

Managing Segments (cont.) (Review)

- To enlarge a segment:
 - If space above the segment is free, OS can just update the segment's limit and use some of that space
 - Move this segment to a larger free space
 - Swap the segment above this one to disk
 - Swap this segment to disk, and bring it back into a larger free space
- Advantages of segmentation:
 - Segments don't have to be contiguous
 - Segments can be swapped independently
 - Segments allow sharing
- Disadvantages of segmentation:
 - Complex memory allocation (first-fit, etc.)
 - External fragmentation

1

Fall 2002, Lecture 27

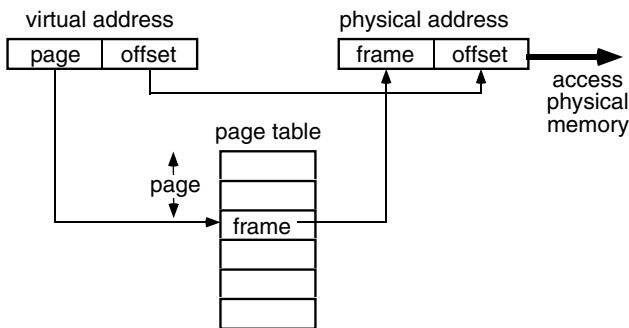
Paging

- Compared to segmentation, paging:
 - Makes allocation and swapping easier
 - No external fragmentation
- Each process is divided into a number of small, fixed-size partitions called *pages*
 - Physical memory is divided into a large number of small, fixed-size partitions called *frames*
 - Pages have nothing to do with segments
 - Page size = frame size
 - Usually 512 bytes to 16K bytes
 - The whole process is still loaded into memory, but the pages of a process do **not** have to be loaded into a contiguous set of frames
 - Virtual address consists of page number and offset from beginning of that page

2

Fall 2002, Lecture 27

Implementing Paging

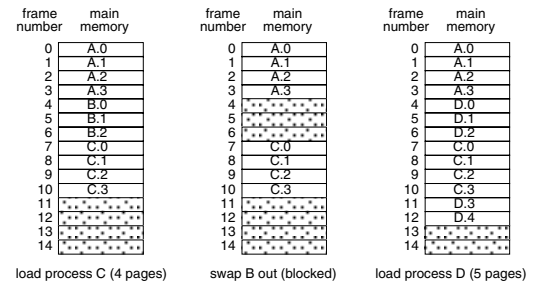
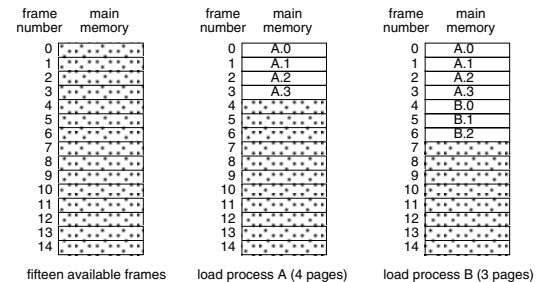


- A *page table* keeps track of every page in a particular process
 - Each entry contains the corresponding frame in main (physical) memory
 - Can add protection bits, but not as useful
- Additional hardware support required is slightly less than for segmentation
 - No need to keep track of, and compare to, limit. Why not?

3

Fall 2002, Lecture 27

Paging Example

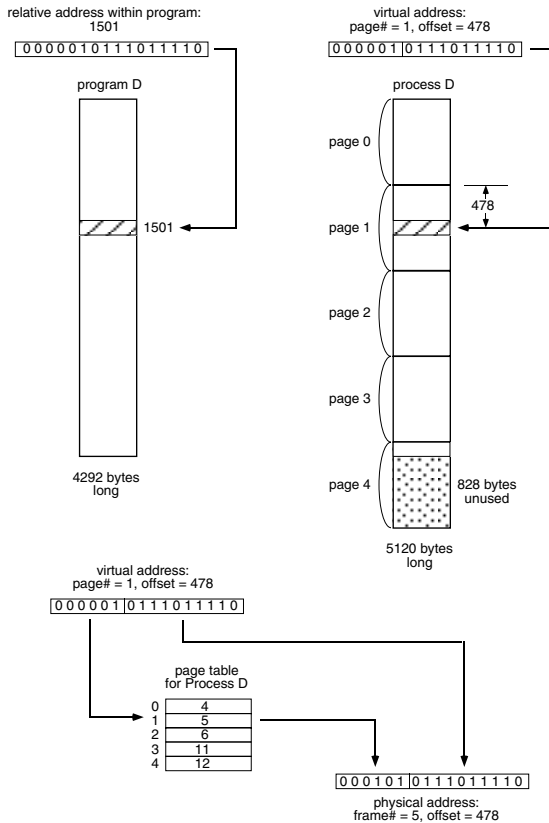


page table for Process A	page table for Process B	page table for Process C	page table for Process D	list of free frames
0 0	0	0 7	0 4	13
1 1	1	1 8	1 5	14
2 2	2	2 9	2 6	
3 3	3	3 10	3 11	
			4 12	

4

Fall 2002, Lecture 27

Paging Example (cont.)



5

Fall 2002, Lecture 27

Managing Pages and Frames

- OS usually keeps track of free frames in memory using a bit map
 - A bit map is just an array of bits
 - 1 means the frame is free
 - 0 means the frame is allocated to a page
 - To find a free frame, look for the first 1 bit in the bit map
 - Most modern instruction sets have an instruction that returns the offset of the first 1 bit in a register
- Page table base pointer (special register) points to page table of active process
 - Saved/restored as part of context switch
 - Page table also contains:
 - Other bits for demand paging (discussed next time)

6

Fall 2002, Lecture 27

Evaluation of Paging

- Advantages:
 - Easy to allocate memory — keep a list of available frames, and simple grab first one that's free
 - Easy to swap — pages, frames, and often disk blocks as well, all are same size
 - One frame is just as good as another!
- Disadvantages:
 - Page tables are fairly large
 - Most page tables are too big to fit in registers, so they must live in physical memory
 - This table lookup adds an extra memory reference for every address translation
 - Internal fragmentation
 - Always get a whole page, even for 1 byte
 - Larger pages makes the problem worse
 - Average = 1/2 page per process

7

Fall 2002, Lecture 27

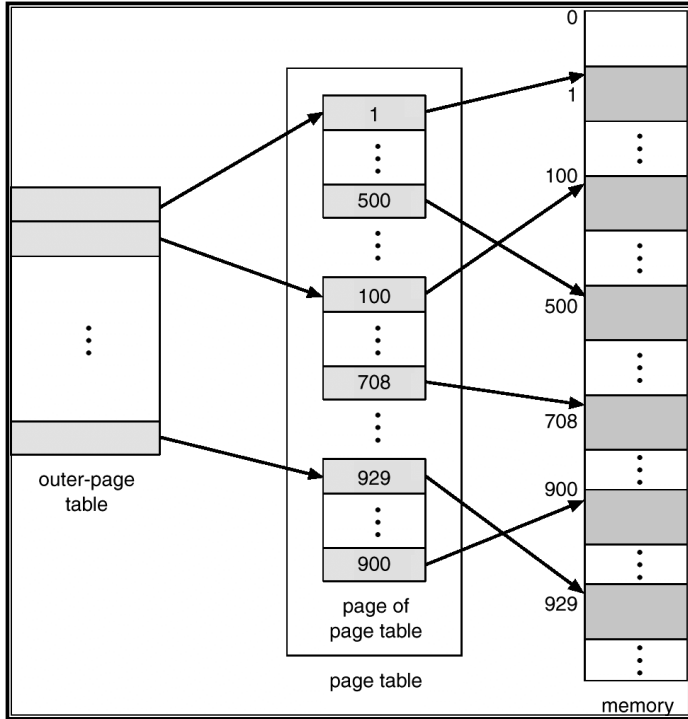
Address Translation, Revisited

- A modern microprocessor and OS has maybe a 32 bit virtual address space per process ($2^{32} = 4 \text{ GB}$)
 - If page size is 4k (2^{12}), $32-12=20$, meaning each page table could have up to 2^{20} (approximately 1 million) page entries, each maybe 4 bytes long = 4MB
 - Problem: page table is too large to store in one page, can't store contiguously
 - Two-level page tables: page tables are also stored in virtual memory
 - New problem: memory access time may double since the page tables are now subject to paging
 - (one access to get info from page table, plus one access to get data from memory)
 - New solution: use a special cache (called a Translation Lookaside Buffer (TLB)) to cache page table entries

8

Fall 2002, Lecture 27

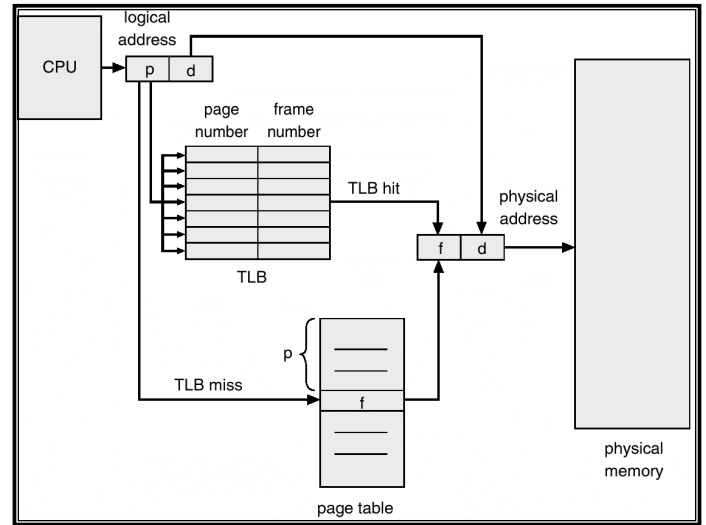
Two-Level Page Table



9

Fall 2002, Lecture 27

Translation Look-Aside Buffer



10

Fall 2002, Lecture 27

Paging and Segmentation

- Use two levels of mapping:
 - Process is divided into variable-size segments
 - Segments are logical divisions as before
 - Each segment is divided into many small fixed-size pages
 - Pages are easy for OS to manage
 - Eliminates external fragmentation
 - Virtual address = segment, page, offset
 - One segment table per process, one page table per segment
- Sharing at two levels: segment, page
 - Share frame by having same frame reference in two page tables
 - Share segment by having same base in two segment tables
 - Still need protection bits (sharing, r/o, r/w)

11

Fall 2002, Lecture 27