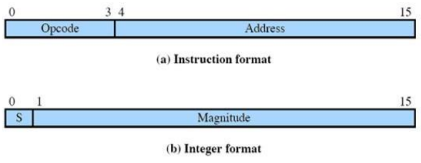
Chapter 1

Operating System: exploits the hardware resources of one or more processors. Provided a set of services to system users. Manages secondary memory and I/O devices.

Microprocessor: Invention that brought about desktop and handheld computing. Processor on a single chip. Fastest general-purpose processor. Multiprocessors. Each chip (socket) contains multiple processors (cores).

Main memory: Referred to as real memory or primary memory.



Formula:
Opcodes = 2^ #bits opcode
Mem size = #pos * size of a pos
pos = 2^#bits of address
Size of a pos = 16 bits = 2 bytes.
HEX notation
Mem range = 000 - FFF Data range = 0000 - FFFF
Octal notation
Mem range = 000 – 777 Data range = 0000 - 777

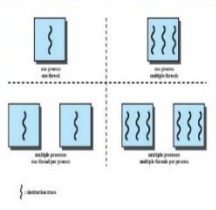
Processes:
The unit or resource allocation and a unit of protection
A virtual address space that holds the process image
Protected access to:
Processors
Other processes
Files
I/O resources

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.



Interrupts: Interrupt the normal sequencing of the processor. Provided to improve processor utilization.

Multiple Interrupts: Two approaches -> disable interrupts while an interrupt is being a processed ->use a priority scheme. (Sequential or Nested.)

Memory Hierarchy: constraints (amount, speed, expense). Must be able to keep up with processor. Cost of memory must be reasonable in a relationship to the other components.

Principle of Locality: memory references by the processor tend to cluster. Data is organized so that percentage of accesses to each successively lower level is substantially less than that of the level above. Can be applied across more than two levels of memory.

Cache memory: Interacts with other memory management hardware. Processor must access memory at least once per instruction cycle. Processor execution is limited by memory cycle time. Exploit the principle of locality with a small fast memory.

Cache principles: Contains a copy of a portion of main memory. Processor first checks cache. If not found, a block of memory is read into cache. Because of locality of reference, it is likely that many of the future memory references will be to other bytes in the block.

Mapping Function: Determines which cache location the block will occupy.

**LRU – Effective strategy is to replace a block that has been in the cache the longest with no references to it. Hardware mechanisms are needed to identify the least recently used block. Chooses which block to replace when a new block is to be loaded into the cache.

I/O Techniques: Programmed I/O, Interrupt Driven I/O, Direct Memory Access (DMA)

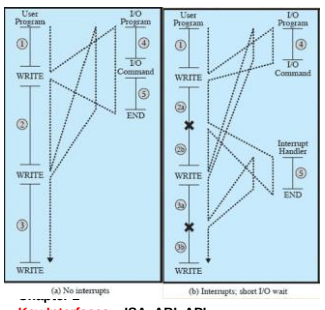
Programmed I/O: I/O module performs the requested action then sets the appropriate bits in the I/O status register. Processor periodically checks the status of the I/O module until it determines the instruction is complete. With programmed I/O the performance level of the entire system is severely degraded.

Interrupt Driven I/O drawbacks: Transfer rate is limited by the speed with which the processor can test and service a device. The processor is tied up in managing an I/O transfer.

Direct memory access: transfer the entire block of data directly to and from memory without going through the processor. Processor is involved only at the beginning and end of the transfer. Processor executes more slowly during a transfer when processor access to the bus is required. More efficient than interrupt-driven or programmed I/O.

Symmetric Multiprocessors: Two or more similar processors of comparable capability. Processors share the same main memory and are interconnected by a bus or other internal connection scheme. Processors share access to I/O devices. All processors can perform the same functions. The system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels.

SMP Advantages: Performance (a system with multiple processors will yield greater performance if work can be done in parallel.) **Scaling** (vendors can offer a range of products with different price and performance characteristics.) **Availability** (the failure of a single processor does not halt the machine.) **Incremental growth** (an additional processor can be added to enhance performance)



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 $= \frac{1}{31} = 0.032 = 3.2\%$

Figure 2.4 System Utilization Example

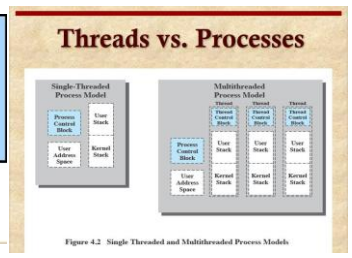


Figure 4.2 Single Threaded and Multithreaded Process Models

Average access time = Hit Ratio * TL1 + Miss ratio * (TL2+TL1) #blocks in main memory = 2^n / K

Chapter 2

Key Interfaces – ISA, ABI, API

**Role of OS – A computer is a set of resources for the movement, storage, and processing of data. The OS is responsible for managing these resources.

**OS a software – Functions in the same way as ordinary computer software. Program, or suite of programs, executed by the processor. Frequently relinquishes control and must depend on the processor to allow it to regain control.

Serial Processing – Earliest Computers (No OS. Programmers interacted directly with the computer hardware. Computers ran from a console with display lights, toggle, switches, some form of input device and a printer. Users have access to the computers in "series". **Problems** (Scheduling: most installations used a hard copy sign-up sheet to reserve computer time. Time allocations could run short or long, resulting in wasted computer time. Set up time: a considerable amount of time was spent just on setting up the program to run.

Simple Batch Systems
- **Monitor Point of View** - Monitor controls the sequence of events. *Resident Monitor* is software always in memory. Monitor reads in job and gives control. Job returns control to monitor.
- **Processor Point of View** - Processor executes instruction from the memory containing the monitor. Executes the instructions in the user program until it encounters an ending or error condition. **"***Control is passed to a job"* means processor is fetching and executing instructions in a user program. **"***Control is returned to the monitor"* means that the processor is fetching and executing instructions from the monitor program.

Modes of Operation:
- **User mode**: User program executes in user mode. Certain areas of memory are protected from user access. Certain instructions may not be executed.
- **Kernel Mode**: Monitor executes in kernel mode. Privileged instructions may be executed. Protected areas of memory may be accessed.

Simple Batch System Overhead
- Processor time alternates between execution of user programs and execution of the monitor
- Sacrifices:
  - some main memory is now given over to the monitor.
  - some processor time is consumed by the monitor.
- Despite overhead, the simple batch system improves utilization of the computer.

Time Sharing System - Can be used to handle multiple interactive jobs. Processor 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.

Development of the process
- **Multiprogramming batch operation**: processor is switched among the various programs residing in main memory.
- **Time Sharing**: be responsive to the individual user but be able to support many users simultaneously.
- **Real-time transaction systems**: a number of users are entering queries or updates against a database.

Causes of errors:
- **Improper Synchronization**: a program must wait until the data are available in a buffer. Improper design of the signaling mechanism can result in loss or duplication.
- **Failed Mutual Exclusion**: more than one user or program attempts to make use of a shared resource at the same time. Only one routine at time allowed to perform an update against the file.
- **Nondeterminate program operation**: program execution is interleaved by the processor when memory is shared. The order in which programs are scheduled may affect their outcome.
- **Deadlocks**: it is possible for two or more programs to be hung up waiting for each other. May depend on the chance timing of resource allocation and release.

Process Management: The entire state of the process at any instant is contained in its context. New features can be designed and incorporated into the OS by expanding the context to include any new information needed to support the feature.

Memory Management: The OS have five principal storage management responsibilities: process isolation, automatic allocation and management, support of modular programming, protection and access control, long-term storage.

Virtual memory - A facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available. Conceived to meet the requirement of having multiple user jobs reside in main memory concurrently.

Paging: Allows processes to be comprised of a number of fixed-size blocks, called pages. Program references a word by means of a virtual address. Provides for a dynamic mapping between the virtual address used in the program and a real (or physical) address in main memory.

Information Protection and Security: The nature of the threat that concerns an organization will vary greatly depending on the circumstances. The problem involves controlling access to computer systems and the information stored in them.

Multithreading: Technique in which a process, executing an application, is divided into threads that can run concurrently. **Thread**: dispatchable unit of work. Includes a processor context and its own data area to enable subroutine branching. Executes sequentially and is interruptible. **Process**: a collection of one or more threads and associated system resources. Programmer has greater control over the modularity of the application and the timing of application related events.

Symmetric Multiprocessing: Term that refers to a computer hardware architecture and also to the OS behavior that exploits that architecture. Several processes can run in parallel. Multiple processors are transparent to the user. The OS takes care of scheduling of threads or processes on individual processors and of synchronization among processors.

Chapter 3

OS Management of Application Execution: Resources are made available to multiple applications. The processor is switched among multiple applications so all will appear to be progressing. The Processor and I/O devices can be used efficiently.

Two essential elements of a process are: **Program code** ->which may be shared with other processes that are executing the same program. **A set of data associated with that code** -> when the processor begins to execute the program code, we refer to this executing entity as a process.

Process Elements: identifier, state, priority, program counter, memory pointers, context data, I/O status information, accounting information.

Process states: **Trace** -> the behavior of an individual process by listing the sequence of instructions that execute for that process -> the behavior of the processor can be characterized by showing how the traces of the various processes are interleaved. **Dispatcher** -> small program that switches the processor from one process to another.

Process creation: **Process spawning** -> when the OS creates a process at the explicit request of another process. **Parent process** -> is the original process. **Child process** -> is the new process.

Process Termination -> There must be a means for a process to indicate its completion. -> A batch job should include a HALT instruction or an explicit OS service call for termination. -> For an interactive application, the action of the user will indicate when the process is completed (e.g., log off, quitting an application)

Swapping -> involves moving part of all a process from main memory to disk. -> when none of the processes in main memory is in the Ready state, the OS swaps one of the blocked processes out on to disk into a suspend queue.

Characteristics of a suspended process -> The process is not immediately available for execution -> The process may or may not be waiting on an event -> The process was placed in a suspended state by an agent: either itself, a parent process, or the OS, for the purpose of preventing its execution -> The process may not be removed from this state until the agent explicitly orders the removal.

Memory Tables -> Used to keep track of both main (real) and secondary (virtual) memory -> Processes are maintained on secondary memory using some sort of virtual memory or simple swapping mechanism.
- **Must include** allocation of main memory to processes, allocation of secondary memory to processes, protection attribute of blocks of main or virtual memory. Information needed to manage virtual memory.

I/O Tables -> Used by the OS to manage the I/O devices and channels of the computer system -> At any given time, an I/O device may be available or assigned to a particular process.
- These tables provide information about existence of files, location on secondary memory, current status, other attributes.

Process tables -> must be maintained to manage processes -> There must be some reference to memory I/O, and files, directly or indirectly -> The tables themselves must be accessible by the OS and therefore are subject to memory management.

Process Control Structures -> OS must know where the process is located -> the attribute of the process that are necessary for its management.

Process Location -> A process must include a program or set of programs to be executed -> A process will consist of at least sufficient memory to hold the programs and data of that process ->The execution of a program typically involves a stack that is used to keep track of procedure calls and parameter passing between procedures.

Process Attributes -> Each process has associated with it a number of attributes that are used by the OS for process control -> The collection of program, data, stack, and attributes is referred to as the process image -> Process image location will depend on the memory management scheme being used.

Process Identification -> Each process is assigned a unique numeric identifier -> Many of the tables controlled by the OS may use process identifiers to cross-reference process tables -> Memory tables may be organized to provide a map of main memory with an indication of which process is assigned to each region -> When processes communicate with one another, the process identifier informs the OS of the destination of a particular communication -> When processes are allowed to create other processes, identifiers indicate the parent and descendants of each process.

Processor State Information: **Consists of the contents of processor registers** -> user-visible registers.
-> control and status registers -> stack pointers. **Program status word (PSW)** -> contains condition codes plus other status information -> EFFLAGS register is an example of a PSW used by any OS running on an x86 processor.

Process Control Information: The additional information needed by the OS to control and coordinate the various active processes.

Role of the process control block -> **The most important data structure in an OS** -> contains all of the information about a process that is needed by the OS -> blocks are read and/or modified by virtually every module in the OS -> defines the state of the OS. **Difficulty is not access, but protection** -> a bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected processes -> a design change in the structure or semantics of the process control block could affect a number of modules in the OS.

Modes of Execution:
- **User Mode** => less-privileged mode => user programs typically execute on this mode.
- **System Mode** -> more-privileged mode => also referred to as control mode or kernel mode.
  => kernel of the operating system.

Process creation (step by step): assigns a unique process identifier to the new process, allocates space for the process, initializes the process control block, sets the appropriate linkages, creates or expands other data structures.

System Interrupts: **Interrupt** -> Due to some sort of event that is external and independent of the currently running process (clock interrupt, I/O interrupt, memory interrupt) -> Time slice (the maximum amount of time that a process can execute before being interrupted). **Trap** -> An error or exception condition generated within the currently running process -> OS determine if the condition is fatal.

If no interrupts are pending the processor -> proceeds to the fetch stage and fetches the next instruction of the current program in the current process.

If an interrupt is pending the processor -> sets the program counter to the starting address of an interrupt handler program -> switches from user mode to kernel mode so that the interrupt processing code may include privileged instructions

Change of process state (step by step): save the context of the processor, update the process control block of the process currently in the running state, move the process control block of this process to the appropriate queue, select another process for execution, update the process control block of the process selected, update memory management data structures, restore the context of the processor to that which existed at the time the selected process was last switched out.

Security issues -> An OS associates a set of privileges with each process -> Typically a process that executes on behalf of a user has the privileges that the OS recognizes for that user -> Highest level of privilege is referred to as administrator, supervisor, or root access -> A key security issue in the design of any OS is to prevent, or at least detect, attempts by a user or a malware from gaining unauthorized privileges on the system and from gaining root access

System access threats: **Intruders** -> often referred to as a hacker or cracker -> objective is to gain access to a system or to increase the range of privileges accessible on a system -> attempts to acquire information that should have been protected. **Malicious software** -> most sophisticated types of threats to computer system -> can be relatively harmless or very damaging.

Countermeasures access control: Implements a security policy that specifies who or what may have access to each specific system resource and the type of access that is permitted in each instance -> Mediates between a user and system resources -> A security administrator maintains an authorization database -> An auditing function monitors and keeps a record of user accesses to system resources **Countermeasures Firewall**: **A dedicated computer that** -> interfaces with computers outside a network -> has special security precautions built into it to protect sensitive files on computers within the network. **Design goals of a firewall** -> all traffic must pass through the firewall -> only authorized traffic will be allowed to pass -> immune to penetration.

Chapter 4

Processes 2 characteristics
- **Resource Ownership**: process includes a virtual address space to hold the processimage.
- **Scheduling/Execution**: Follows an execution path that may be interleaved with other processes.

Processes and threads -> The unit of dispatching is referred to as a thread or lightweight process -> The unit of resource ownership is referred to as a process or task -> Multithreading - The ability of an OS to support multiple, concurrent paths of execution within a single process.
**Single Threaded Approach: A single thread of execution per process, in which the concept of a thread is not recognized, is referred to as a single-threaded approach.

One or more threads in a process: **Each thread has** -> an execution -> saved thread context when not running -> an execution stack -> some per-thread static storage for local variables -> access to the memory and resources of its process (all threads of a process share this)
Benefits of Threads -> Takes less time to create a new thread than a process -> less time to terminate a thread than a process -> switching between two threads takes less time than switching between processes -> Threads enhance efficiency in communication between programs.
Thread use in a single-user system -> Foreground and background work -> Asynchronous processing.
-> Speed of execution -> Modular program structure

Suspending Threads -> suspending a process involves suspending all threads of the process -> termination of a process terminates all threads within the process.
Thread execution states: **Key states** (Running, ready, blocked), Thread operations associated with a change in thread state are (Spawn, block, unblock, finish)
Thread Synchronization: It is necessary to synchronize the activities of the various threads -> all threads of a process share the same address space and other resources -> any alteration of a resource by one thread affects the other threads in the same process.
Types of threads: User level thread (ULT) and Kernel level thread (KLT)
User-level threads -> all thread management is done by the application -> the kernel is not aware of the existence of threads.
Advantages of ULTs: Thread switching does not require kernel mode privileges -> scheduling can be application specific -> ULTs can run on any OS.
Disadvantages of ULTs: In a typical OS many system calls are blocking, as a result, when a ULT executes a system call, not only is that thread blocked, but all of the threads within the process are blocked -> In a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing.
Overcoming ULT disadvantages: **Jacketing** -> converts a blocking system call into a non-blocking system call -> writes an application as multiple process rather than multiple threads.
Kernel-Level threads (KLTs) -> thread management is done by kernel -> no thread management is done by the application -> Windows is an example of this approach.
Advantages of KLTs -> The Kernel can simultaneously schedule multiple threads from the same process on multiple processors -> If one thread in a process is blocked, the kernel can schedule another thread of the same process -> Kernel routines can be multithreaded.
Disadvantages of KLTs: The transfer of control from one thread to another within the same process requires a mode switch to the kernel.
Combine approaches -> thread creation is done in the user space -> bulk of scheduling and synchronization of threads is by the application -> Solaris is an example.

Exam Review

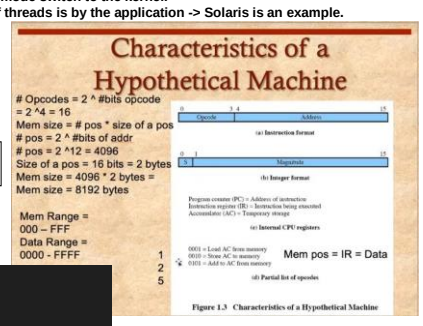
1. Calculate the following parameters of a hypothetical computer system with thesefeatures:
   a. Octal notation
   b. IR = Opcode + Mem Address
   c. IR = 12 bits
   d. PC = 3 octal digits
   e. Mem word size = Data (unsigned integer) = IR

Number of different Opcodes: 2^3 = 6
Mem size in bits: 2^9 * 12 = 6144
Mem range: 000 -777
Data range: 0000-7777

0	2 | 3	11

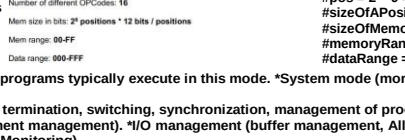
Libraries: pthread.h, stdio.h, stdlib.h, sys/types.h, sys/wait.h Declaration: pid_t id;
Threads Syntax:
if(pthread_create(&id[i], NULL, name_of_function,&dynamic_or_static_arrName[i]))
  for(int i = 0; i < size; i++)
    pthread_join(id[i], NULL)
  struct struct_name* your_name = (struct* struct_name) name_of_param;
  ***
Libradies: iostream, unistd.h, sys/wait.h
Fork Syntax:
  pid_t pid;
  pid = fork();
  if (pid == 0) Means that this is child process has no child processes
  use _exit(0) and wait(0) so that the fork won't have zombie processes
PC = memory address, 1 octal digit = 3 bits,
3 octal digits = 9 bits for mem addr, 12 - 9 = 3 bits for opcode. 2^3 = 8
mem size in bits = # positions * size of position,
mem word size = IR so each position is 12 bits.
So we need # of positions.
2^9 (9 bits for memory address size) = 512 * 12 = 6144
every fork needs to be paired with a wait (outside the loop you did fork)

#opcodes = 2^4=16 #pos = 2^12=4096
#sizeOfPos = 16bits=2bytes #sizeOfMemory=#pos * sizeOfPos
#memoryRange = 000-FFF #DataRange=0000-FFFF




```
if (i == 1)
{
    pid = fork();
    if (pid == 0)
    {
        cout << "I am a granchild process from child process " << i << end
        _exit(0);
    }
    wait(nullptr);
}
```

```
for (int j = 0; j < 2; j++)
{
    if ((pid=fork()) == 0)
    {
        std::cout << "I am a granchild process from child process " << i << std::endl;
        _exit(0);
    }
    wait(nullptr);
}
```



#PC = Octal Digits = log2(8)= 3 bits * 3 = 9 bits
-PC is length of mem address!!!!
#IR = opcode + address = Data = memOfAPosition = 12
#pos = 2 ^ 9 = 512 bits
#sizeOfAPosition = 12 bits
#sizeOfMemory = #pos * #sizeOfAPosition = 512 bits * 12
#memoryRange = 000-777
#dataRange = 0000-7777

Process Control (continued)

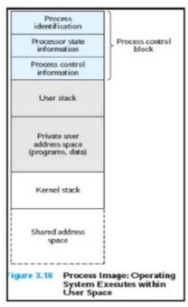
Modes of Execution: *User mode (less-privileged mode) -user programs typically execute in this mode. *System mode (more privileged) - also referred to as control mode or kernel mode - kernel of the OS.
KERNEL functions - *Process management (process creation, termination, switching, synchronization, management of proc. control block). *Memory management (allocation of address space to process, swapping, page/segment management). *I/O management (buffer management, Allocation of I/O channels and devices to process). *Support functions (interrupt handling, Accounting, Monitoring).
Process creation (step-by-step): 1) OS assigns a unique process ID to the new process 2) allocates space for the process. 3) initializes the process control block. 4) sets the appropriate linkages 5)creates or expands other data structures.
Process Switching - a process switch may occur anytime that the OS has gained control from the currently running process. Events giving OS control are *interrupt(reaction an asynchronous external event)
*Trap(Halding of an error or an exception condition) *Supervisor call (call to an operating system function)
System Interrupt: *due to some sort of event that is external to and independent of the currently running process. *Clock interrupt *I/O interrupt *memory fault *time slice - the max amount of time that a process can execute before being interrupted.
Trap: *an error or exception condition generated within the currently running process *OS determines if the condition is fatal - moved to the exit state and a process switch occurs *action will depend on the nature of the error.
If no interrupts are pending the Processor: 1) proceeds to the fetch stage and fetches the next instruction of the current program in the current process.
If interrupt is pending: 1) sets the program counter to the starting address of an interrupt handler program. 2) switch from user mode to kernel mode so that the interrupt processing code may include privileged instructions.
Change Of Processor State: 1) save the context of the proc 2) update the process control block of the process current running. 3) move the process control block of this process to the appropriate que. 4) select another process for exe. 5) update the process control block of the process selected. 6) update memory management data structures. 7) restore the context of the processor to that which existed at the time the selected process was last switched out.
Security Issues: * An OS associates a set of privileges with each process
*Typically a process that exe on behalf of a user has the privileges that the OS recognizes for that user. *Highest level of privilege is referred to as admin, supervisor, root.
System Access Threats: *Intruders (hacker) *Malicious Software. Intrusion Detection counter measure: Intrusion detection system(IDS_ comprises of three logical components 1)sensors 2)analyzers 3)user interface *IDS are designed to detect human intruder behavior.
Authentication Countermeasure: consists of two steps 1) Identification
2) verification
Access Control Countermeasure: *implements a security policy that specifies who or what may have access to each specific system resource and the type of access that is permitted in each instance.
*Mediates between a user and system resource. *A security admin
#opcodes = 2^4 = 16 #pos = 2^12 = 4096 #sizeofMemory = #pos * SizeOfPos
Hexadecimal: #memoryRange 000-FFF #dataRange 0000-FFFF
Octal #memoryRange 000-777 #dataRange 0000-7777

Processes and Threads - *have two characteristics. 1) Resource ownership - process includes a virtual address space to hold the process image. *The OS performs a protection function to prevent unwanted interference between the processes with respect to resources. 2) Schedule/- Execution *a process has an execution state (running, ready, etc.)and a dispatching priority and is scheduled and dispatched by the OS.
Threads: *The unit of dispatching is referred to as thread or lightweight process *unit of resource ownership is referred to as a process or task.
*In an OS that supports threads, scheduling and dispatching is done on a thread basis.
*Most of the state information dealing with execution is maintained in thread-level data structures.
Multithreading - the ability of an OS to support multiple, concurrent paths of execution within a single process.
Single Thread Approach: * a single thread of execution per process, in which the concept of a thread is not recognized, is referred to as a single-threaded approach. *MS-DOS is an example. Multithreaded approaches: *A java run-time environment is an example of a system of one process with multiple threads.
Processes: *The unit of resource allocation and a unit of protection. *A virtual address space that holds the process image. *Protected access to: -processors, -other processes, -files, -I/O resources.
One or more threads in a process: [each thread has]: *an execution that's (running, ready, etc.),
*Saved thread context when not running, *an execution stack, *some per-thread static storage for local variables, *access to the memory and resources of its process(all threads of a process share this).
Benefits of Threads: 1) takes less time to create a new thread vs. new process.
2) less time to terminate a thread than a process. 3) switching between threads takes less time than switching between processes. 4) threads enhance efficiency in communication between programs.
Thread use in a single-user system: *foreground and background work,
*Asynchronous processing, *speed of execution, *modular program structure. Suspending a process/thread: *involves suspending all threads of the process.
*Termination of a process terminates all threads within the process.
Thread execution state: *key states(running, ready, blocked), *Thread operations associated with a change in thread state are: 1)spawn 2) block 3) unblock 4) finish. thread synchronization: *It is necessary to synchronize the activities of the various threads. *All threads of a process share the same address space and other resources. *Any alternation of a resource by one thread affects the other threads in the same process.
Types of threads: 1) Upper-level thread(ULT) 2)Kernel level thread(KLT)
ULT Upper-level threads: *all thread management is done by the application. *The kernel is not aware of the existence of threads.
ULT Advantages: *thread switching does not require kernel mode privileges,
*Scheduling can be application specific, *ULTs can run on any OS.
ULT Disadvantages: *in a typical OS many system calls are blocking -as a result, when a ULT executes a system call, not only is that thread blocked, but all of the threads within the process are blocked.
*In pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing.
Overcoming ULT disadvantages: *Jacketing -converts a blocking system call into a non-blocking system call. *Writing an application as multiple processes rather than multiple threads.
Kernel-Level Threads(KLTs): *Thread management is done by the kernel. *No thread management is done by the application., *Windows is an example of this approach.
KLT Advantages: *The kernel can simultaneously schedule multiple threads from the same process on multiple processors. *If one thread in a process is blocked, the kernel can schedule another thread of the same process. *Kernel routines can be multithreaded.
KLT Disadvantages: The transfer of control from one thread to another within the same process requires a mode switch to the kernel.
Combined Approaches: *Thread creation is done in the user space. *Bulk of scheduling and synchronization of threads is by the application, *Solaris is an example.
Applications that Benefit: *Multithreaded native applications -characterized by having a small number of highly threaded processes. *Multiprocessor applications - characterized by the presence of many single-threaded process. *Java Applications., *Multistranded applications – multiple instances of the application in parallel

```
struct operation *pos_ptr = (struct operation *)pos_void_ptr;
if(pthread_create(&tid[i], NULL, calculator, &operations[i]))
```

***Operating system** -exploits hardware resources of one or more processors, most complex pieces of software -provides a set of services to system users, manages second. mem and I/O dev.
*Pc - holds address of the instruction to be fetched next (incr. after each fetch) PROCESSOR IR - instruction being executed. MAR
- MBR -I/O AR - I/O BR -
*AC (accumulator) - temporary storage
microprocessor - fastest general-purpose proc. proc on single chip, multiprocessor, each chip cont. cores **GPU** - provide efficient computation or arrays, physic simulation, comp. on large spreadsheet.
*Basic instruction cycle - start -> fetch inst. -> execute inst. (repeat if needed) -> halt
***Interrupts** - interrupt the normal sequencing of a proc provided to improve proc utilization (i/o devices are slower than the proc)
*Classes of interrupt(Program, Timer, I/o, hardware failure)
***Single interrupt** process - dev. controller/sys hard issues inter -> proc finishes current instruction -> proc ack. interrupt -> proc pushes PSW and PC onto control stack ->proc loads new PC value from inter -> save remainder of proc state info -> process interrupt-> restore proc state info -> restore old PSW and PC
***Multiple interrupts** - can be done sequentially or nested
***MEMORY** HIERARCHY major constraints (amount, speed, expense), memory must keep up with proc, cost of mem must be reasonable to other components
*Down pyramid (decreasing cost per bit, increasing cap., increasing access time, decreasing. freq. of acc. to mem by proc.)
***PRINCIPLE OF LOCALITY** - 1) memory references by the proc. tend to cluster 2) data is org. so that the percentage of accesses to each successively lower level is less than that of the level above 3) can be applied to more than 2 levels'
***CACHE** 1) invisible to OS. 2) interacts with mem management. hardware. 3) Proc. must access mem. at least once per instr. cycle. 4) exploits principle of locality with small, fast mem.
***Cache Princ.** 1) contains copy of portion of main mem. 2) Proc first checks cache 3) not found? block of mem is read into cache. 4) locality of ref- it is likely that many of future mem. references will be other bytes in block.
MAPPING FUNCTION - Determines which cache location the block will occupy
*Replacement Algorithm - Least recently used (LRU) Algo (effective Strat to replace a block that has been in the cache the longest with no refer. *Hardware mechanisms are needed to identify the least recently used block *chooses which block to replace when a new block is to be loaded into cache)
*I/O Techniques - When the proc encounters an instr. relating to I/O, its exe that instr. by issuing a comm to the appropriate I/O module
***PROGRAMMED I/O** - 1) the I/O module performs the requested action then sets the appropriate bits in the I/O status reg. 2) The Proc periodically checks the status of the I/O module until it determines the instructions. is complete. 3)With programmed i/O the performance level of the entire system is severely degraded
Interrupt Driven I/O drawbacks - transfer rate is limited by the speed with which the proc can test/service a device. 2)The proc is tied up in managing an I/O transf. - number of instructions. must be exe for each I/O transfer.
DIRECT MEMORY ACCESS (DMA) - performed by a separate module on the sys bus or incorporated into an I/O module.
1) transfer the entire block of data directly to and from mem without going through the proc
   -proc is involved only at the beg. and end of transfer
   - proc exe more slowly during a transfer when proc access to the bus is req.
2) more efficient than interrupt driven I/O
3) **Symmetric Muti-Proc (SMP)** - stand-alone computer system with-two or more similar processors of comparable capable.
   -processors share the same main mem and are interconnected by bus
   -proc share access to I/O devices
   -all processors can perform the same functions
   -system is controlled by integrated

OS that provides interaction between processes and their programs at the job, task, file, data element levels.
Multi-core comp (chip multi-proc) - combines two or more proc(cores) on a single silicon
   -each core consists of all components of an independent. proc.
-multicore chips include L2/L3 cache.
OPERATING SYSTEM - program that controls the execution of application programs, interface between application and hardware.
*Services OS - program development, program exe, access i/o devices, controlled access to files, system access, error detection and response.



OPERATING SYSTEM - program that controls the execution of application programs, interface between application and hardware.
*Services OS - program development, program exe, access i/o devices, controlled access to files, system access, error detection and response.
Key Interfaces *Instruction set arch. (ISA), application binary interface. (ABI) Application programming interface(API)
Role of OS - computer is a set of resources for the movement, storage, and proc. of data, the OS is responsible for managing these resources.
Operating system as software 1) functions in the same way as ordinary comp software. 2) program, or suite of programs, executed by the proc. 3) frequently relinquishes control and must depend on the proc. to allow it to regain control.
Serial processing - (earliest computers, no OS, programs interacted directly with hardware, ran with display lights, toggle switches, printer, users had to interact with the computer in 'series')PROBLEMS: scheduling - time allocations could run short or long, wasted computer time. set up-time - a considerable amount of time was spent on just setting up the program to run.
Simple Batch system - early computers were expensive, important to maximize proc utilization. *Monitor - user no longer has direct access to processor., job is submitting to computer operator who batches them together and places them on an input device. Program branches back to the monitor when finished.
Monitor Point of view - monitor controls the sequence of events, *resident monitor software always in memory, *monitor reads in jobs and gives control, * job returns control to monitor.
Processor POV - *Proc exe instructions from the mem containing the monitor, *Exe the instructions in the user program until it encounters an ending or ERR condition., *"Control is passed to a job" means proc is fetching and exe instructions in a user program., *"control is returned to the monitor" means that the proc is fetching and exe instructions from the monitor program.
Simple Batch System Overhead - *Proc alternates between execution of user program and execution of the monitor.
***Sacrifices**: 1) some main mem is now given over to the monitor, 2) some proc time is consumed by the monitor. *Despite overhead, the simple batch system improves utilization of the comp. **Nonprogramming** - The proc spends a certain amount of time exe, until it reaches an I/O instruction; it must then wait until that I/O instruction concludes before proceeding.
Multiprogramming - *there must be enough memory to hold the OS (resident monitor) and one user program, * When one job needs to wait for I/O, the processor cans witch to the other job, which is likely not waiting for I/O *Also known as multitasking, *mem is expanded to hold three, four, or more programs and switch among all of them.
Time-sharing system- *can be used to handle multiple interactive jobs, *Proc time is shared among multiple users. *Multiple users simultaneously access the system through terminals, with the OS interleaving the exe of each user program in a short burst or quantum of computation.
Process - fundamental to the structure of operating systems.
Development of the process - Three major lines of computer system dev.
created problems in timing and synchronization that contributed to the dev.
Causes of error: 1) improper synchronization - a program must wait until the data are available in a buffer, improper design of the signaling mechanism can result in loss or duplication. 2) failed mutual exclusion - more than one user or program attempts to make user of a shared resource at the same time. *Only one routine at a time allowed to perform an update against the file. 3) nondeterminate program operation - program exe is interleaved by the proc when mem is shared. *The order in which programs are scheduled may affect their outcome 4) Deadlocks - it is possible for two or more programs to be hung up waiting for each other. *May depend on the chance timing of resource allocation and release.
Components of a process: *a process contains three components 1) an executable program 2) the associated data needed by the program (variables, work space, buffers) 3) The execution context 'Process State' of the program **The execution context:** *is the internal data by which the OS is able to supervise and control the process, * includes the contents of the various process registers, *includes information such as the priority of the process and whether the process is waiting for the completion of a particular I/O event.
Process management: *The entire state of the process at any instant is contained in its context, * New features can be designed and incorporated into the OS by expanding the context to include any new information needed to support the feature.
Memory Management: the OS has five principal storage management responsibilities
Virtual memory: *a facility that allows programs to address memory from logical point of view, without regard to the amount of main memory physically available. *Conceived to meet the requirement of having multiple user jobs reside in main memory concurrently.
Paging: *Allows processes to be comprised of a number of fixed-size blocks. *Program references a word by means of virtual address (consists of a page number and an offset with the page), (each page may be located anywhere in main mem.). *Provides for a dynamic mapping between the virtual address used in the program and a real address in main mem.
OS Management of Application Execution *Resource made available to multiple applications *The proc is switched among multiple applications so all will appear to be progress elements *The proc and I/O devices can be used efficiently.
Process Control block *Contains the process elements *It is possible to interrupt a running process and later resume execution as if the interrupt had not occurred. *Key tool that allows support for multiple processes
Two state process model: state 1) running state2) not running.
Reasons for Process creation: 1)new batch job 2) interactive logon 3) created by OS to provide a service 4) spawned by existing process.
Process termination: *there must be a means for a process to indicate its completion. *a batch job should include HALT instruction or an explicit OS service call for termination. *For an interactive application, the action of the user will indicate when the process is completed (e.g. logoff, quitting an application).
Reasons for process termination: 1)normal completion 2)time limit exceeding 3) memory unav 4) bounds viol 5)protection err 6) arithmetic err 7) I/O failure 8) parent termination
Suspended Processes: *Swapping - involves moving part of all of a process from main memory to disk *when none of the processes in main memory is in the ready state, the OS swaps on of the blocked process out on to disk into a suspend queue.
Characteristics of a Suspended Process: *the process is not immediately available for exe. *The process may or may not be waiting on an event. *The process was placed in a suspended state by an agent: either itself, a parent process, or the OS, for the purpose of preventing its exe. *The process may not be removed from this state until the agent explicitly orders the removal.
Reasons for suspended process: 1) swapping, 2) OS may suspend a background or utility 3) interactive user request 4) timing, may be executed periodically 5) parent process request, parent may wish to suspend exe of a descendent to modify.
OS CONTROL TABLES
Memory Tables: *used to keep track of both main and secondary memory. *Processes are maintained on secondary memory using some sort of virtual memory or simple swapping mechanism. Must include *allocation of main mem to process *allocation secondary mem to process *protection attributes of blocks of main or virtual mem. *Information needed to manage virtual mem.
I/O tables: *used by the OS to manage the I/O devices and channels of the computer system. *At any given time, and I/O device may be available or assigned to a particular process.
File Tables: *information may be maintained and used by a file management system in which case the OS has little or no knowledge of files. *In other OS, much of the detail of file management is managed by the OS itself. *Existence of files *location of secondary mem *current status
Process Tables: Must be maintained to manage processes. *There must be some reference to memory, I/O, and files, directly or indirectly. *The tables themselves must be accessible by the OS and therefore are subject to memory management.
PROCESS CONTROL STRUCTURE *OS must know *where the process is located *The attributes of the process that are necessary for its management.
Process Control Location: Process location - *process must include a program to be exe, *Process will consist of at least sufficient memory to hold the programs and data of the process. *The execution of a program typically involves a stack that is used to keep track of procedure calls and parameter passing between procedures.
Process Attributes: each process has associated with it a number of attributes controlled by the OS. *The collection of programs, data, stack, attributes are referred to as process image. (Location depends on memory management scheme being used.)
Process Image: 1) user data (modifiable part of user space, stack, etc.) 2) User program 3) stack(each process has one or more LIFO stacks associated. stack is used to store parameters and calling addresses for system calls.) 4) Process control block (data needed by the OS to control the process.
Process Identification: *each process is assigned a unique numeric ID, or there must be a mapping that allows the OS to locate the appropriate tables based on the process ID. *many of the tables controlled by the OS may use process ID to cross-reference process tables. *Memory tables may be organized to provide a map of main memory. *When processes communicate with each other the process ID informs the OS of the destination of a particular communication. *When processes are allowed to create other processes, identifiers indicate the parent and descendants.
Processor State information: Consists of the contents of the process registers *user-visible registers *control and status registers *stack pointers
Program Status Word (PSW): contains condition codes plus other status information. *EFLAGS register is an example of a PSW used by any OS running on an x86 proc.
Process Control Information: the additional info needs by the OS to control and coordinate the various active processes.
Process Control Block: *most important data structure in an OS *contains all of the information about a process that is needed by the OS. *Blocks are read and/or modified by virtually every module in the OS. *Defines the state of the OS. *Difficulty is not access but protection. *a bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected process. *A design change in the structure or semantics of the Process control block could affect a number of modules in the OS.

New interrupt will be paused if processing a current interrupt to prevent kernel overflow. Either disable interrupts or priority scheme.**Main** Memory is volatile, & content lost when comp. shut down #OpCodes = 2^#Bits Opcodes, MemSize = #Pos * Size of Pos, #Pos = 2^#bits of addr, Data Range (1 less digit for mem range) 0000-FFFF (7777 of octal)
Memory must be able to keep up with processor. Cost must also be reasonable. Faster access time = greater cost per bit. Greater capacity

= smaller cost per bit but also slower access speed.

Going down hierarchy decreases cost per bit, increase capacity &access time & decreases frequency of access to memory by CPU.Avg access time = T1 * hit Ratio + (T2+T1) * miss Ratio
Principle of Locality is when memory references by CPU tend to cluster, data organized so % of access to lower levels < upper levels. **Cache memory** invisible to OS, interacts with other memory mgmt. hardware. CPU must access memory at least once/instruction. CPU execution is limited by memory cycle time. Exploit principal locality w/small, fast memory. CPU first checks cache. LRU effective strategy to replace block least used.
Programmed IO performance level of system degraded. CPU periodically checks status IO module until complete. IO modulepreforms requested action then sets appropriate bits.
Serial processing was no OS, computers ran from console & access to computer in "series". Problems with scheduling & setup time. Simple batch systems maximized CPU utilization, now have monitor, no longer direct access to CPU. User mode & Kernel modes of operation. **Multiprogramed batch** systems has CPU often idle as IO devices are slow. Multiprogramming (multitasking) when memory is expanded to gold 3,4 or more programs & switch between them. Maximize CPU use& JCL commands provided with job.
Time sharing systems used to handle multiple interactive jobs, CPUtime shared, minimize response time & commands entered at the terminal.
Process defined as program in execution, instance of running program, entity that can be assigned to & executed on processor, unitof activity characterized by single sequential thread of execution, current state & associated set of system resources.
Causes of errors can be improper synchronization, nondeterminateprogram operation, failed mutual exclusion and deadlocks.
Virtual memory allows programs to address memory from logical point of view w/o regard to amount main memory available. A type ofmemory mgmt. technique using HDD.
Microkernel architecture assigns few essential functions to kernel: address spaces, interposes communication (IPC) & basic scheduling.This simplifies implementation, provides flexibility & well suited to distributed environment.
Symmetric multiprocessing refers to computer hardware architecture& OS behavior that exploits it. Several processes can run parallel. OS takes care of scheduling threads & processes in individuals CPU's. failure does not halt system; performance can be enhanced, and it canscale depending on CPU count.
Distributed OS provides illusion single main memory space, unifiedaccess facilities. State of the art for distributed OS lags behind uniprocessor & SMP OS. Object Oriented Design used for adding modular extensions to small kernel, enables programmers to customize OS w/o disrupting system integrity.

Information Protection and Security: the nature of the threat that concerns an organization will vary greatly depending on the computer systems and the information stored in them. Scheduling and Resource Management: *key responsibility of an OS is managing resources *Resource allocation policies must consider
Different Architectural Approaches: *Demands on operating systems require new ways of organizing the OS
Microkernel Architecture: *Assigns only a few essential functions to the kernel:
Multithreading *technique in which a process, executing and application is divided into threads that can run concurrently
Symmetric Multiprocessing (SMP): *term that refers to a computer hardware architecture, and also to the OS behavior that exploits the architecture *several processes can run in parallel. *Multiple processors are transparent to the user *these processors share the same main memory and I/O facilities. *All processors can perform the same functions *The OS takes care of scheduling of threads or processes on individual processors and of synchronization among processors
OS Design: [Distributed OS] *provides the illusion of 1) single main memory space 2) single secondary memory space 3) unified access that of uniprocessor and SMP operating systems [Object-Oriented Design] *used for adding modular extensions to a small kernel *enables programmers to customize an operating system without disrupting system integrity *eases the development of distributed tools and full-blown distributed operating systems
Virtual Machine and Virtualization: *enables a PC or server to of a single OS *a machine can host multiple operating systems on a single platform *host operating system can support a number of virtual machines *each has the characteristics of a particular and in some versions of virtualization, characteristics of a particular hardware

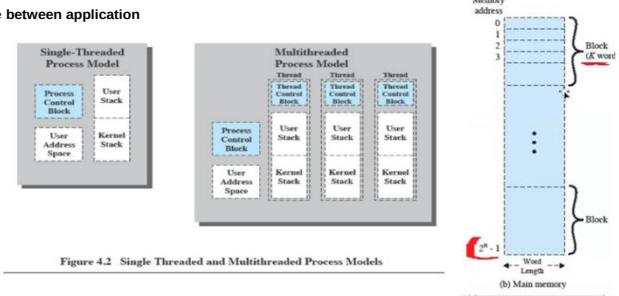


Figure 4.2 Single Threaded and Multithreaded Process Models

Blocks in Main Memory = 2 ^ n / K

Threads:Processes	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:N	Combines attributes of M:1 and 1:M cases.	TRIX

Table 4.2 Relationship between Threads and Processes

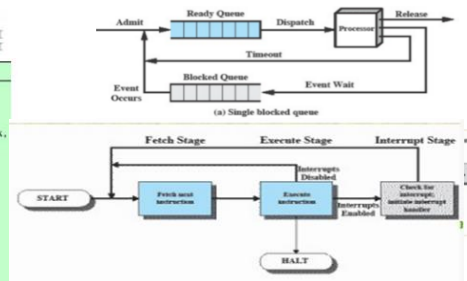
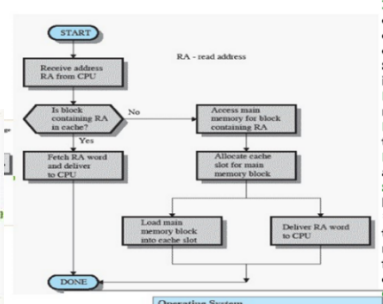


Figure 1.7 Instruction Cycle with Interrupts



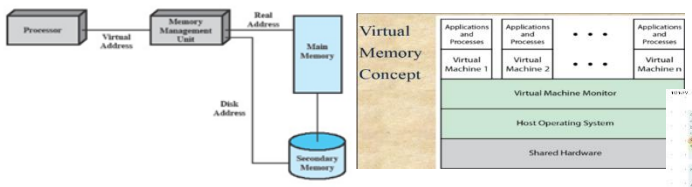
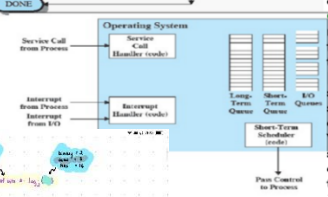
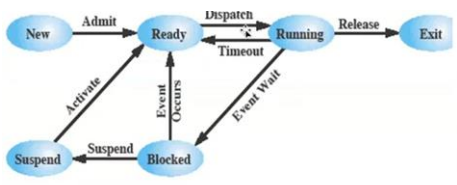
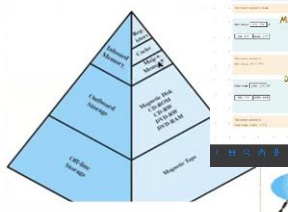


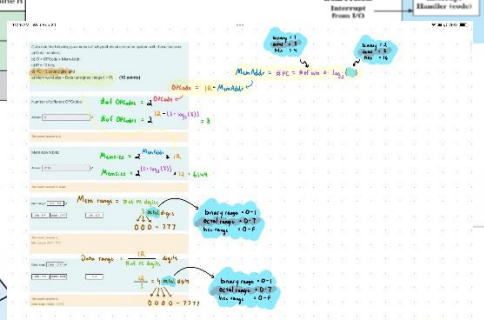
Figure 2.10 Virtual Memory Addressing

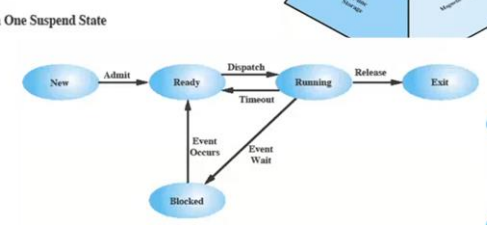




(a) With One Suspend State







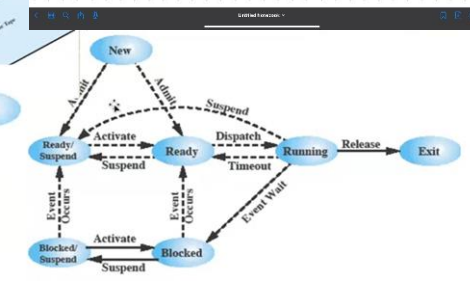


Figure 3.6 Five-State Process Model
(b) With Two Suspend States

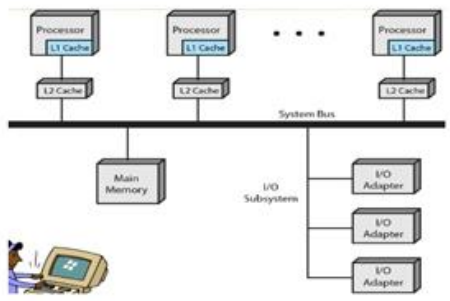
Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

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

Figure 2.4 System Utilization Example

- Processor is often idle
  - even with automatic job sequencing
  - I/O devices are slow compared to processor

	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



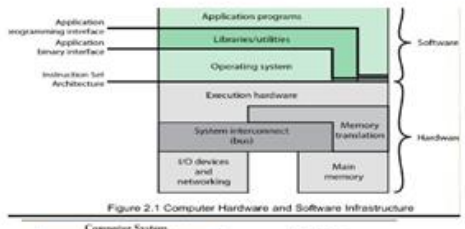
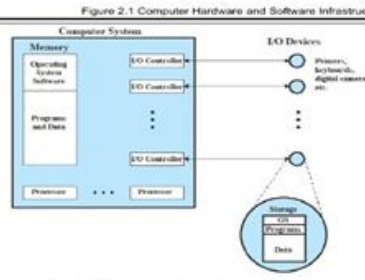


Figure 2.1 Computer Hardware and Software Infrastructure



Two essential elements of process are program code, may be shared with other processes that are executing same program & set of data associated with that code (referred to as process)
Process control block contains process elements, possible to pick upwere left off if interrupted, created & managed by OS.
Memory tables must include allocation of main memory to process,allocation of secondary memory to process, protection attributes ofblocks of main or virtual memory & information needed to manage virtual memory. Used to keep track of main (real) and secondary (virtual) memory.
IO tables used by OS to manage IO devices & channels of computer system. Has status of IO operation & location in main memory used assource/destination of Io transfer.
File tables provide info on existence of files, location in secondary memory, current status & other attributes.
Process tables must be maintained to manage processes, some reference to memory, IO and files (directly or indirectly). Tables must beaccessible to OS therefore subject to memory mgmt.
Process control structures has what OS must know: process location and attributes of process necessary for mgmt. Process location has setof programs to be executed & consist of at least sufficient memory to hold programs. Process attributes has collection of program, data, stack, and attributes referred to as process image.
Process switching may occur anytime the OS gained control from currently running process. There is interrupt, trap and supervisor call. Trap is handling of error or exception condition (depends on nature of error) and supervisor call is call to an OS.
If an interrupt is pending the processor, set Pc to starting address of interrupt handling program & switches from user mode to kernel mode so interrupt processing code may include privileged instructions. Else continues like normal and gets the next program/instruction.
Processes have 2 characteristics: Resource ownership (process includesvirtual address space to hold process image) and scheduling/execution (follows an execution path that may be interleaved with other processes).
A single threaded approach is a single thread execution per processwhere concept of thread not recognized (MS-DOS is an example).
A multithreaded approach has > 1 thread running per process and canbe the case for multiple processes (java run-time environment is an example).
Each thread has an execution state, saved thread context when notrunning, execution stack, some per-thread static storage for local variables & access to memory and resources of its process.
Benefits of threads is taking less time to create and terminate, switching between threads and threads enhance efficiency in communication between programs.
Key states for threads are running, ready and blocked. Change in threadstate are spawn, block, unblock & finish.
Thread synchronization is necessary as all threads of process hare samespace and resources (any alteration affects other threads).
ULT when thread mgmt. done by application & kernel not aware. Benefits include switching not need kernel mode, scheduling can bespecific & run on any OS. However when ULT executes system call, whole threads in process blocked. Overcome via jacketing.
KLT when thread mgmt. done by kernel, no mgmt. by application (Windows). Benefits include scheduling multiple threads from same process on multiple CPU's. If 1 thread in process blocked, can scheduleanother thread of same process. Kernel routines can be multithreaded. Disadvantage is that transfer of control from 1 thread to another requires mode switch to kernel.
Combined approach is thread creation done in user space & scheduling& synchro done by application (Solaris).

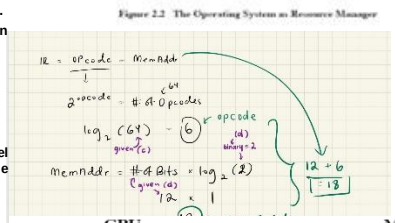
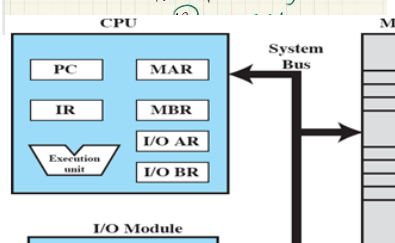
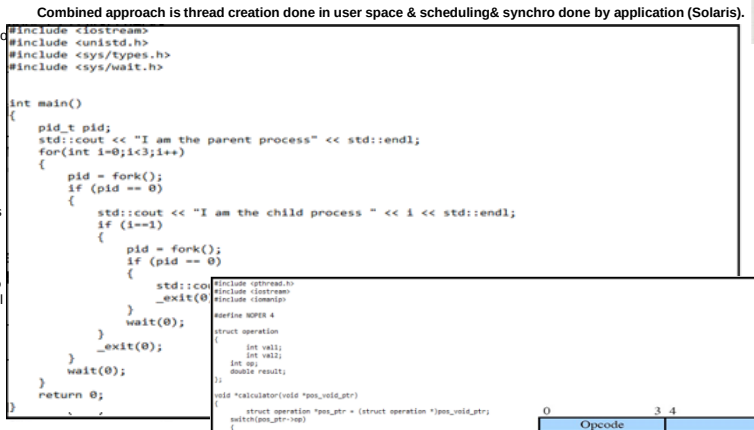


Figure 2.2 The Operating System as Resource Manager



PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register



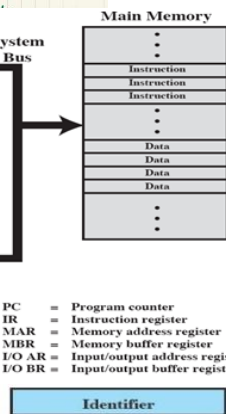


Figure 3.1 Simplified Process Control Block