



CS 3530 Operating Systems

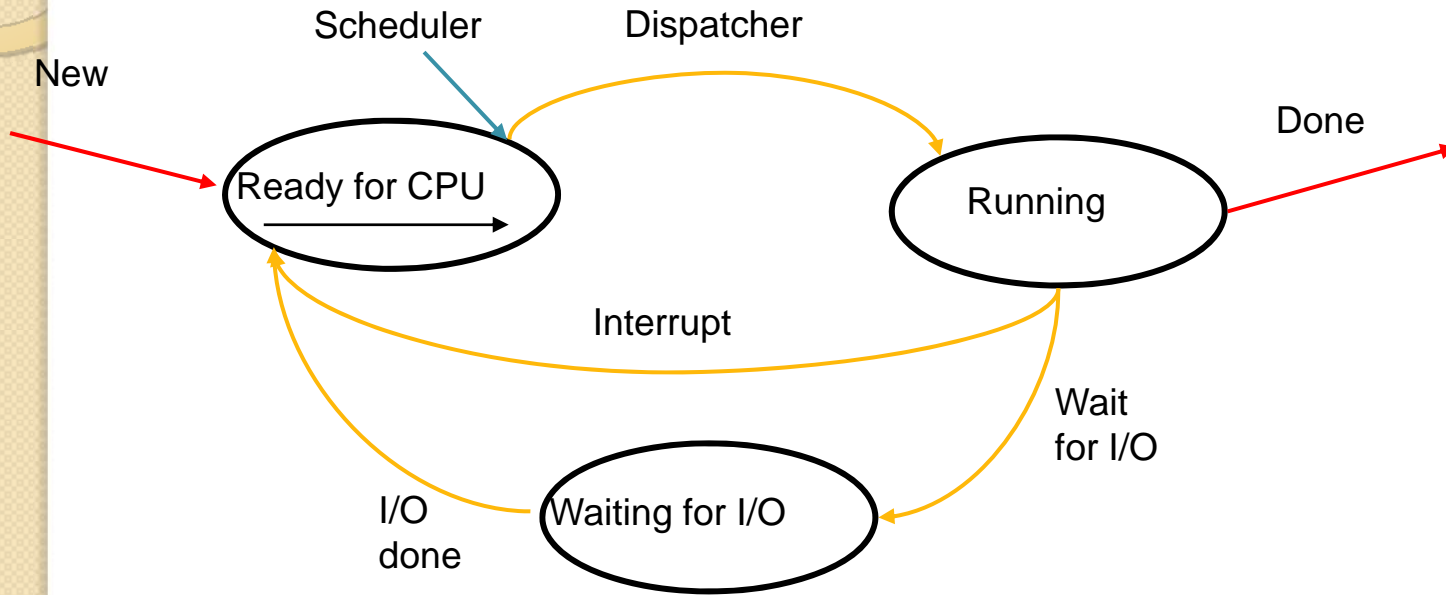
Lecture Subject/Title

CS 3530 Operating Systems. Dr. Ken Hoganson, Copyright © 2008

- Memory Management is a critical function of an OS
- Failure to correctly manage memory causes many “crashes”
 - Corrupted memory
 - Running out of memory
 - Failure to reclaim free memory
 - Invalid memory access
 - Historic Microsoft problems, while LINUX and other OS avoid problems with memory management

- Single process in memory – simple memory management
- Not efficient – CPU wastes time while process does Input/Output
- Many processes share memory – better use of CPU and other components – less wasted time waiting for I/O
- How to share memory between processes?
 - Divide memory into pages of fixed size
 - Processes are allocated memory in page increments as needed
 - Portions of pages may be unused.

- Other strategies have been attempted:
 - allocate variable-size memory chunks
 - As chunks are created, used, and released, small pieces (fragments) of memory are created, too small for anything to fit.
 - Requires periodic memory management to reclaim fragments into larger pieces that can hold.



- Each program consists of a set of pages of memory
- Process is a running program
- Each process needs only a subset of its pages at any one instant (the current pages)
- This subset is called the “Working Set” – the set of pages the process is working with at that time
- So, the **entire** program does NOT need to be in the computer’s memory in order to run the program

- If the **entire** program **does NOT** need to be in the computer's memory in order to run the program
- Then, available memory can be used for other program's working sets
- CPU can then multitask between programs, just by changing from one set of pages to another
- Improved efficiency and performance
- Requires that a process's pages be swapped in and out of memory as the current Working Set for that process changes dynamically

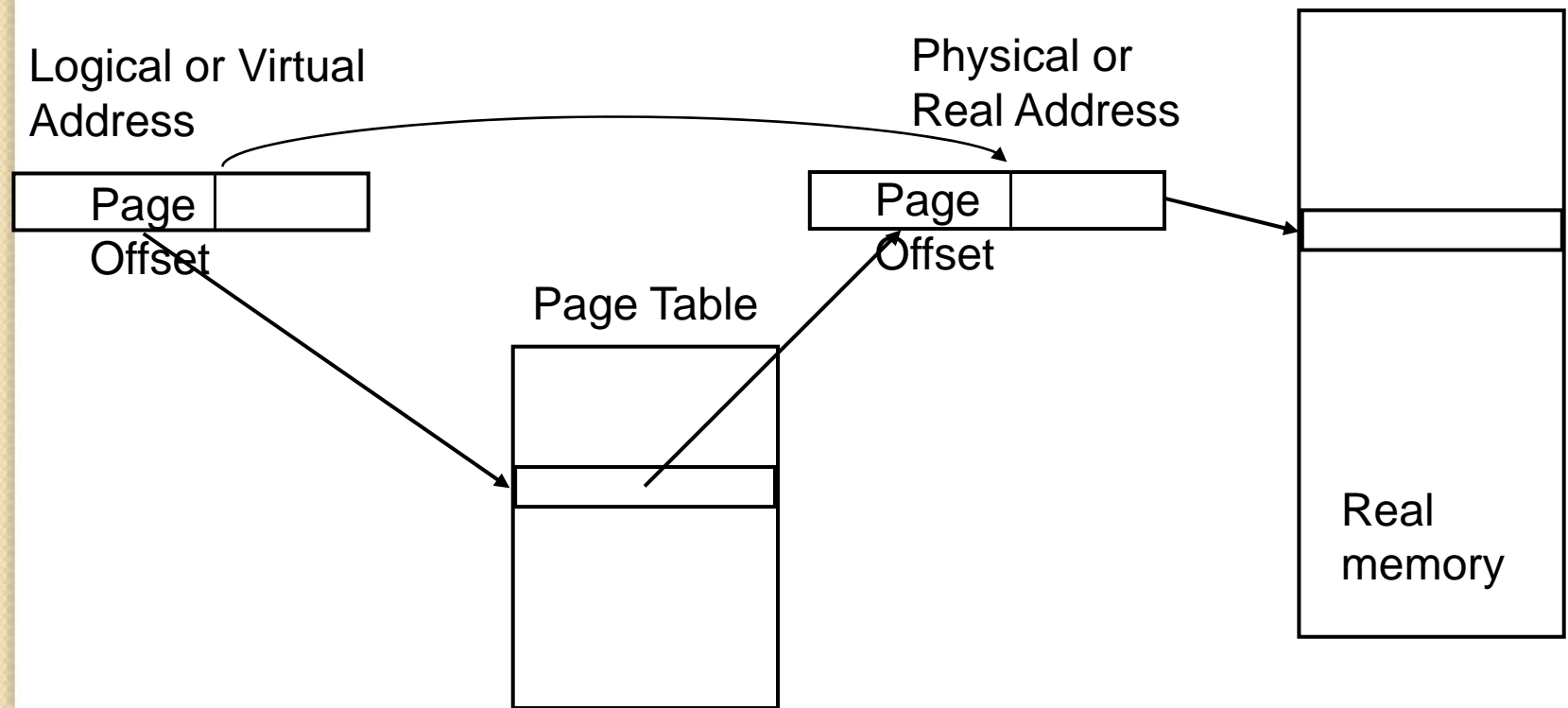
- **Scheduler:** Part of OS that determines which process in the ready queue will be executed next by the CPU
- **Dispatcher:** Part of OS that switches the CPU from one process to the next
- **Context Switch:** A switch from one process to another. The CPU changes from one context (the memory space and register contents of a process) to another.
- Time to run the scheduler and do the context switch is overhead
- These need to be fast and efficient algorithms

- Which pages to keep in memory?
 - the ones currently being used **plus** the ones most likely to be needed in the near term
- Similar strategy to caching!!
- Need to track the most recently used pages
- When a new page must be loaded in, unload the least recently used page
- How to implement this concept? (more later)
- The size of the computer's memory along with the size of the applications and their working sets, determines the number of process working sets that can be in the computer's memory at any one time
- Which determines utilization, efficiency, and performance

- Memory limits performance in a multi-tasking system
- Virtual memory is a technique used to make a system operate as if it has more real memory than it actually has.
 - More processes “ready” for execution – performance and efficiency
 - Allows processes that are larger than the computer’s real memory to run
- Virtual memory uses Secondary Storage (disk) memory to emulate real memory
- Works like a cache system:
 - real memory is like the high-speed, expensive cache
 - disk storage is the slower cheaper “main” memory
- The virtual memory space can be many times larger than all of real memory.

- Establishes a two-level scheduling system
- **Groups** of process working sets are moved between VM and real memory (high level of scheduling – long term queue)
- Multiple groups of process working sets in VM at a time.
- The CPU is switched between processes within the current group that is in real memory (lower level of scheduling – short term queue)
- Ideally, a group of processes that are switched between VM and real memory should be balanced in terms of the CPU and I/O work

- Physical Memory Address: page number + offset within page
- Logical Memory Address: virtual page number + offset within page
- CPU sends out Logical Addresses (virtual addresses)
- Logical Addresses must be translated to Physical Addresses
- Uses a page table for lookup and substitution



- Note that each address must now be translated – logical page replaced with physical page
- Means **two** memory accesses instead of one (lookup in page table): **Performance problem!**
- Solution: use a small cache just for page table translations:
Translation Lookaside Buffer (TLB)
- Translation overhead is very small
- TLB buffers are integrated on the CPU chip for modern processors

**End
Of
Today's
Lecture.**



This slide intentionally left blank