Segmentation

Another memory management scheme

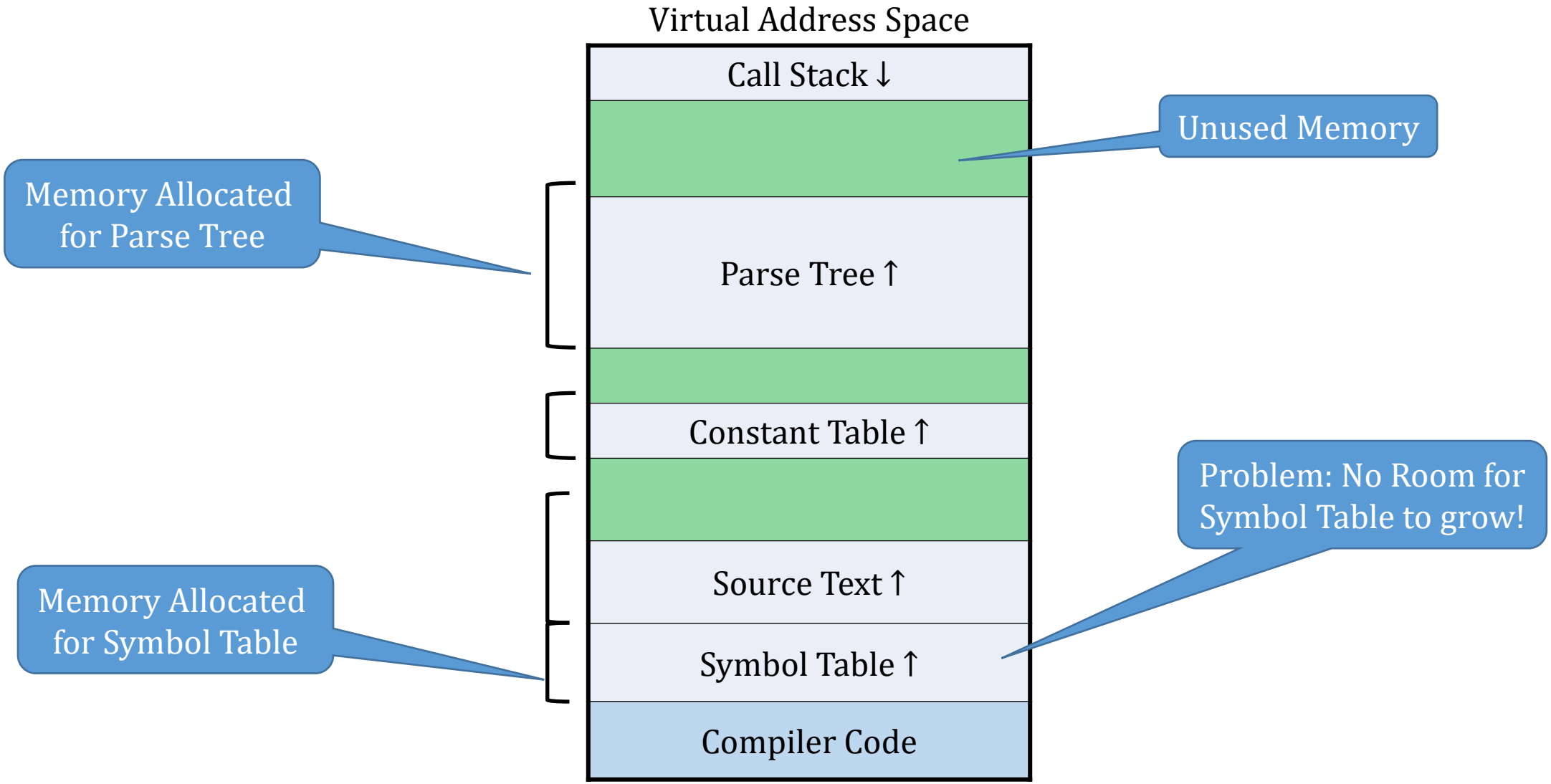
Modern Operating Systems, by Andrew Tanenbaum Chap. 3

Wikipedia Memory Segmentation

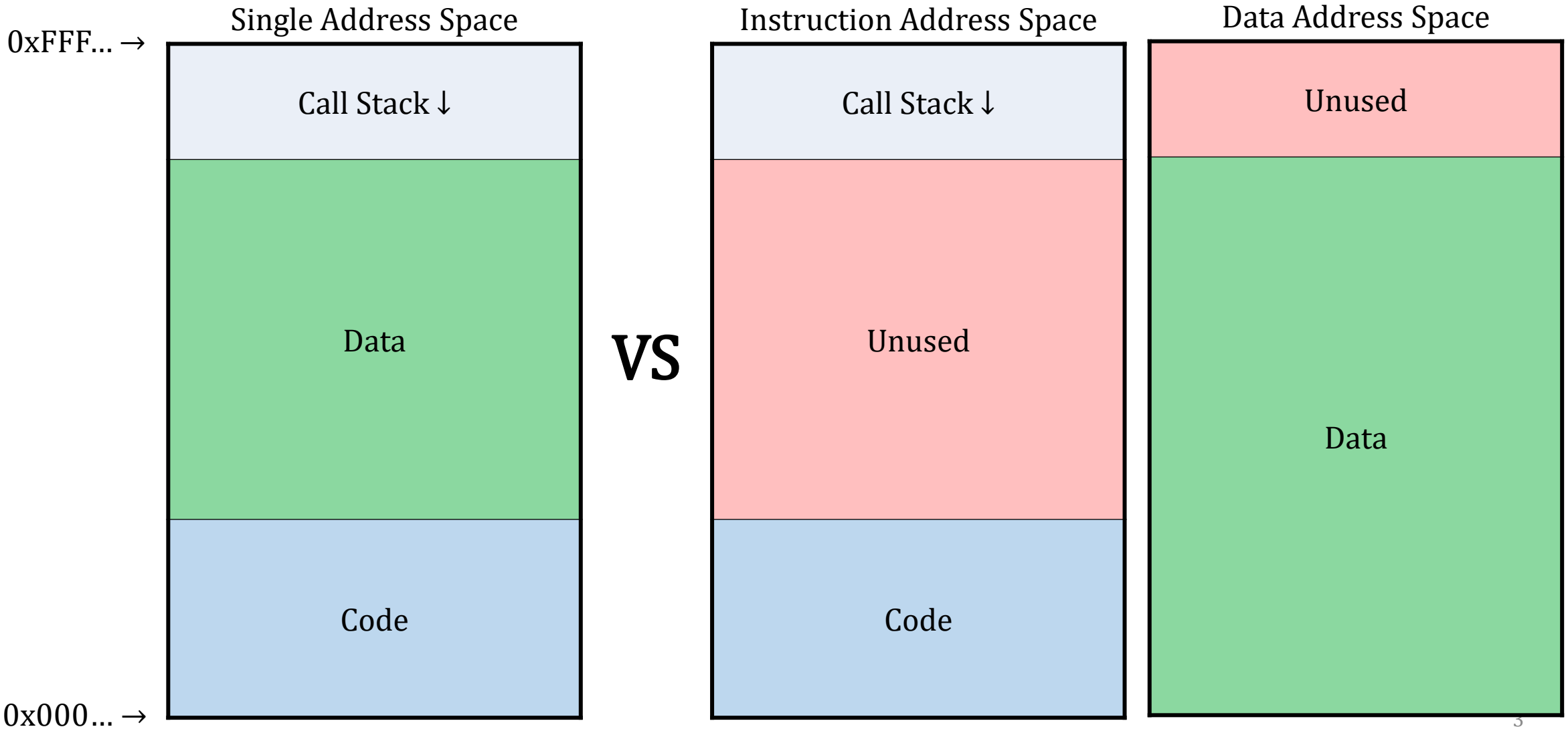
Wikipedia X86

Wikipedia Intel Memory Model

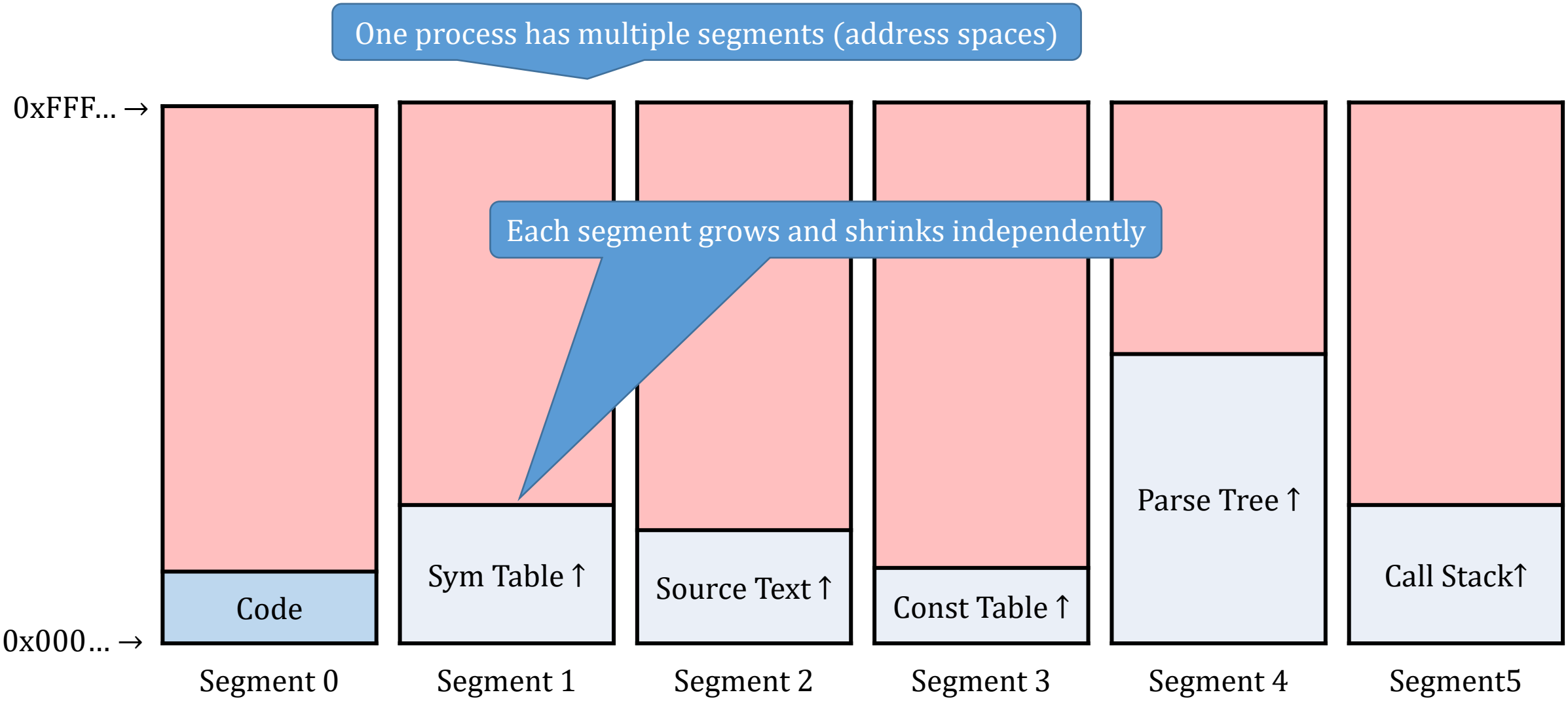
Compiler – Standard Address Space



Separate I and D Address Spaces



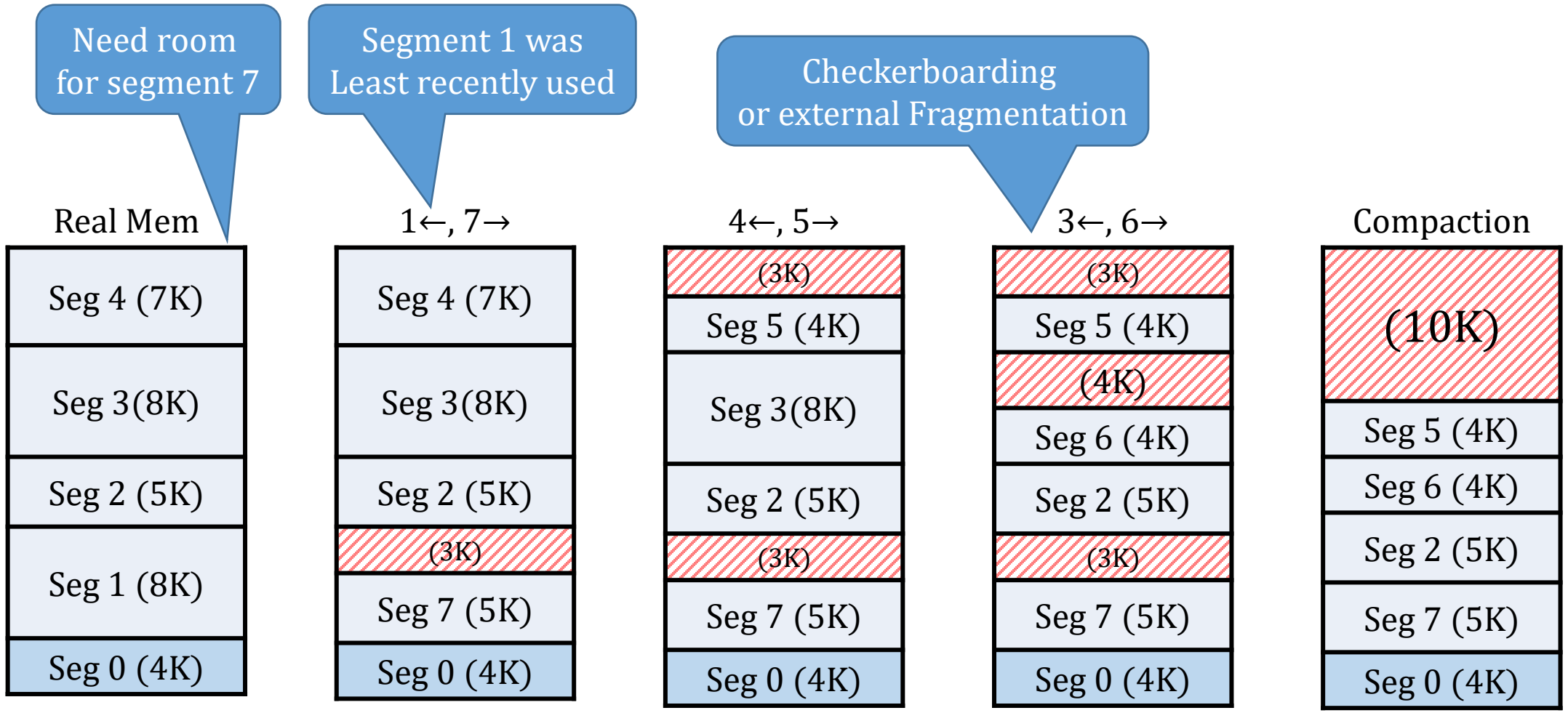
Segmentation: Unlimited Address Spaces



Comparison of Paging and Segmentation

Consideration	Paging	Segmentation
Does the programmer need to be aware of this memory management technique?	No	Yes
How many address spaces?	One	Many
Can Total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	Sort of	Yes
Can fluctuating table sizes be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	Not really	Yes
Why was this technique invented?	Large address spaces without expensive physical memory	Separate program and data and to help with sharing and protection

Pure Segmentation and Fragmentation

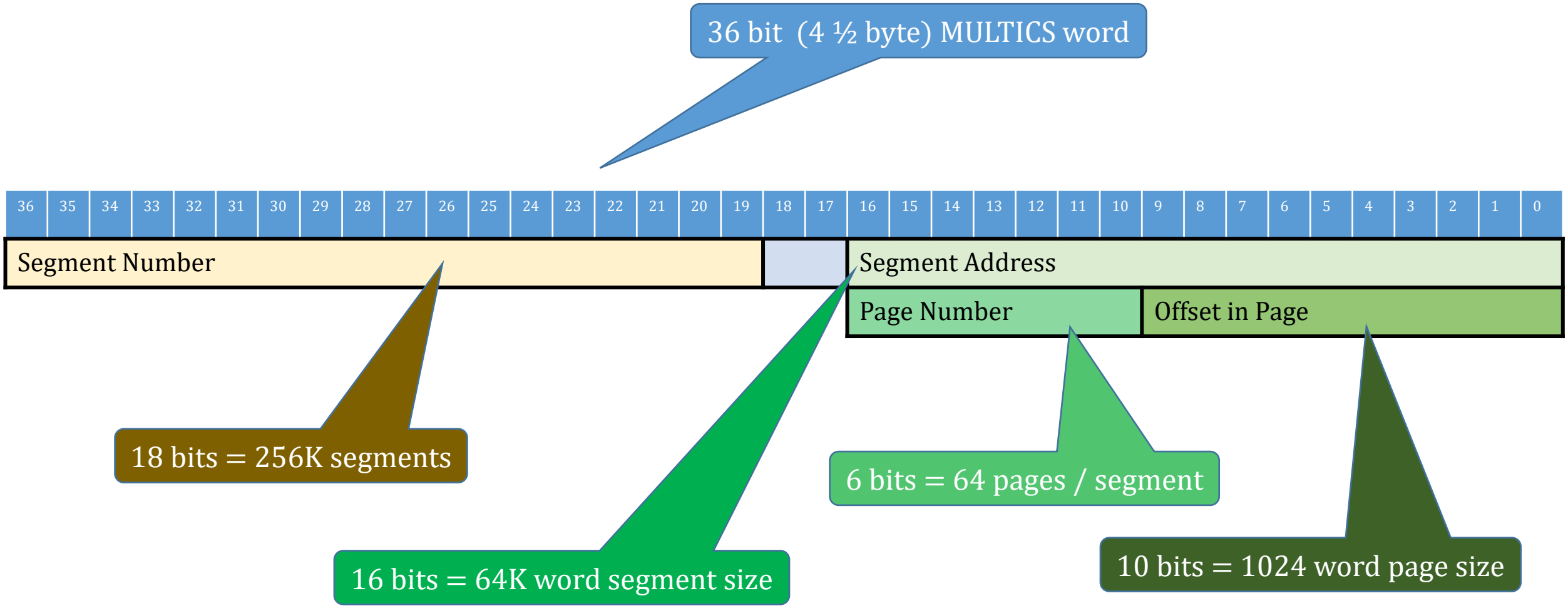


MULTICS – Paged Segmentation

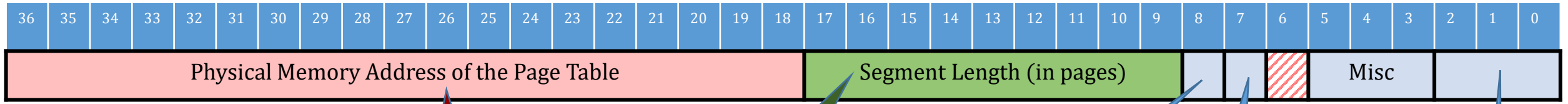
- Every process can have multiple segments (address spaces)
  - Segment number is 18 bits = 256K segments
  - Segment address is 16 bits = 64K words (words are 36 bits in Multics)
- Each segment has its own page table

Advantage	Disadvantage
Each segment can have a full 64K address space	Switching to a different segment has a high context switch penalty, even within the same process

MULTICS Virtual Address



Segment Descriptor Table Entry



Physical addresses are 24 bits but page tables are always on 64 byte boundaries, so we can pad with 6 bits of zero

9 bits =0 - 511 pages

Page size:
0=1024 words
1=64 words

Segment is paged:
At least one page of the segment and the page table are in real memory

Protection bits

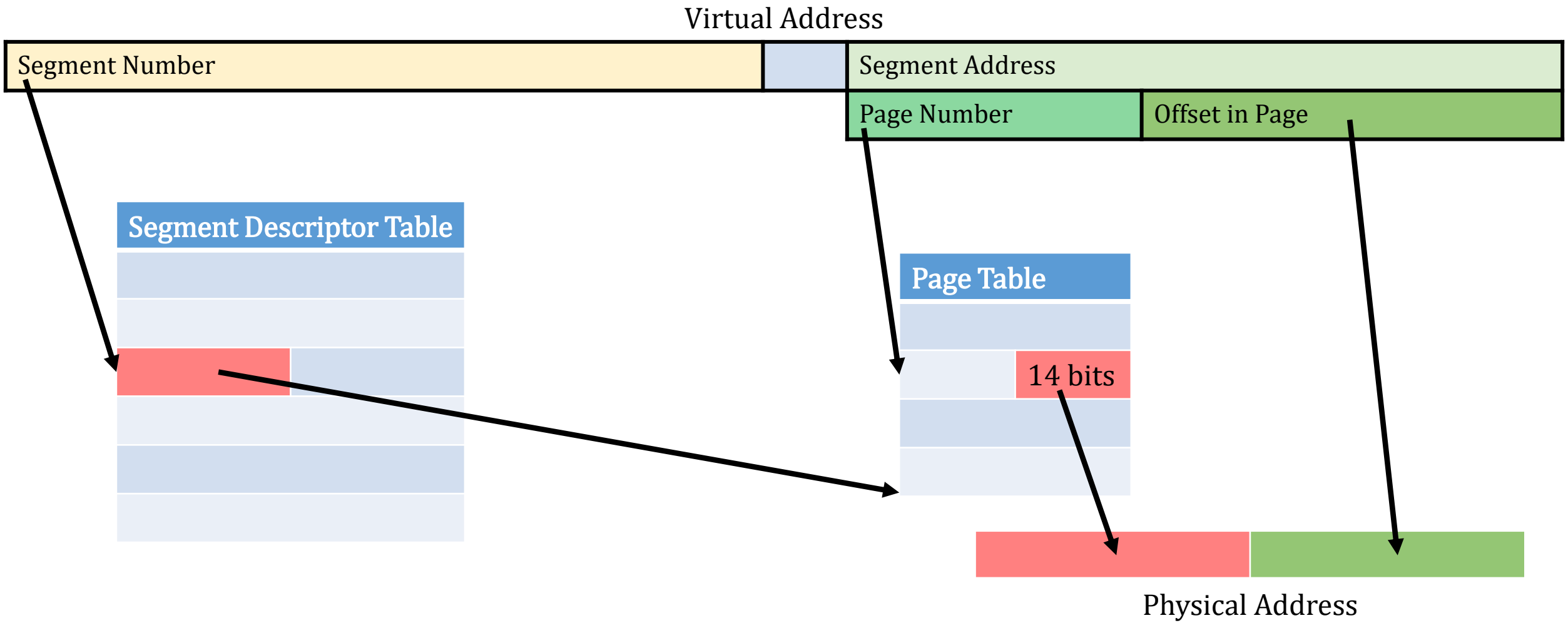
Segment Descriptor Table

- 256K Words long – one for each possible segment
  - Each word is a Descriptor Table Entry
- Segment number is an index into this table
- Kept in it's own segment
- If any part of a segment is in memory, then it's page table is in memory
  - Page table contains 64 words, and fits in a single 1024 word page

MULTICS MMU Processing

1. Use the segment number from the virtual address to find the segment descriptor (may require paging in parts of the segment descriptor table)
2. If the segment is not in memory, issue a segment fault
3. Check protection bits and issue a protection fault if needed
4. Read the page table for this segment using the page number from the virtual address
5. If the page is not in memory, issue a page fault
6. Add the offset to the physical memory start of page to get the physical address

MULTICS MMU

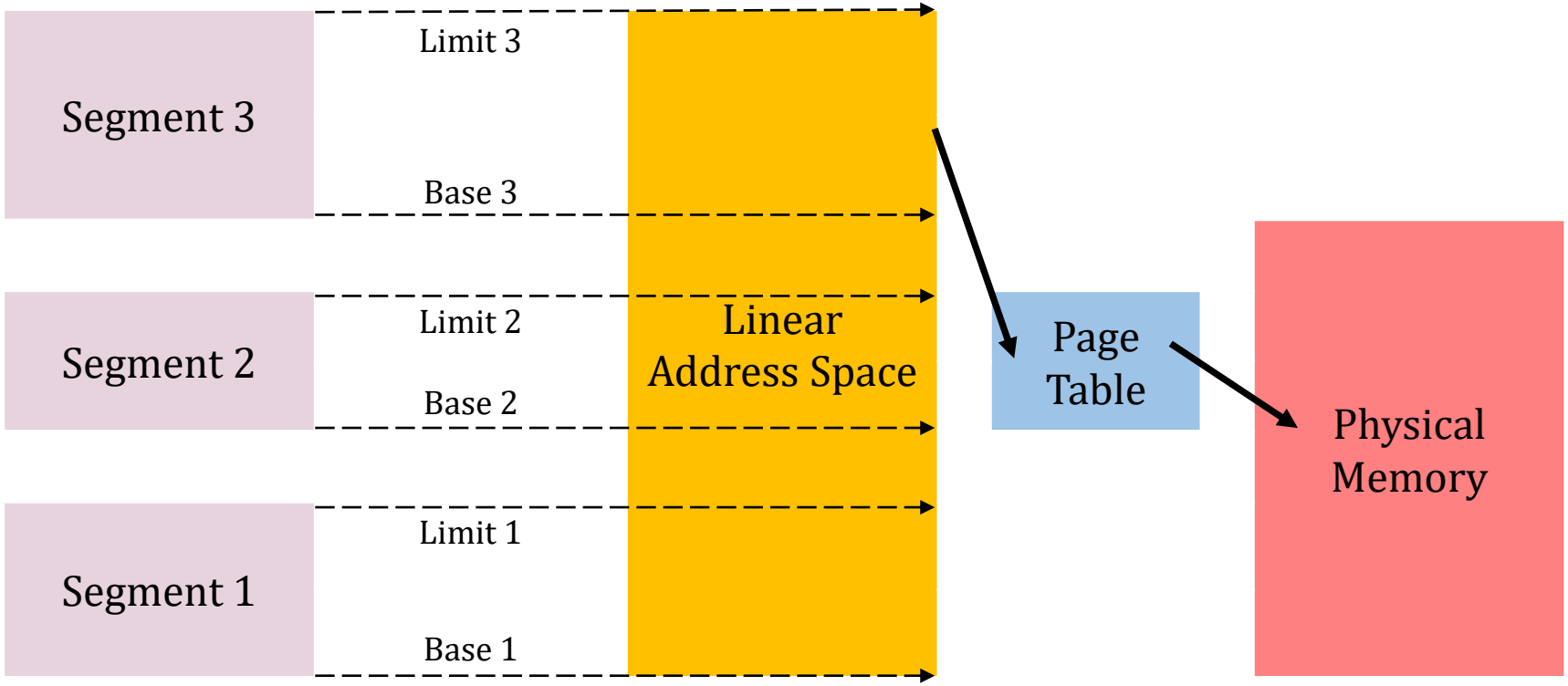

Virtual Address
Segment Number
Segment Address
Page Number
Offset in Page
Segment Descriptor Table
Page Table
14 bits
Physical Address

MULTICS File I/O

- In MULTICS, every file was memory mapped
- "Open" returned the segment that contained the file
  - Segments are 64K words = 256K+ bytes (288K w/ no parity)
  - Bigger files needed "Multi-Segment" handling
- Very nice to get RAM memory speeds for file I/O!
- Easy to package data in a file with code that works on a segment

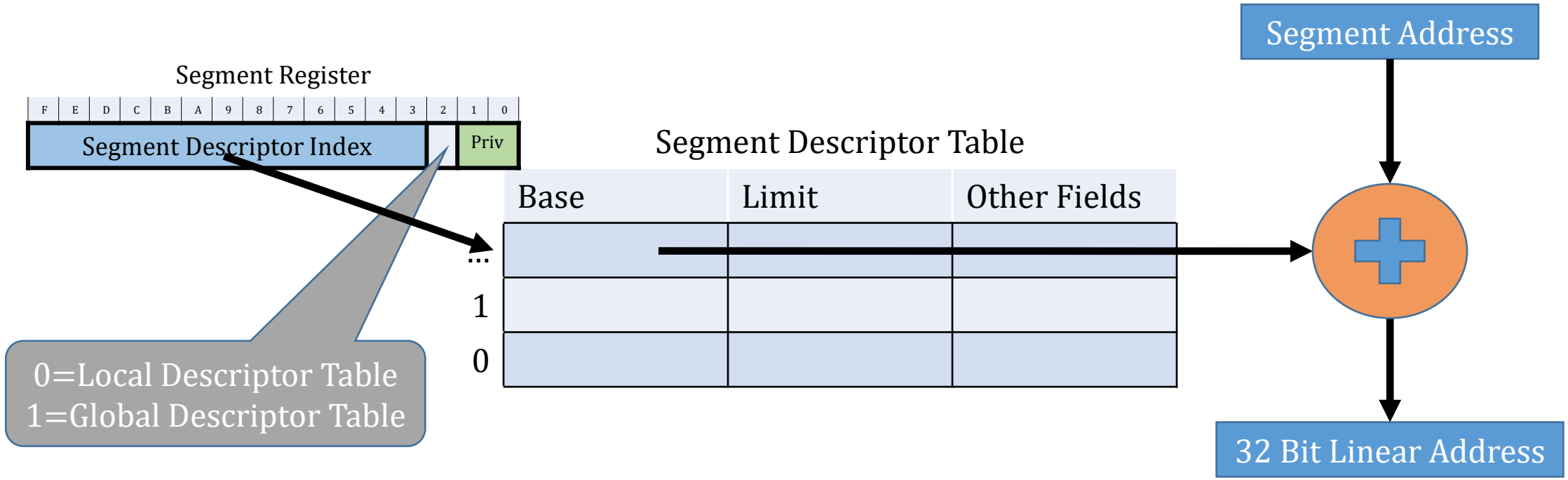
Pentium – Paged Segmentation

- Each process can have multiple segments
- Multiple segments map to one linear address space
- Linear address space has "one" page table (two stage)

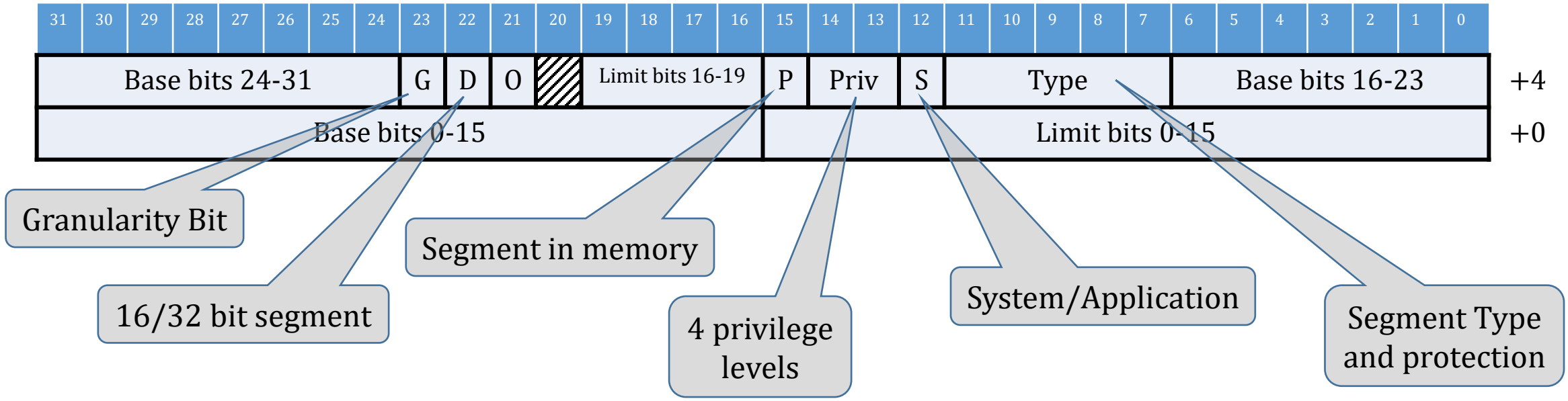


Pentium Paged Segmentation

- Segment index contained in the Segment Register
- Index into the Segment Descriptor Table



Segment Descriptor Table Entry Details



This is a Code Segment Descriptor Table Entry
Data Segment Descriptor Table Entries are slightly different.

Pentium Page Tables