CISC 3320

Main Memory: Segmentation

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the following textbook
  - Andrew S. Tanenbaum and Herbert Bos. 2014. Modern Operating Systems (4th ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.

One Logical Address Space?

- Thus far, we assume we have only one logical address space

More Logical Address Spaces?

- Examine the following scenario: a compiler generates tables when compiling a program

1. The source text being saved for the printed listing
2. The symbol table, names and attributes of variables.
3. The table containing integer and floating-point constants used.
4. The parse tree, syntactic analysis of the program.
5. The stack used for procedure calls within compiler.

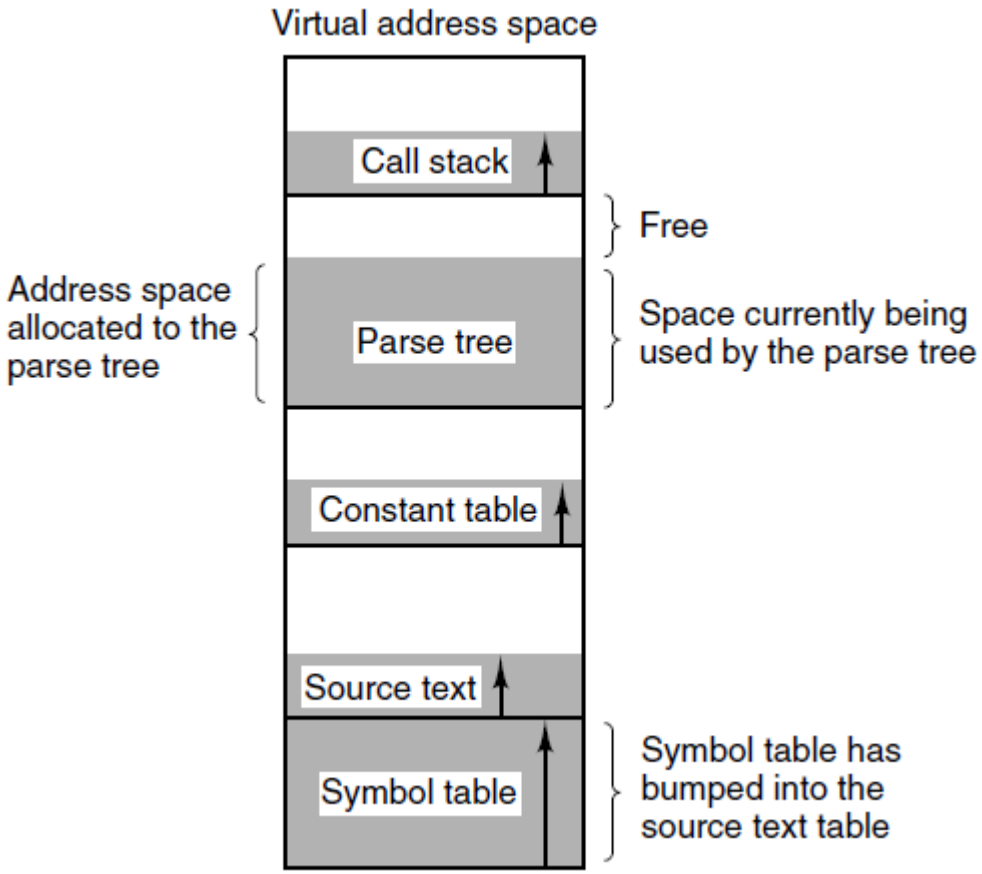


Figure 3-30. In a one-dimensional address space with growing tables, one table may bump into another [In Tanenbaum and Bos (2014)]

Segmentation

- Provide to the CPU with many completely independent logical address spaces, each called a *segment*

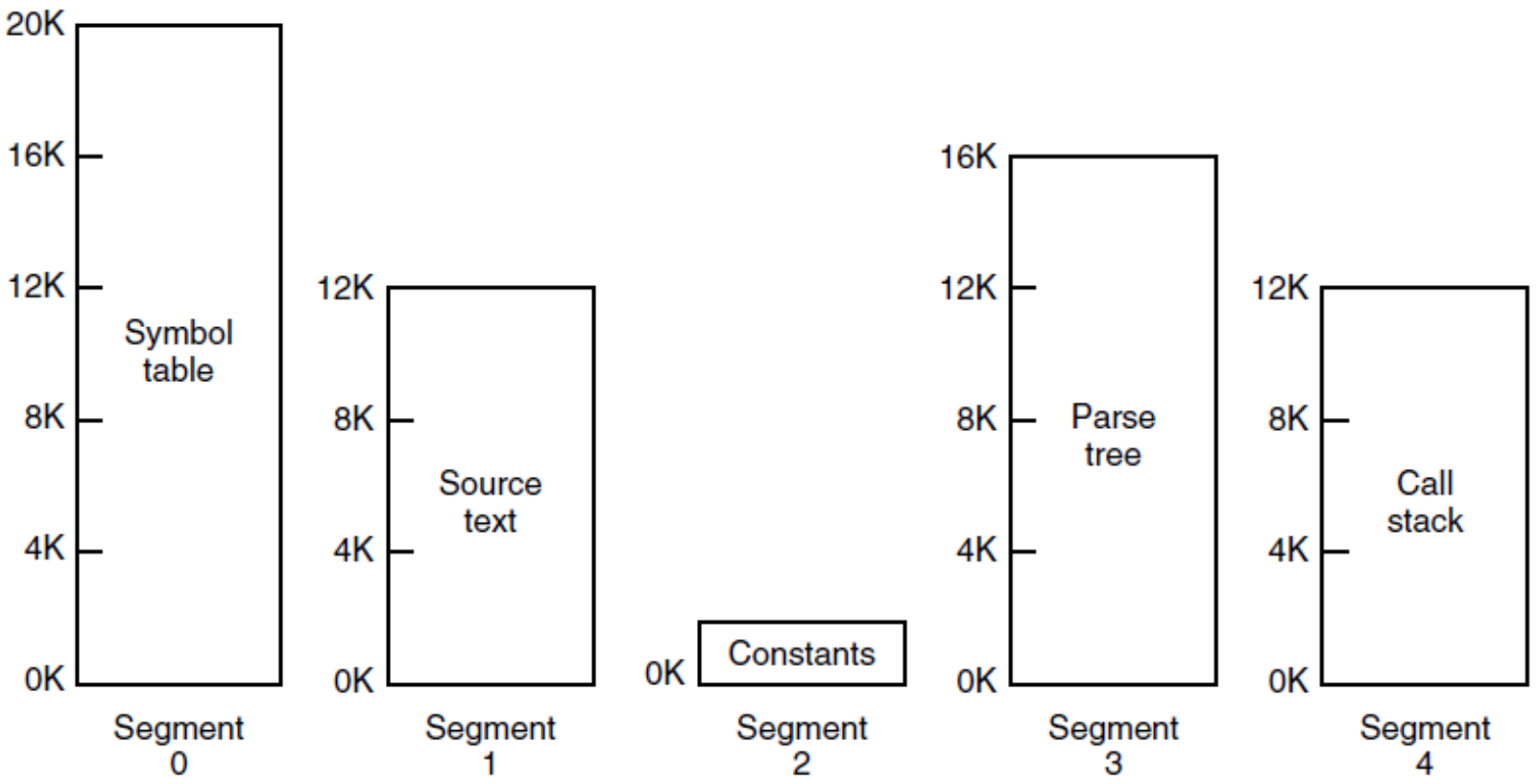


Figure 3-31. A segmented memory allows each table to grow or shrink independently of the other tables [In Tanenbaum and Bos (2014)]

Segmentation vs. Paging

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Figure 3-32. in In Tanenbaum and Bos (2014)

Pure Segmentation and External Fragmentation

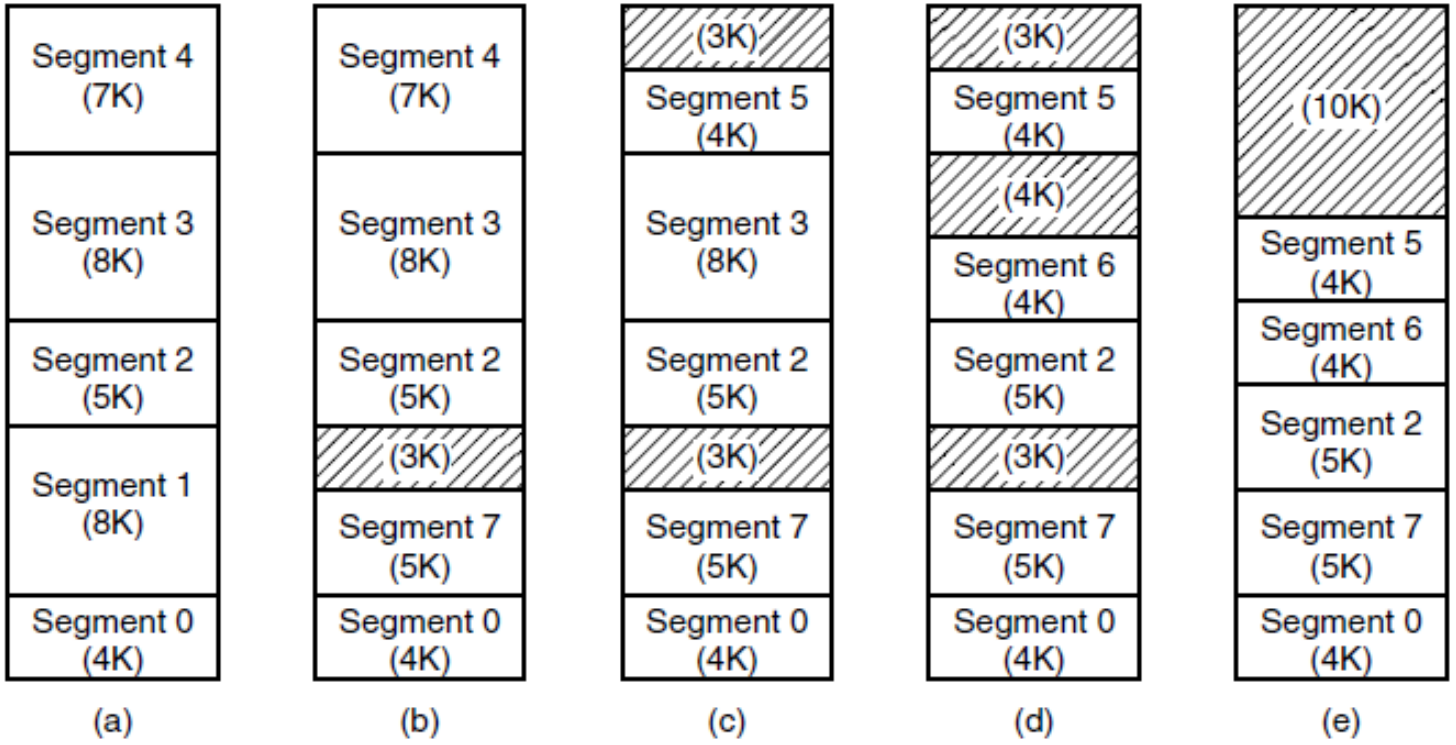


Figure 3-33. in In Tanenbaum and Bos (2014): (a)-(d) showing occurrences of external fragmentation; (e) showing memory compaction to remove external fragmentation

Segmentation with Paging

- Take advantage of multiple logical address spaces
- Eliminate external fragmentation

Segmentation with Paging: MULTICS

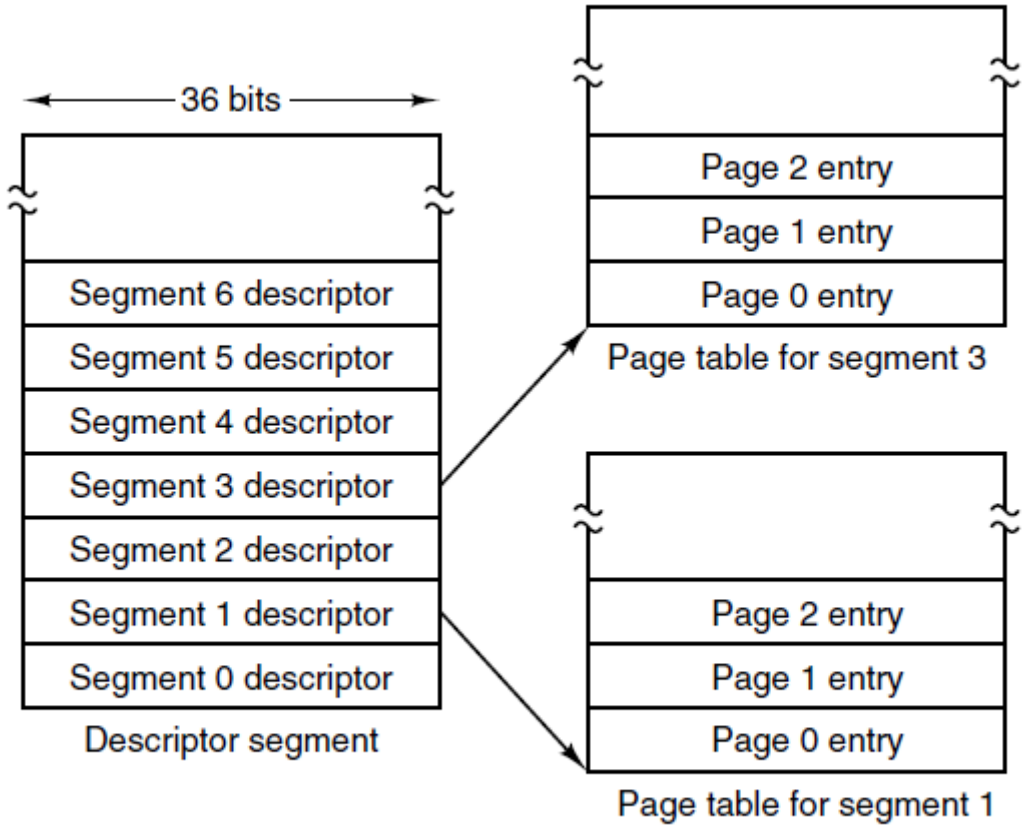


Figure 3-34 The descriptor segment pointed to the page tables. [In Tanenbaum and Bos (2014)]

MULTICS Segment Descriptor

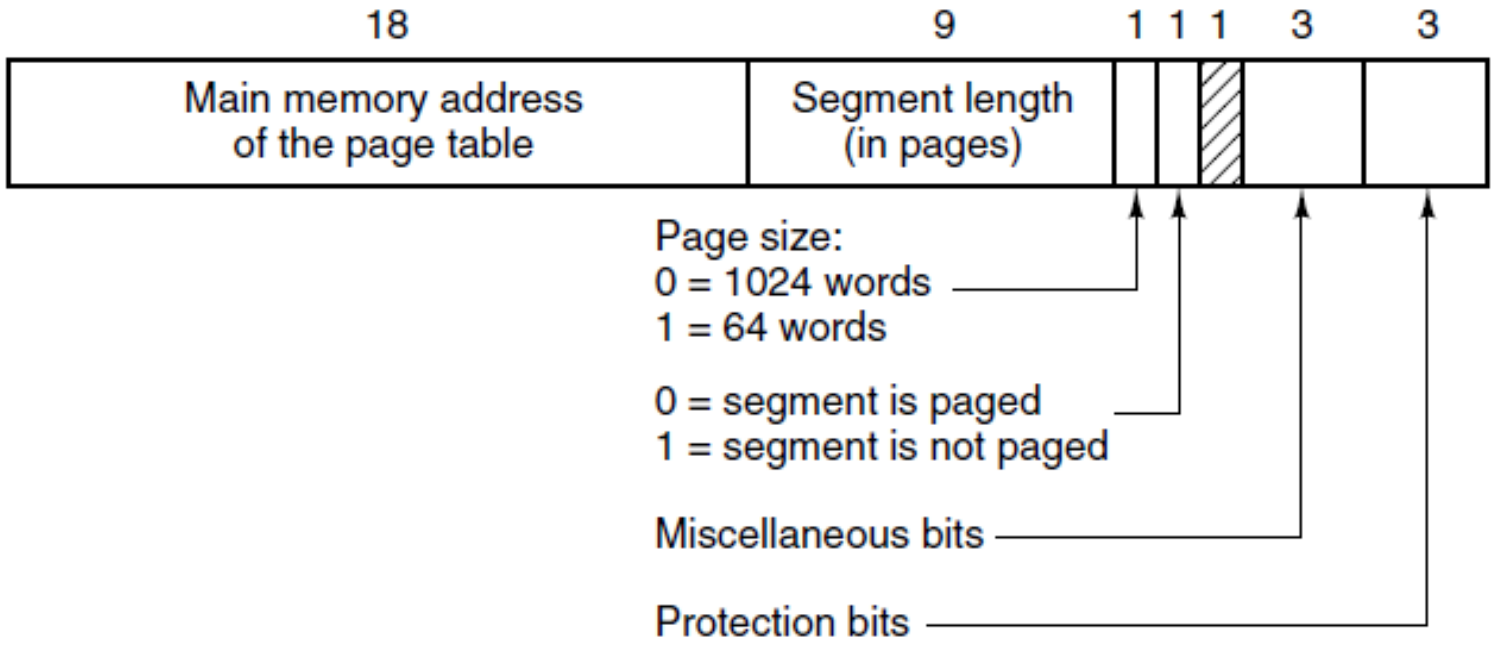


Figure 3-34. A segment descriptor. The numbers are the field lengths. [In Tanenbaum and Bos (2014)]

MULTICS Logical Address

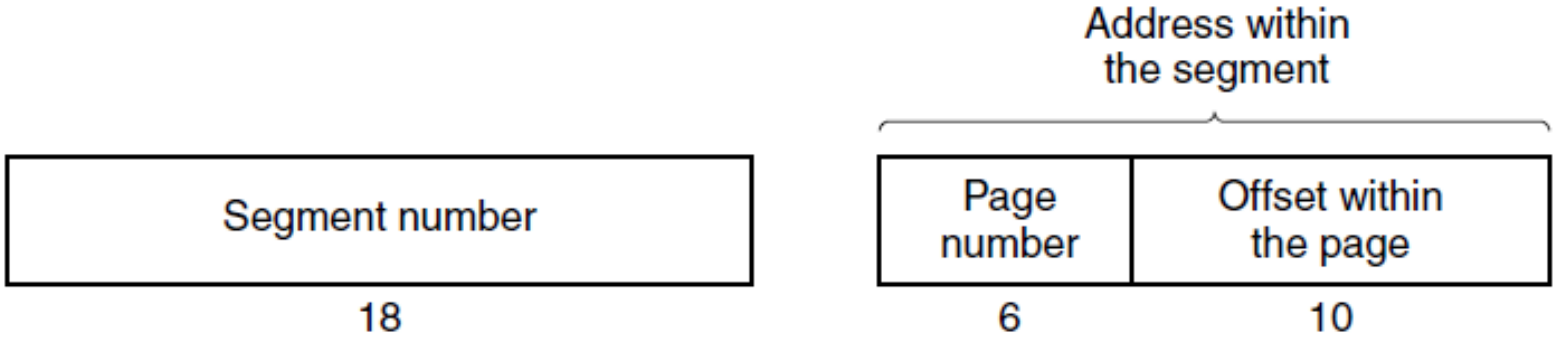


Figure 3-35. A 34-bit MULTICS logical address [In Tanenbaum and Bos (2014)]

MULTICS Logical to Physical Address Translation

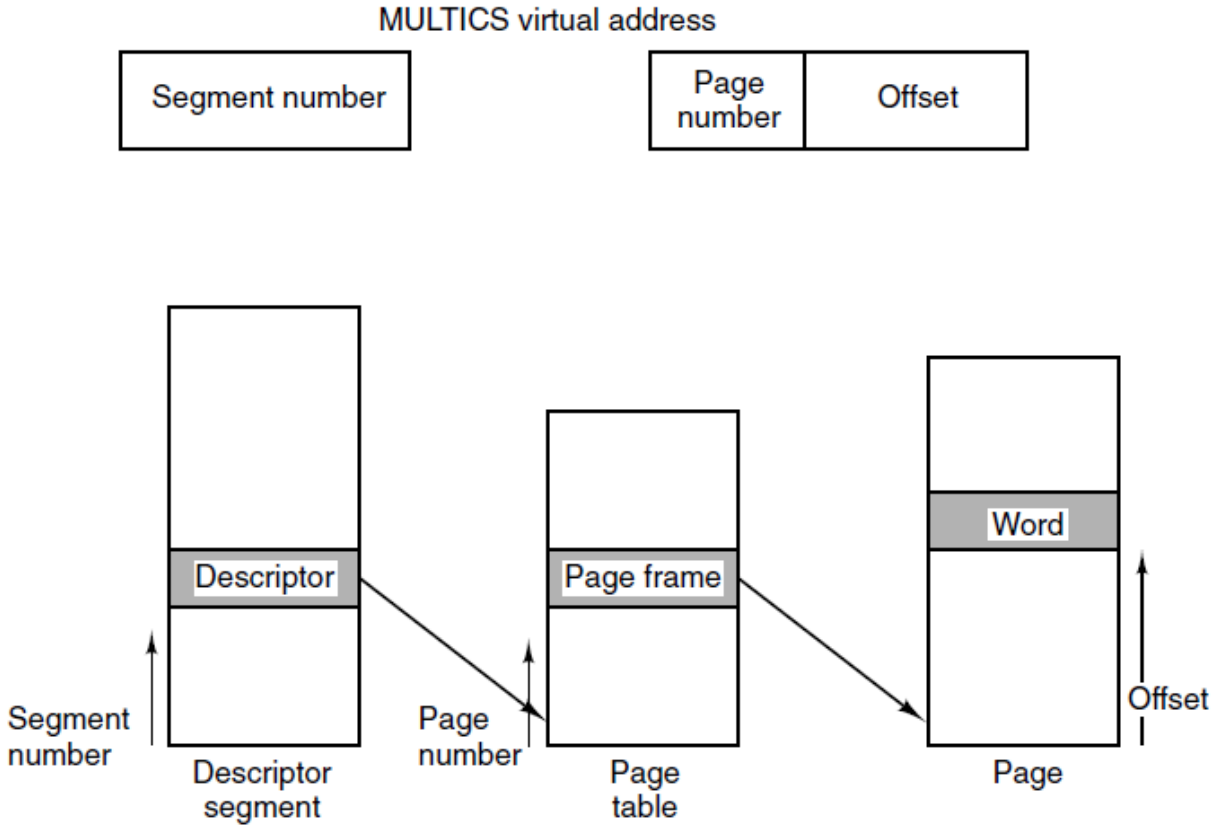
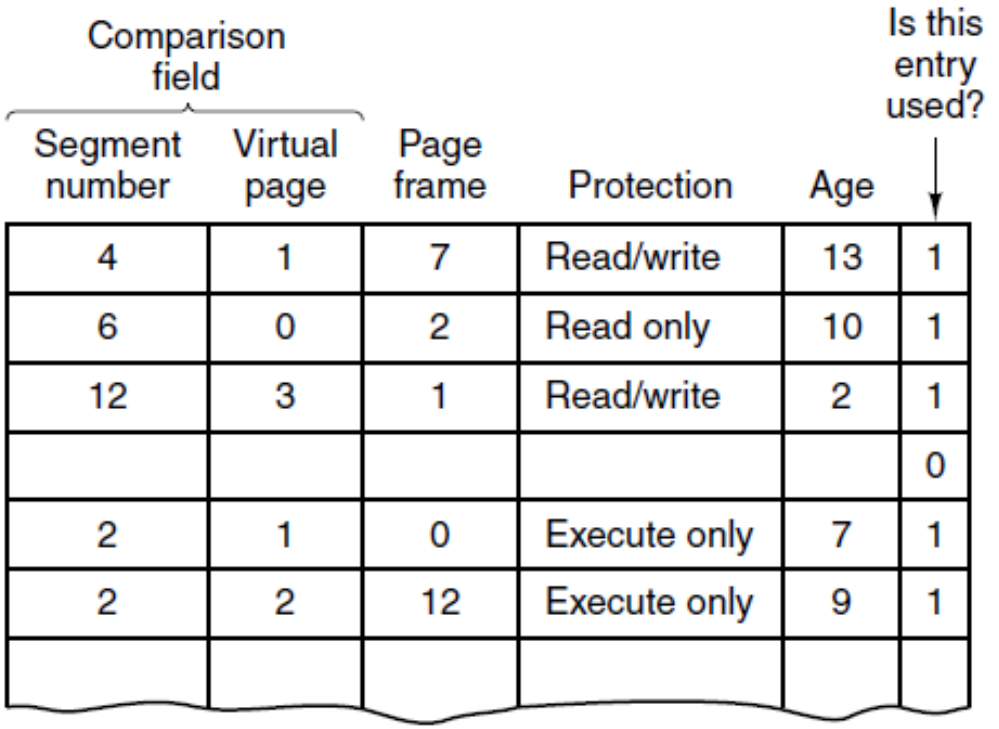


Figure 3-36. Conversion of a two-part MULTICS address into a physical address. [In Tanenbaum and Bos (2014)]

MULTICS TLB



Comparison field: Segment number	Comparison field: Virtual page	Page frame	Protection	Age	Is this entry used?
4	1	7	Read/write	13	1
6	0	2	Read only	10	1
12	3	1	Read/write	2	1
					0
2	1	0	Execute only	7	1
2	2	12	Execute only	9	1

Figure 3-37. A simplified version of the MULTICS TLB. The existence of two page sizes made the actual TLB more complex. [In Tanenbaum and Bos (2014)]

Segmentation with Paging: x86

- x86 supports segmentation and has two tables
  - Local Descriptor Table (LDT) and Global Descriptor Table (GDT)
    - LDT describes segment local to each program, and GDT describes system segments, such as, those for the operating system itself
    - To access a segment, an x86 program loads a selector for that segment to one of the CPU's 6 segment registers
      - CS holds code segment, DS data segment

An x86 Segment Selector

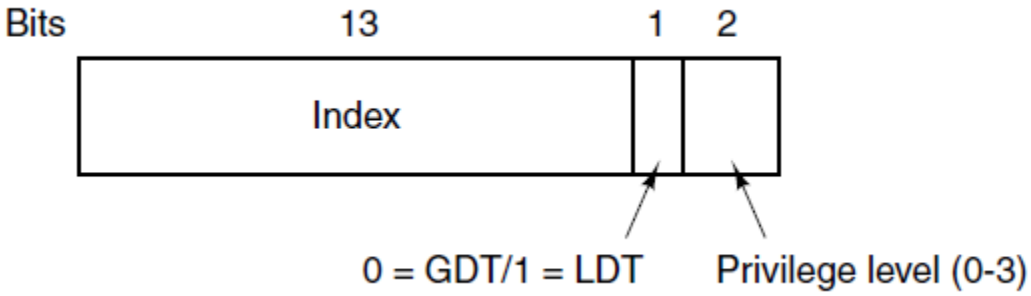


Figure 3-38. An x86 selector. [In Tanenbaum and Bos (2014)]

An x86 Code Segment Descriptor

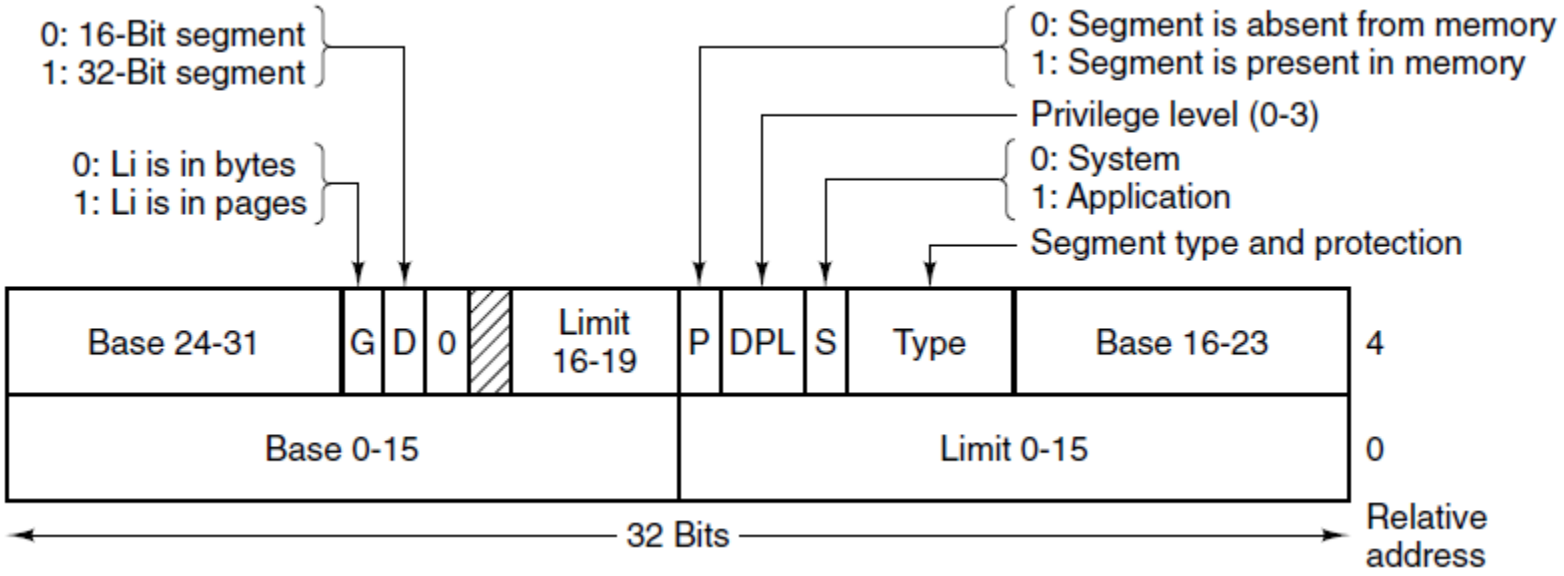


Figure 3-39. x86 code segment descriptor. Data segments differ slightly. [In Tanenbaum and Bos (2014)]

Translate Logical Address to Physical Address

- Essentially, two steps

1. Translate (selector, offset) to linear address
2. Translate linear address to physical address

Translate Logical Address to Linear Address

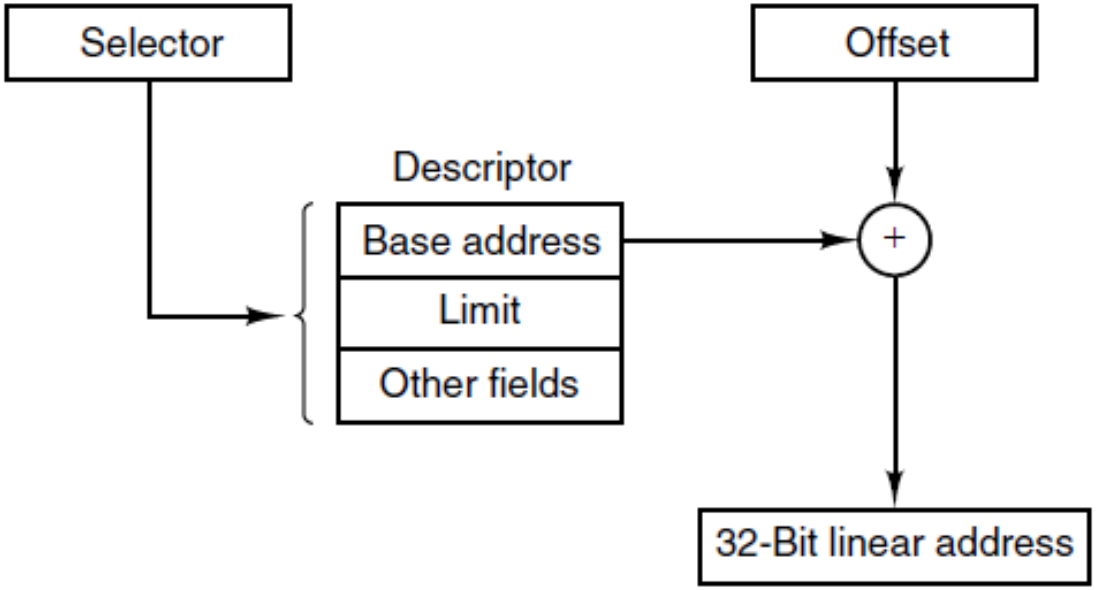


Figure 3-40. Conversion of a (selector, offset) pair to a linear address. [In Tanenbaum and Bos (2014)]

Translate Linear Address to Physical Address

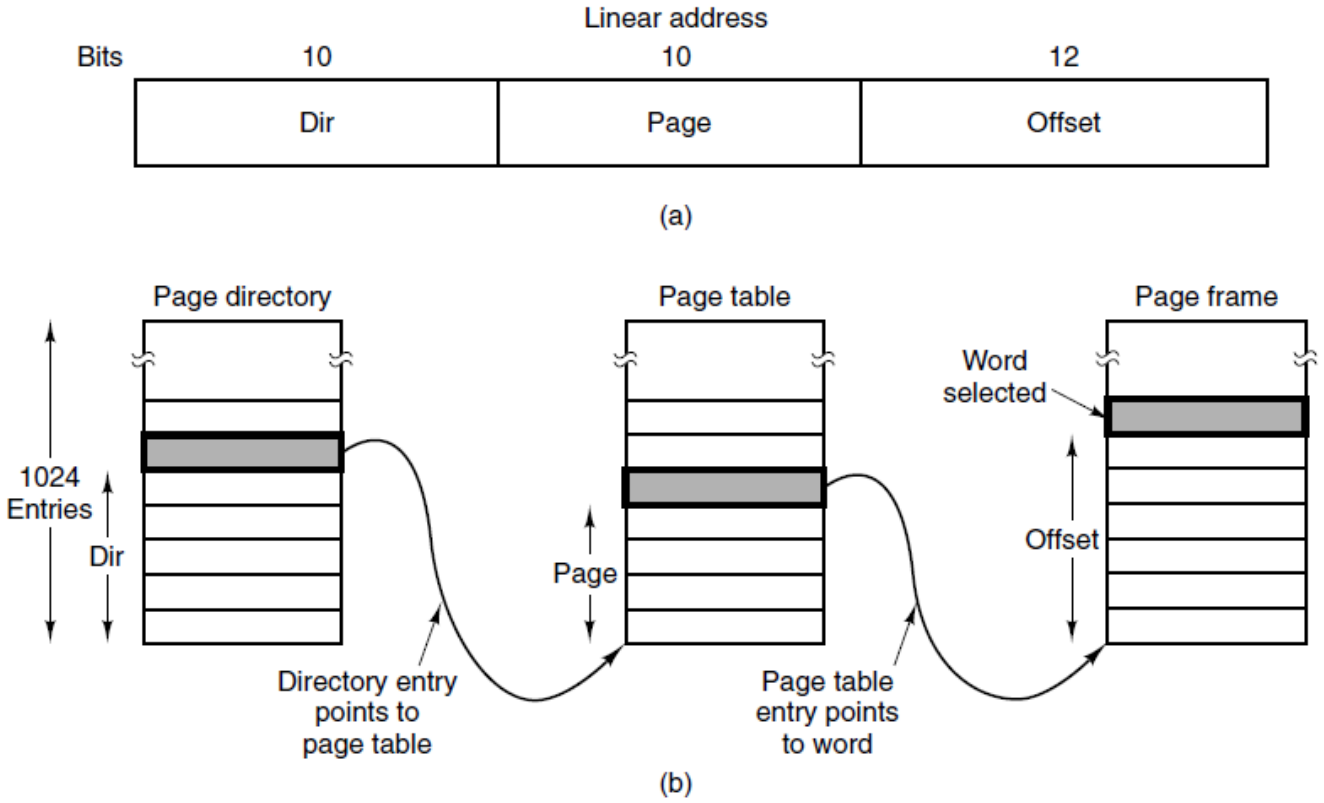


Figure 3-41. Mapping of a linear address onto a physical address [In Tanenbaum and Bos (2014)]

Questions?

- Concept of segmentation
- Examples of segmentation
  - MULTICS
  - x86 (a.k.a., x86-32)
- How does x86-64 deal with segmentation (at all)?