 segment registers
    - Segment registers under OS control

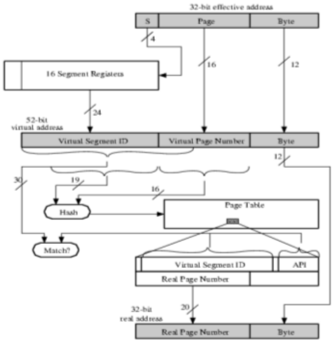
59

PowerPC 32-bit Memory Management Formats



60

PowerPC 32-bit Address Translation



61

62