
EE282
Computer Architecture

Lecture 10: Memory Systems

October 30, 2001

Andrew Wolfe
Computer Systems Laboratory
Stanford University
awolfe@csl.stanford.edu

WJD/AW

Lecture 10, 10/30/01

1

Assignment

- Read
 - H&P 5.2 through 5.5
- Midterm - Thursday 7PM - Terman Aud.
 - Open book, open notes

WJD/AW

Lecture 10, 10/30/01

2

Today's Lecture

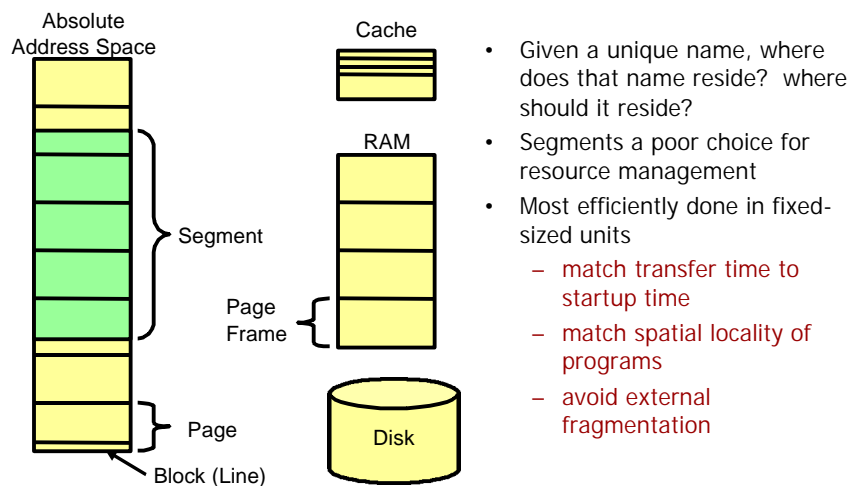
- Resource management in memory systems
 - how to make a large slow memory look fast
 - hierarchy
 - cache, main memory (DRAM), disk
 - virtual memory
 - read pages from disk on demand
 - requires restartable instructions
- page table organization
 - flat, hierarchical, inverted
- managing page frames
- translation lookaside buffers (TLBs)
- segmentation and paging
- case study: x86 memory management

WJD/AW

Lecture 10, 10/30/01

3

Resource Management



- Given a unique name, where does that name reside? where should it reside?
- Segments a poor choice for resource management
- Most efficiently done in fixed-sized units
 - match transfer time to startup time
 - match spatial locality of programs
 - avoid external fragmentation

WJD/AW

Lecture 10, 10/30/01

4

Paging and Virtual Memory

The diagram illustrates the mapping of logical node space (LNS) to physical memory (Phys Mem). On the left, LNS1 consists of three yellow boxes, and LNS2 consists of three green boxes. On the right, Phys Mem consists of three frames: a yellow one, a green one, and another yellow one. Arrows show the mapping: LNS1's top two boxes map to Phys Mem's top two frames, and LNS2's top box maps to Phys Mem's middle frame. A label 'page' is placed next to LNS2's top box, and a label 'frame' is placed next to Phys Mem's middle frame.

- All resource management schemes involve
 - mapping - locating a 'name'
 - where can it be put
 - how do you find it
 - policy - deciding where to put a name
 - what pages to put in memory
 - what pages to kick out (replacement or eviction)
- Paging maps pages to page frames

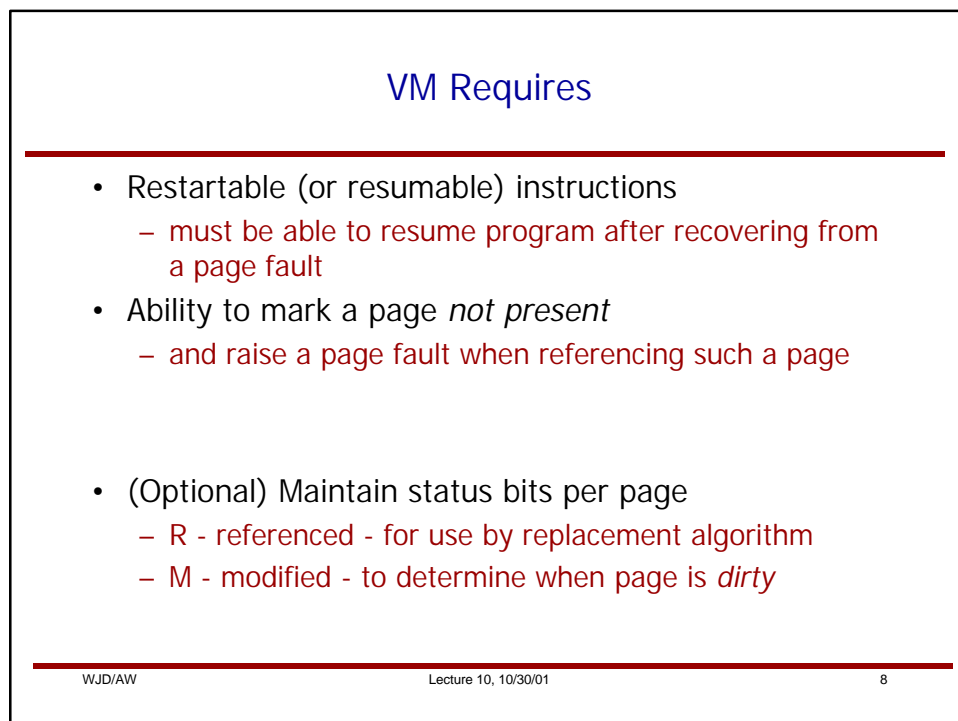
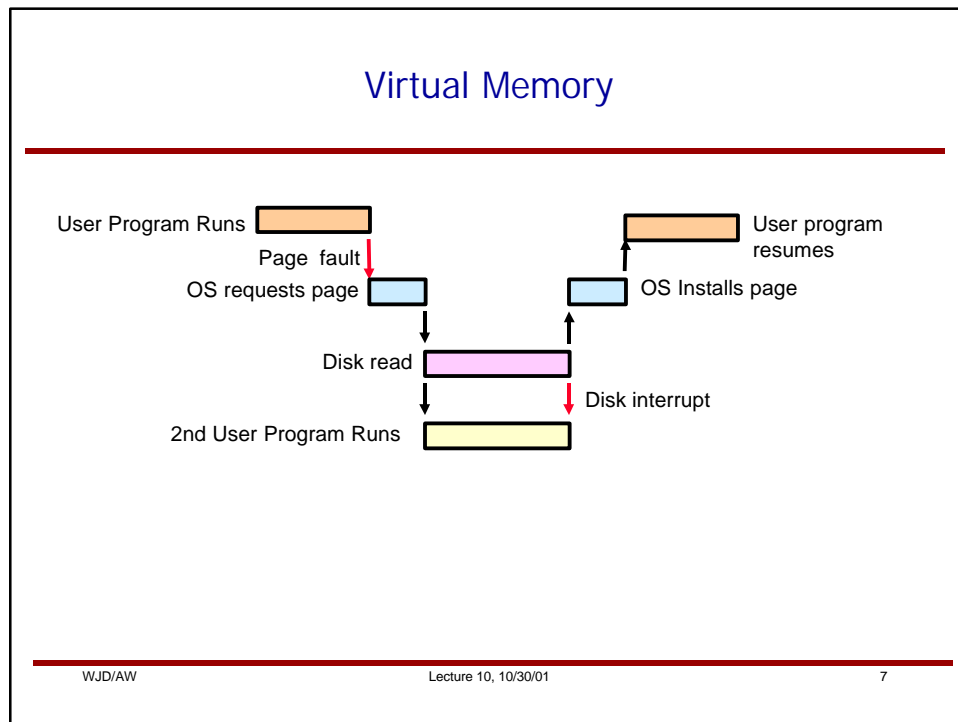
WJD/AW
Lecture 10, 10/30/01
5

Paging Translation

The diagram shows the process of translating a virtual address into a physical address. At the top, a 'Virtual Address' is split into 'Page' and 'Offset'. The 'Page' part is used to index into a 'Page Table' (PTP), which returns a 'Page Descriptor' containing 'S' (state) and 'Frame'. A green circle with a '+' sign indicates that the 'Page' and 'Offset' are concatenated to form the 'Physical Address'. The 'Physical Address' is also shown as a concatenation of 'Frame' and 'Offset'.

- Virtual address divided into *page* and *offset*
- Page used to index into a *page table* to read a *page descriptor*
- Page descriptor contains state and page frame
- Frame and Offset concatenated to form physical address

WJD/AW
Lecture 10, 10/30/01
6



Page Frame Management

Page Frame Table

Shows free list and list of frames allocated to Proc1

- OS maintains
 - **page table** for each user process
 - **page frame table**
 - free page list
 - pages evicted when number of free pages falls below a *low water mark*.
 - pages evicted using a *replacement policy*
 - random, FIFO, LRU, *clock*
 - if M-bit is clear, need not copy the page back to disk

WJD/AW
Lecture 10, 10/30/01
9

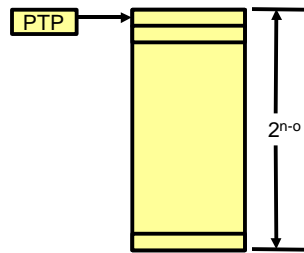
Page Management and Thrashing

Reference four pages in sequence, mapped to three page frames

- Need to keep a process' **working set** in memory or *thrashing* will occur
- Find working set size by increasing page frame allocation until PF/s falls below limit
- Swap out whole process if insufficient page frames for working set

WJD/AW
Lecture 10, 10/30/01
10

Page Table Organization



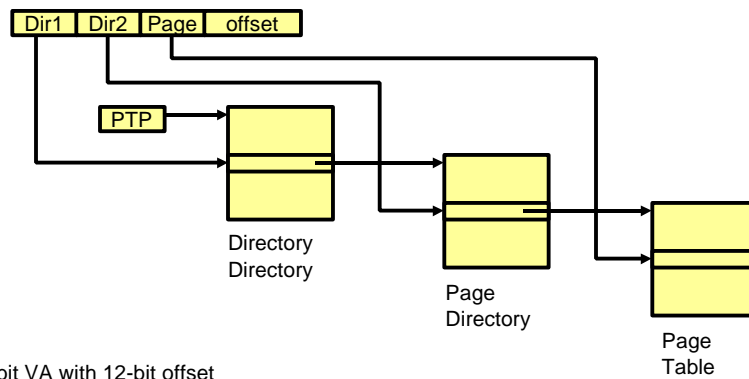
- Flat page table has size proportional to size of *virtual* address space
 - can be very large for a machine with 64-bit addresses and several processes
- Three solutions
 - page the page table (fixed mapping)
 - what really needs to be *locked down*?
 - multi-level page table (lower levels paged - Tree)
 - inverted page table (hash table)

WJD/AW

Lecture 10, 10/30/01

11

Multi-Level Page Table



e.g., 42-bit VA with 12-bit offset
 10-bits for each of three fields
 1024 4-byte entries in each table (one page)

WJD/AW

Lecture 10, 10/30/01

12

Inverted Page Tables

The diagram illustrates the inverted page table mechanism. It starts with a 'Virtual Address' box divided into 'Page' and 'Offset'. The 'Page' part goes through a 'Hash' function (represented by a green oval). The output of the hash function points to a table with three columns: 'Page', 'Frame', and 'S'. Below this table is a green circle containing an equals sign (=). An arrow from the 'Page' column of the table points to this circle. Another arrow from the 'Page' part of the 'Virtual Address' also points to this circle. Below the circle is the text 'OK'. An arrow from the 'OK' text points to a 'Frame' and 'Offset' box. Additionally, an arrow from the 'Offset' part of the 'Virtual Address' points directly to the 'Offset' part of the 'Frame Offset' box.

- Store only PTEs for pages *in* physical memory
- Miss in page table implies page is on disk
- Need KP entries for P page frames (usually $K > 2$)
- Page table organization can be defined by software if TLB miss raises an exception

Key point: Size of page table is proportional to size of **physical** memory, not the size of **virtual** memory

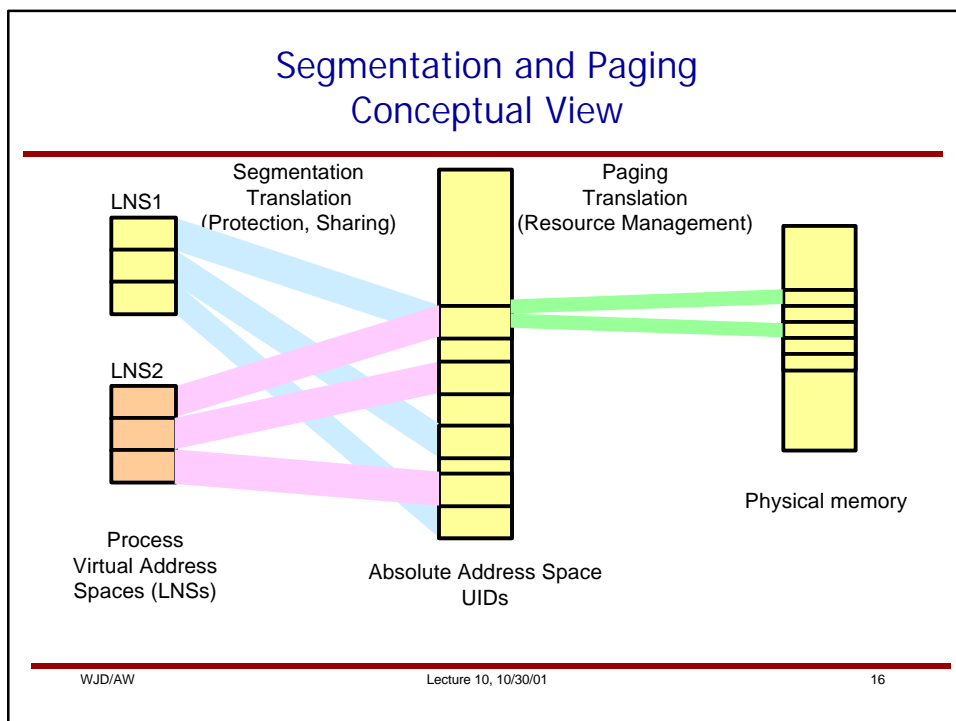
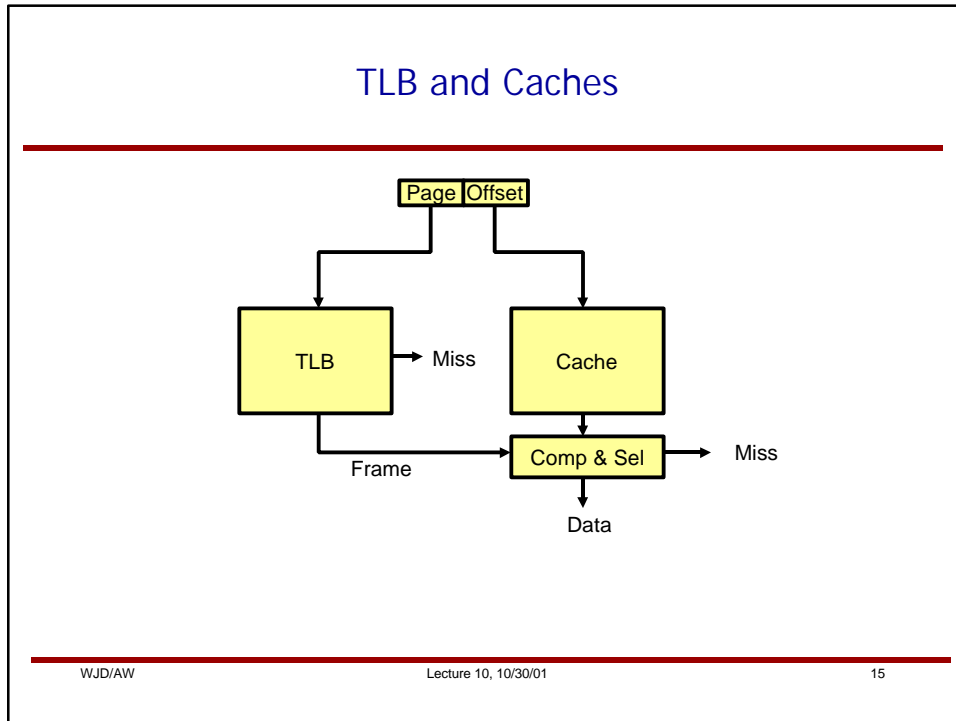
WJD/AW
Lecture 10, 10/30/01
13

Paging Translation Lookaside Buffers

The diagram shows a TLB (Translation Lookaside Buffer) table with three columns: 'VWRM', 'Page', and 'Frame'. Above the table is a box for 'PID | Page | Offset'. An arrow from the 'Page' part of this box points to the 'Page' column of the TLB table. Another arrow from the 'Page' part of the box points to a 'Frame | Offset' box at the bottom. A third arrow from the 'Page' column of the TLB table points to the 'Frame' part of the 'Frame | Offset' box. The 'Offset' part of the 'Frame | Offset' box is also pointed to by an arrow from the 'Offset' part of the top box.

- Store most frequently used translations in small, fast memory
- On miss - two approaches
 - hardware state machine *walks the page table*
 - fast but inflexible
 - exception raised and software *walks the page table*
- TLB must include process ID or be flushed on each process switch or system call
- Referenced and modified bits must be copied back on changes
 - alias problem

WJD/AW
Lecture 10, 10/30/01
14



Segmentation and Paging - a simple approach

- Align all segments to page boundaries
- Round up segment lengths to page boundaries
- Segment becomes a collection of contiguous pages
 - need not map to contiguous set of frames
- Two approaches to protection
 - duplicate protection bits on each page
 - check segment protection in parallel with paging translation
- Protection and sharing in variable-sized logical chunks
- Resource management in fixed-size, aligned physical chunks

WJD/AW

Lecture 10, 10/30/01

17

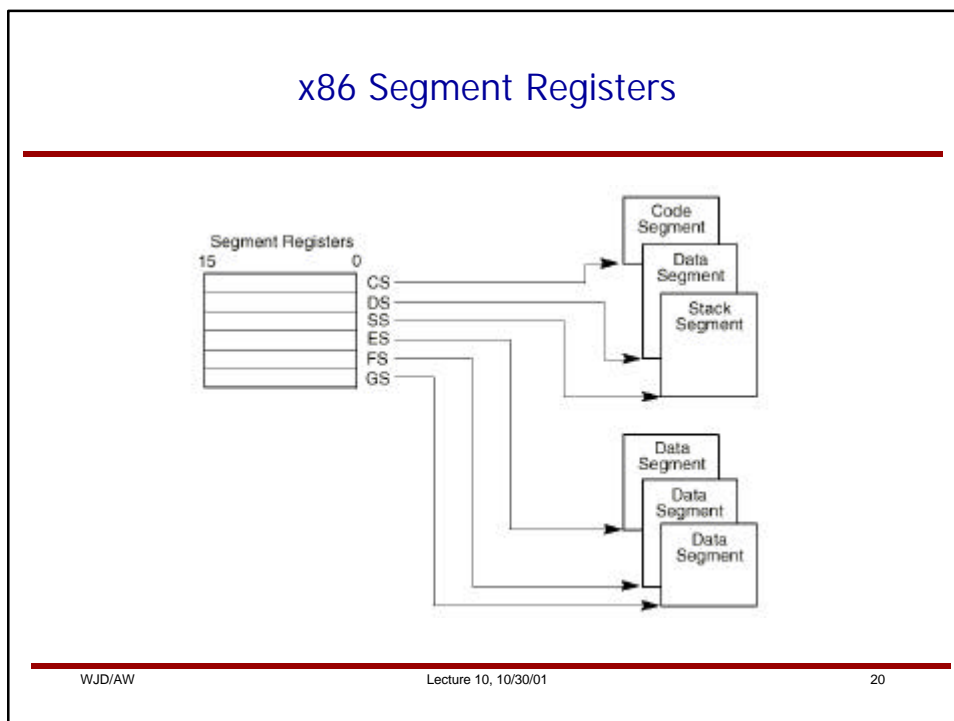
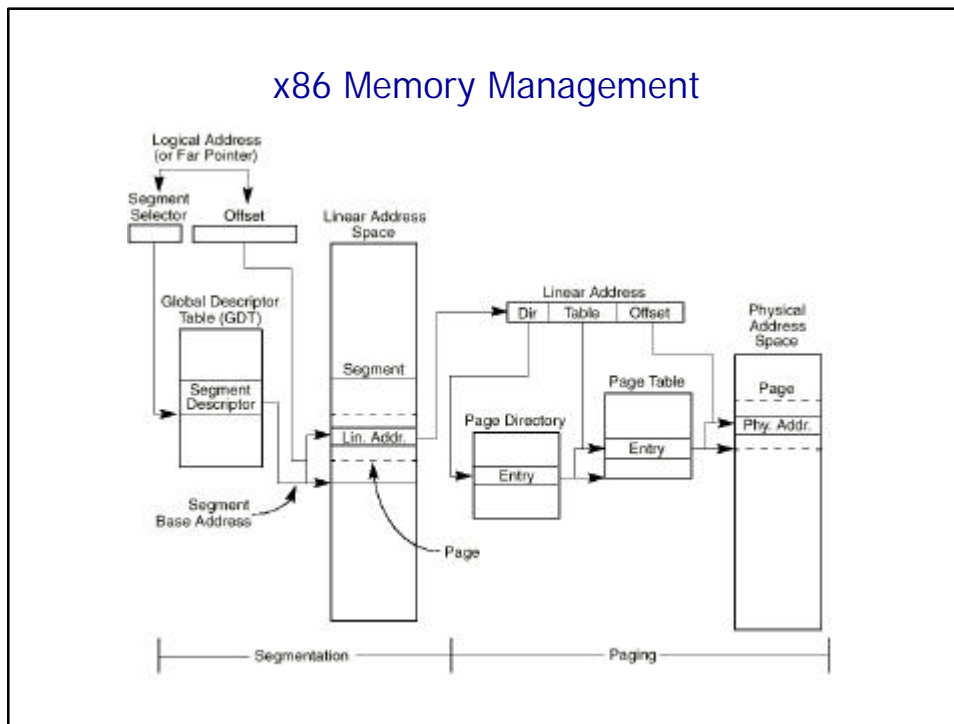
Case Study - x86 Memory Management

- Segmentation translation to *linear* address space
 - 16-bit *selector* appended to 32-bit offset
- Gate mechanism for change of privilege level
- Paging translation with two-level page table

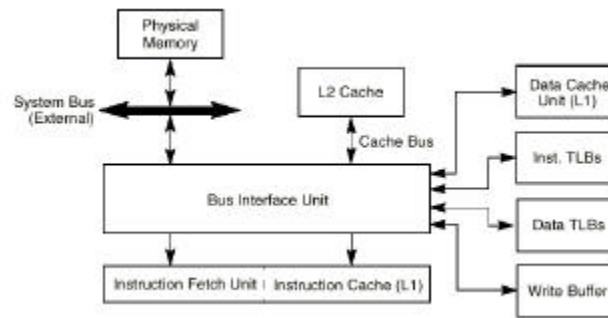
WJD/AW

Lecture 10, 10/30/01

18



Caches in the P6 Implementation of x86



Next Time

- Caches