Memory: Segmentation

Instructor: Hengming Zou, Ph.D.

In Pursuit of Absolute Simplicity 求于至简，归于永恒

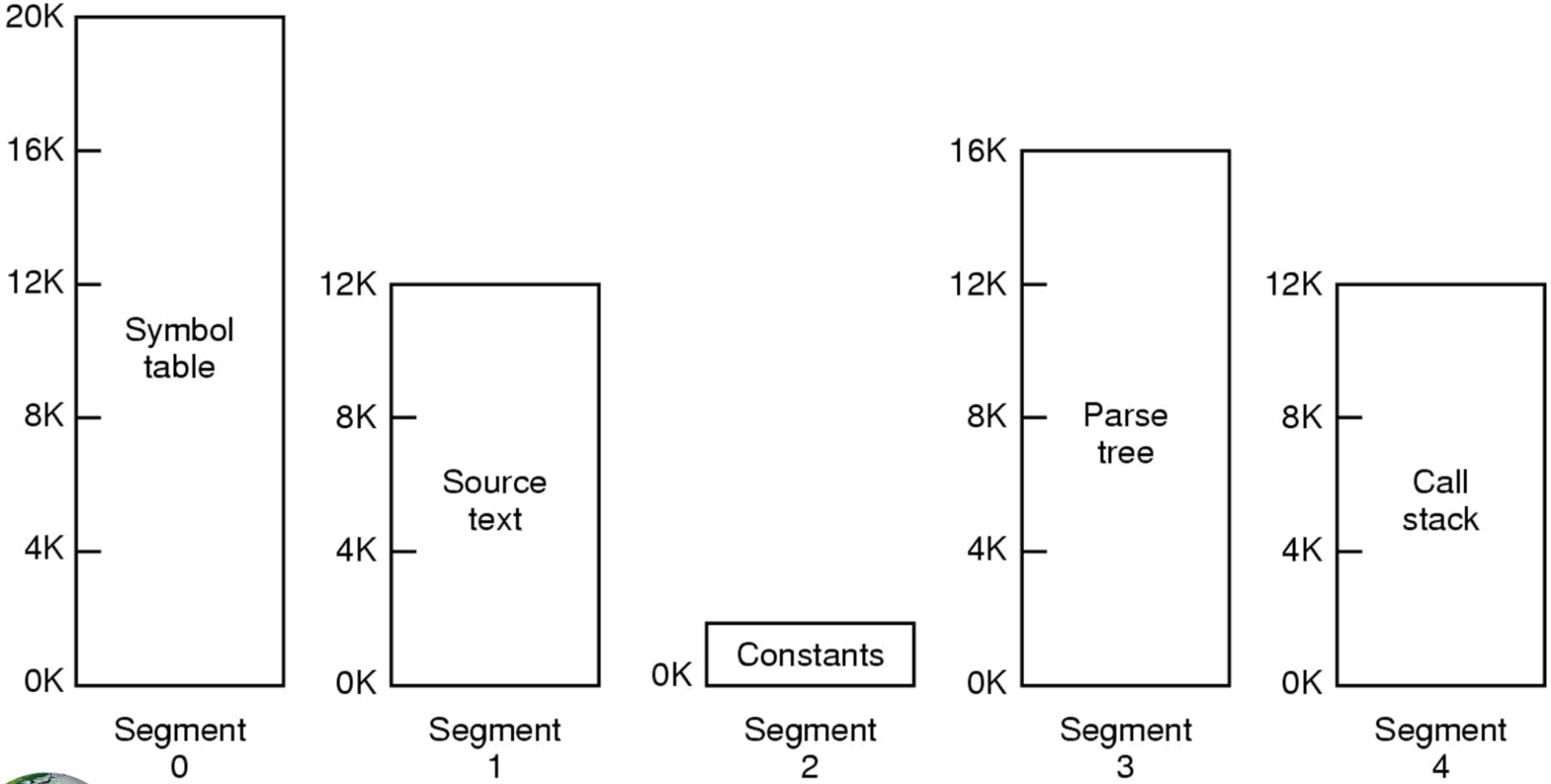
Content

- Logical segmentation
- Segmentation with paging
- Case study: Multics

(Logical) Segmentation

- Segmentation divides both physical and virtual memory into segments
  - i.e. regions of contiguous memory space
- Each segment is dedicated to one or more sections of a process
  - i.e. logical unit of separation, such as data, code, stack, etc.
- The pure segmentation use entire process

Segmentation



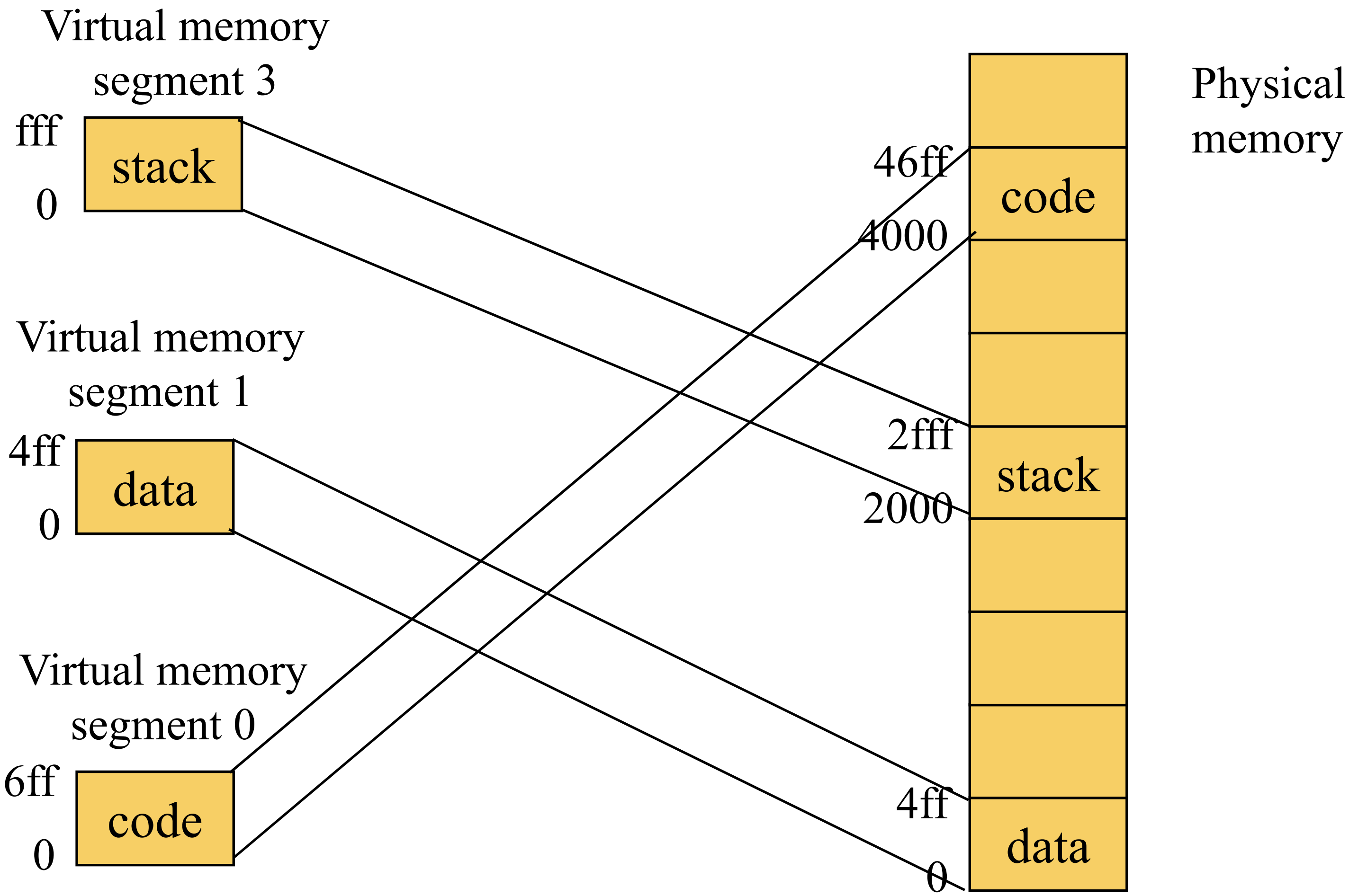
Allows each table to grow or shrink, independently

Segmentation

- Let's generalize this to allow multiple segments
  - described by a table of base & bound pairs

Segment #	Base	Bound	Description
0	4000	700	Code segment
1	0	500	Data segment
2	Unused		
3	2000	1000	Stack segment

Segmentation



Segmentation

- Note that not all virtual addresses are valid
  - e.g. no valid data in segment 2,
    - →i.e. the process has no such specific segment
  - no valid data in segment 1 above 4ff
- Valid means the region is part of the process's virtual address space
  - Could be multiple set of virtual address spaces
- Invalid means this virtual address is illegal for the process to access
  - Accesses to invalid address will cause OS to take corrective measures
  - usually a core dump

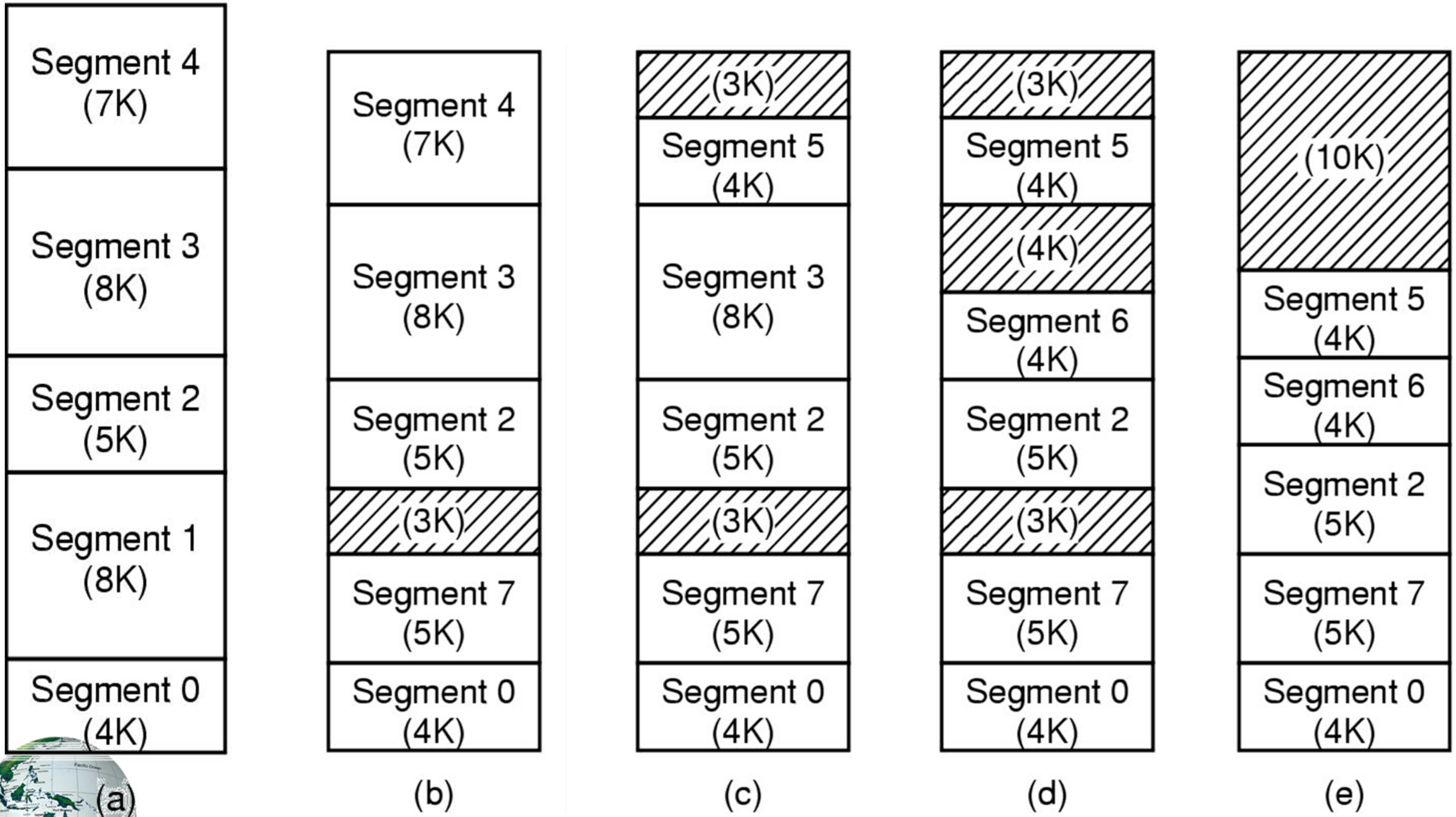
Segmentation

- Protection:
  - different segments can have different protection
  - e.g. code can be read-only (allows inst. fetch, load)
  - e.g. data is read/write (allows fetch, load, store)
- In contrast, pure segmentation gives same protection to entire space

Segmentation

- In segmentation, a virtual address takes the form:
  - (virtual segment #, offset)
- Could specify virtual segment # via
  - The high bits of the address,
  - Or a special register,
  - Or implicit to the instruction opcode

Implementation of Segmentation



Segmentation

- What must be changed on a context switch?
  - Segmentation table or descriptor segment

Pros and Cons of Segmentation

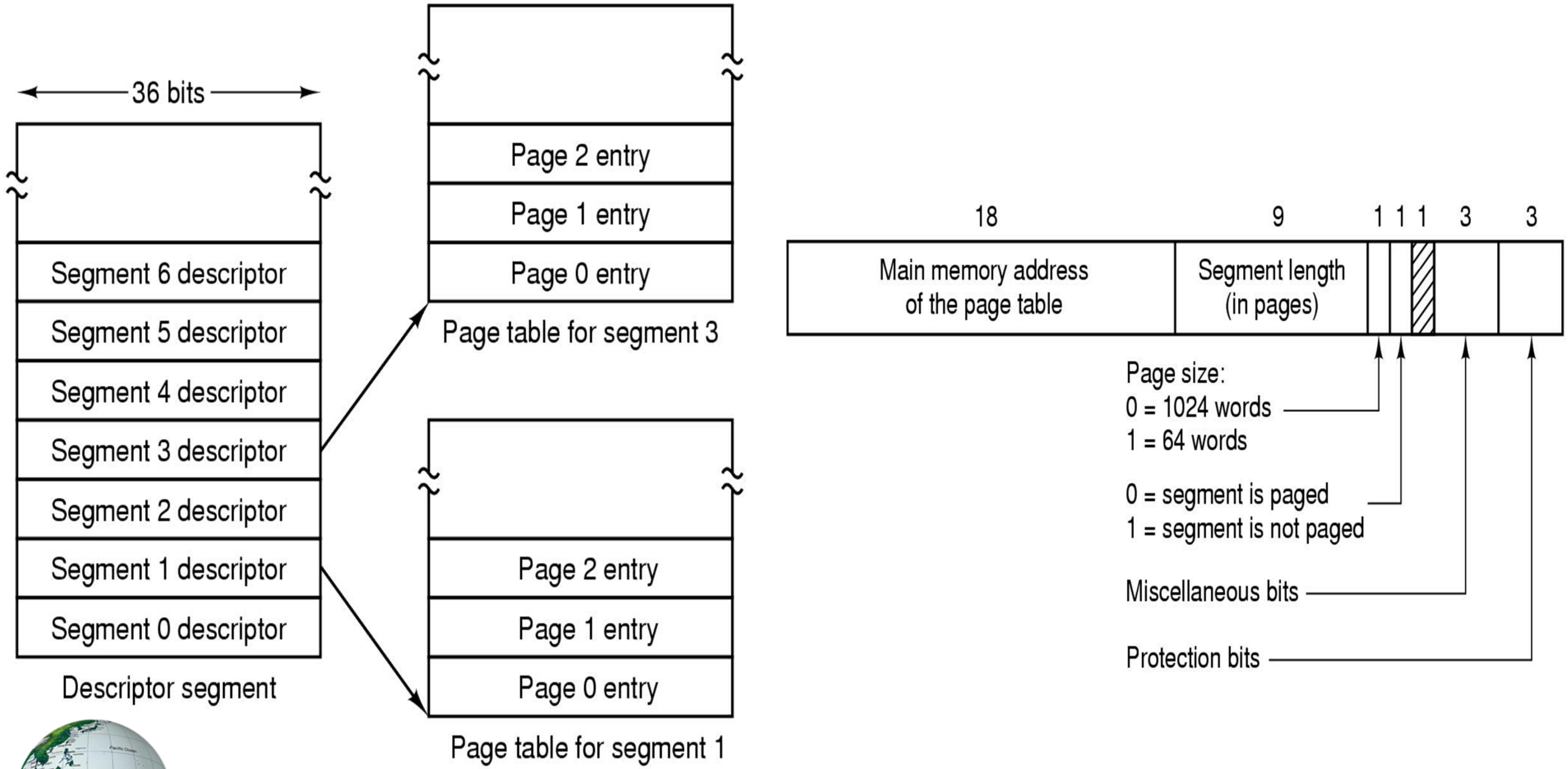
- + easy to share whole segments without sharing entire address space
- + avoid collision within virtual address space
- + works well for sparse address spaces
  - with big gaps of invalid areas
- - complex memory allocation
- - external fragmentation

Segmentation

- How to make memory allocation easy and
  - But still keeps the advantages of segmentation?

- SEGMENTATION WITH PAGING!
  - Divide program into logical segmentations
  - Use paging within each segment

Segmentation with Paging



Segmentation with Paging

- Descriptor segment points to page tables
- Page tables points to physical frames
- MULTICS use this method

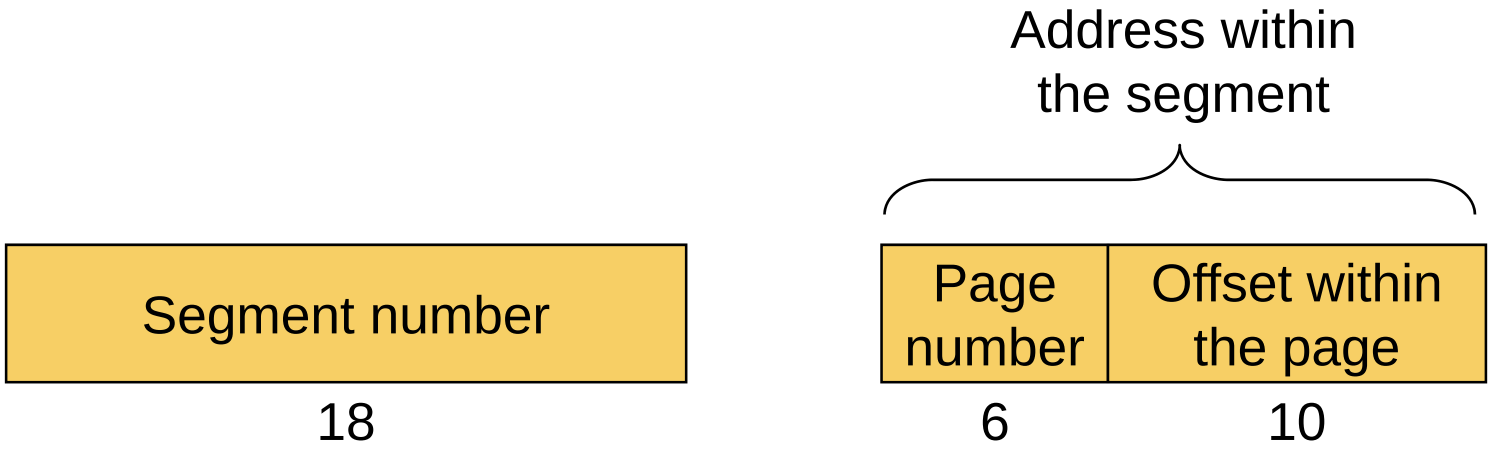
Compare Paging and Segmentation

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

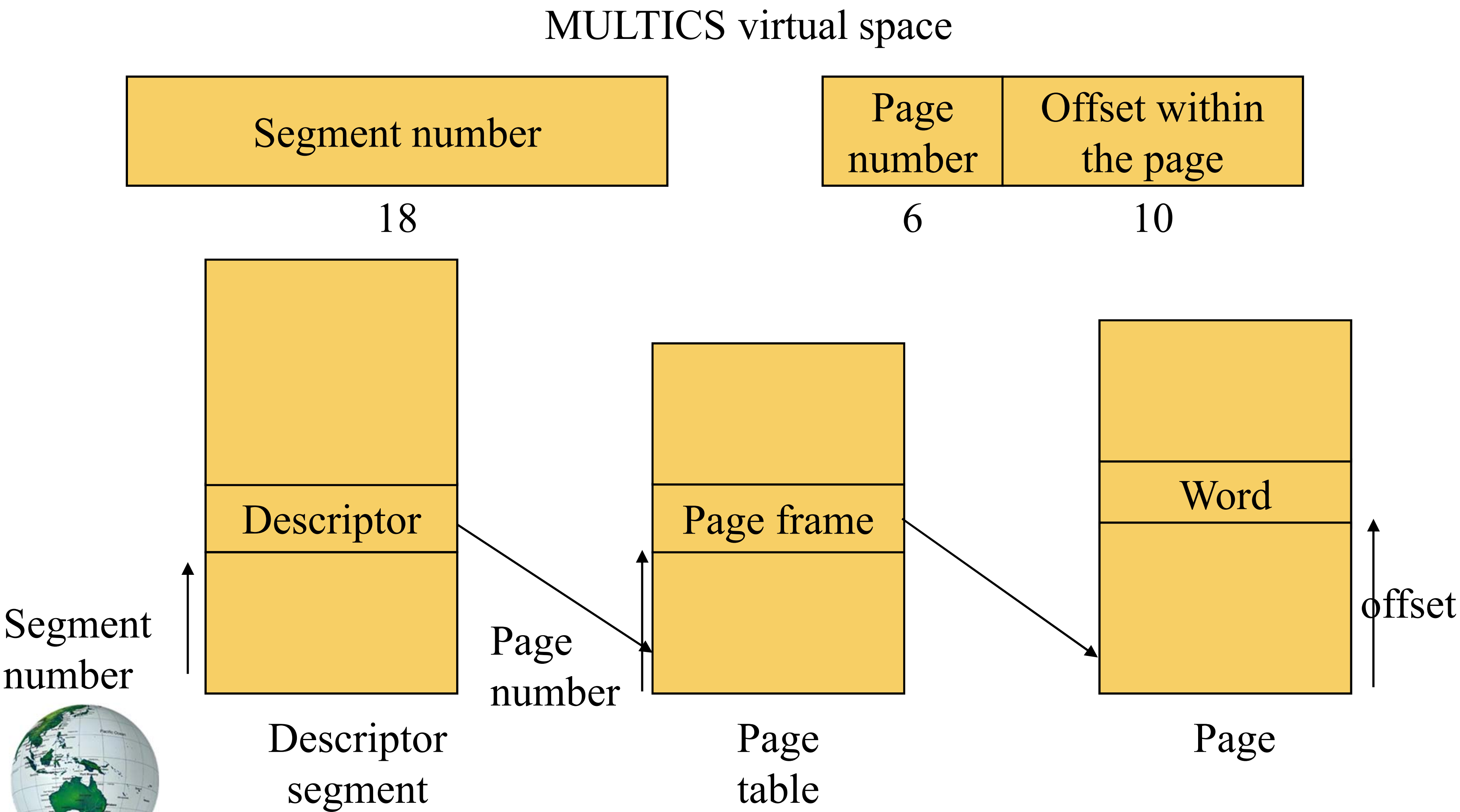
SP Case Study

SP Example: MULTICS

- A 34-bit MULTICS virtual address



SP Example: MULTICS



SP Example: MULTICS TLB

Comparison field					Is this entry used?
Segment number	Virtual page	Page frame	Protection	Age	
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

SP Example: MULTICS

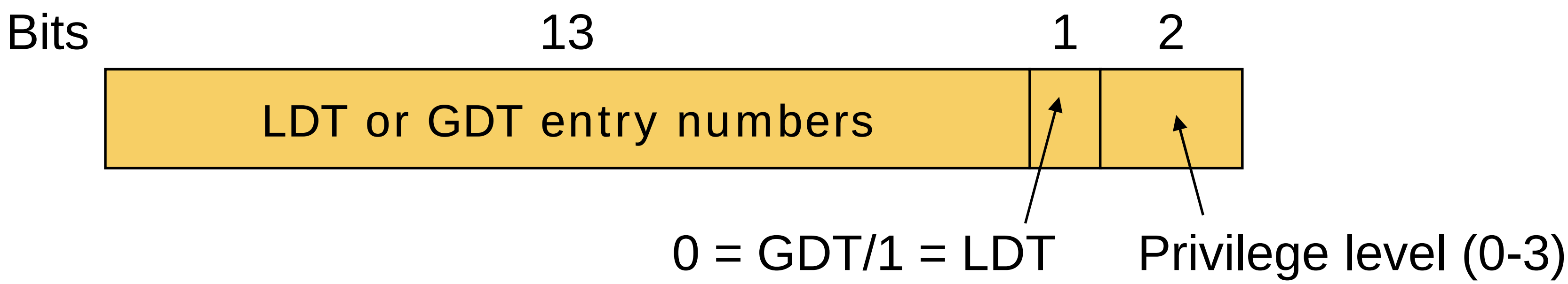
- Do not show this slide
- Simplified version of the MULTICS TLB
- Existence of 2 page sizes makes actual TLB more complicated

SP Example: Pentium

- Pentium virtual memory contains two tables:
- Global Descriptor Table:
  - Describes system segments, including OS
- Local Descriptor Table:
  - Describes segments local to each program

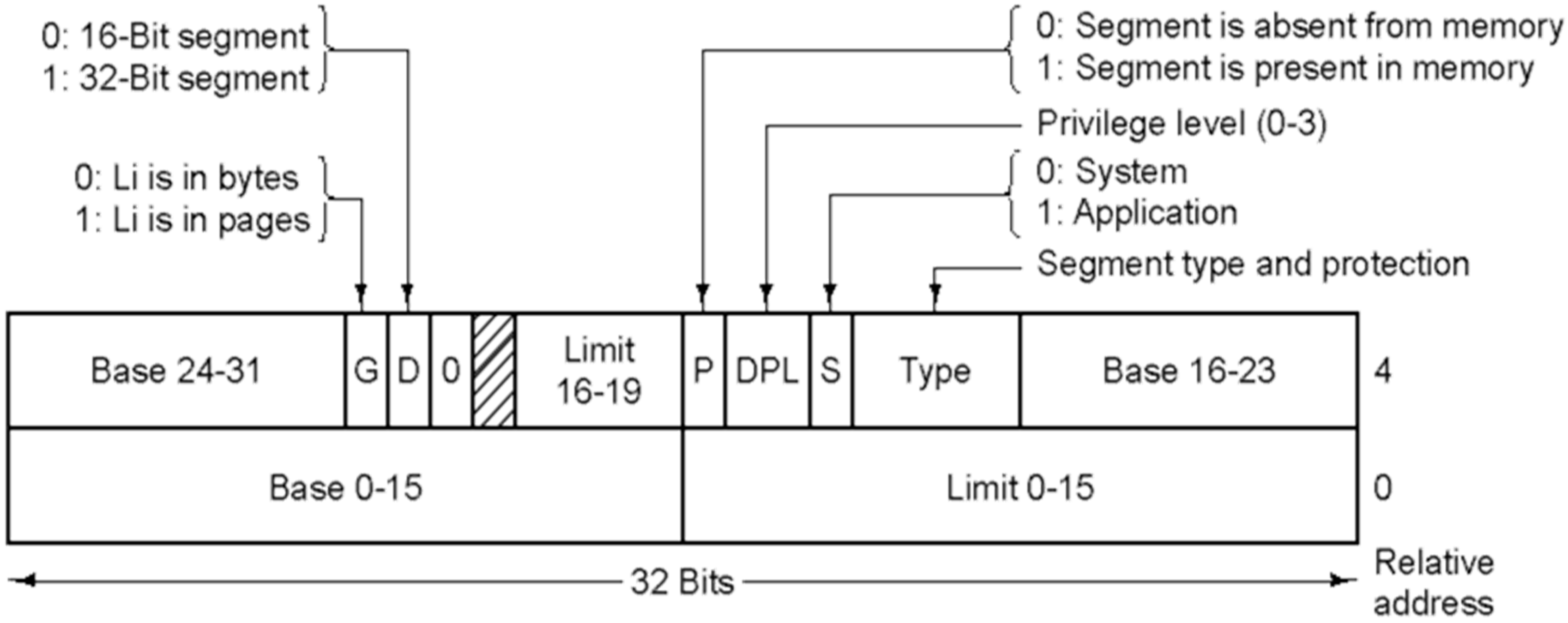
SP Example: Pentium

- Pentium selector contains a bit to indicate if the segment is local or global



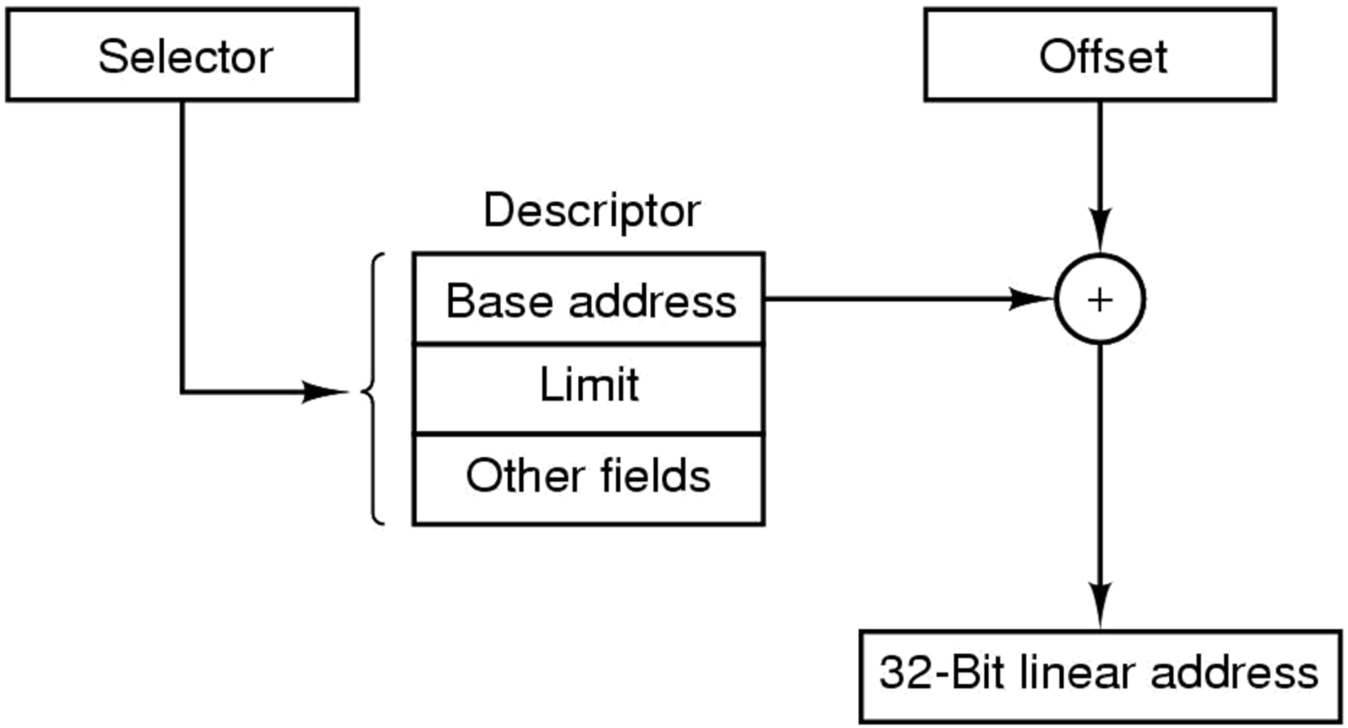
A Pentium selector

SP Example: Pentium



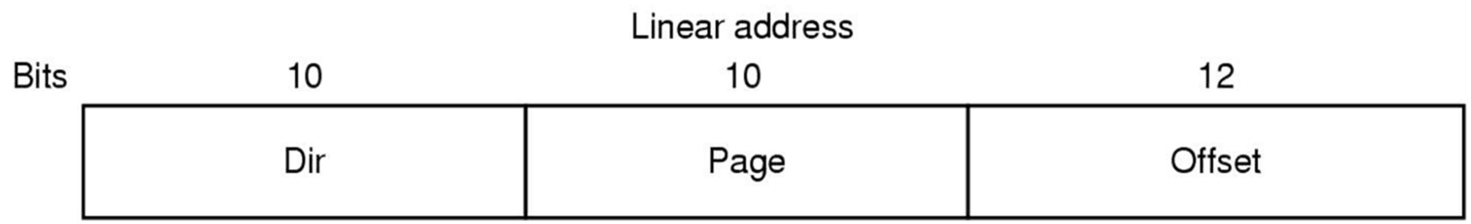
Pentium code segment descriptor (Data segments differ slightly)

SP Example: Pentium

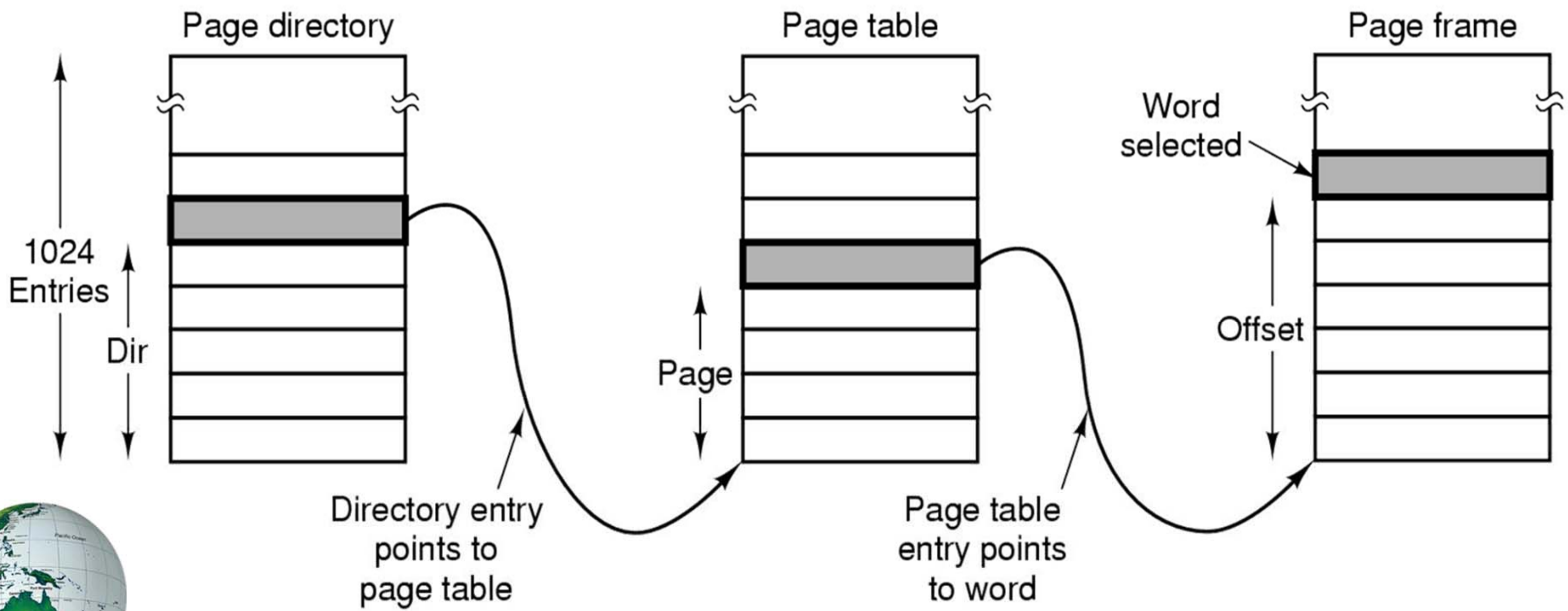


Conversion of a (selector, offset) pair to a linear address

Pentium Address Mapping

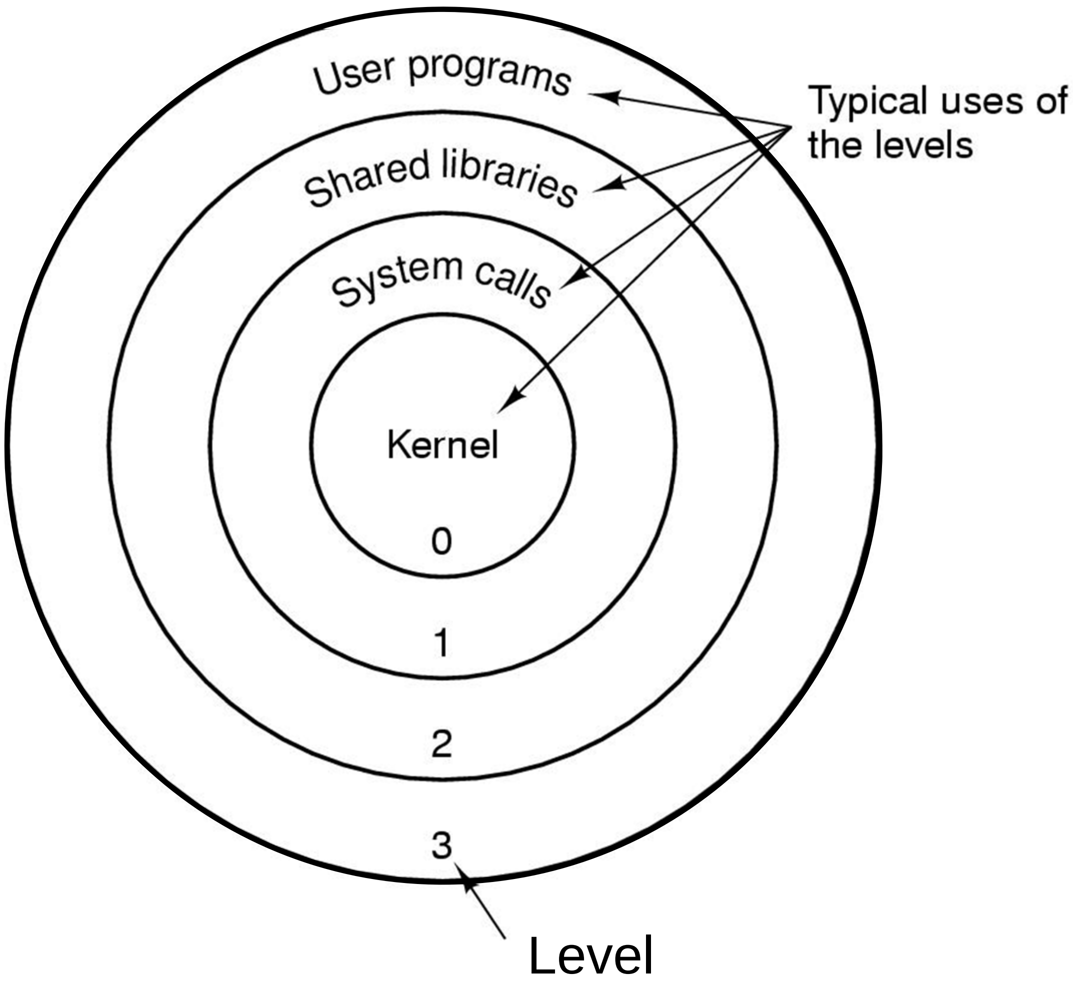


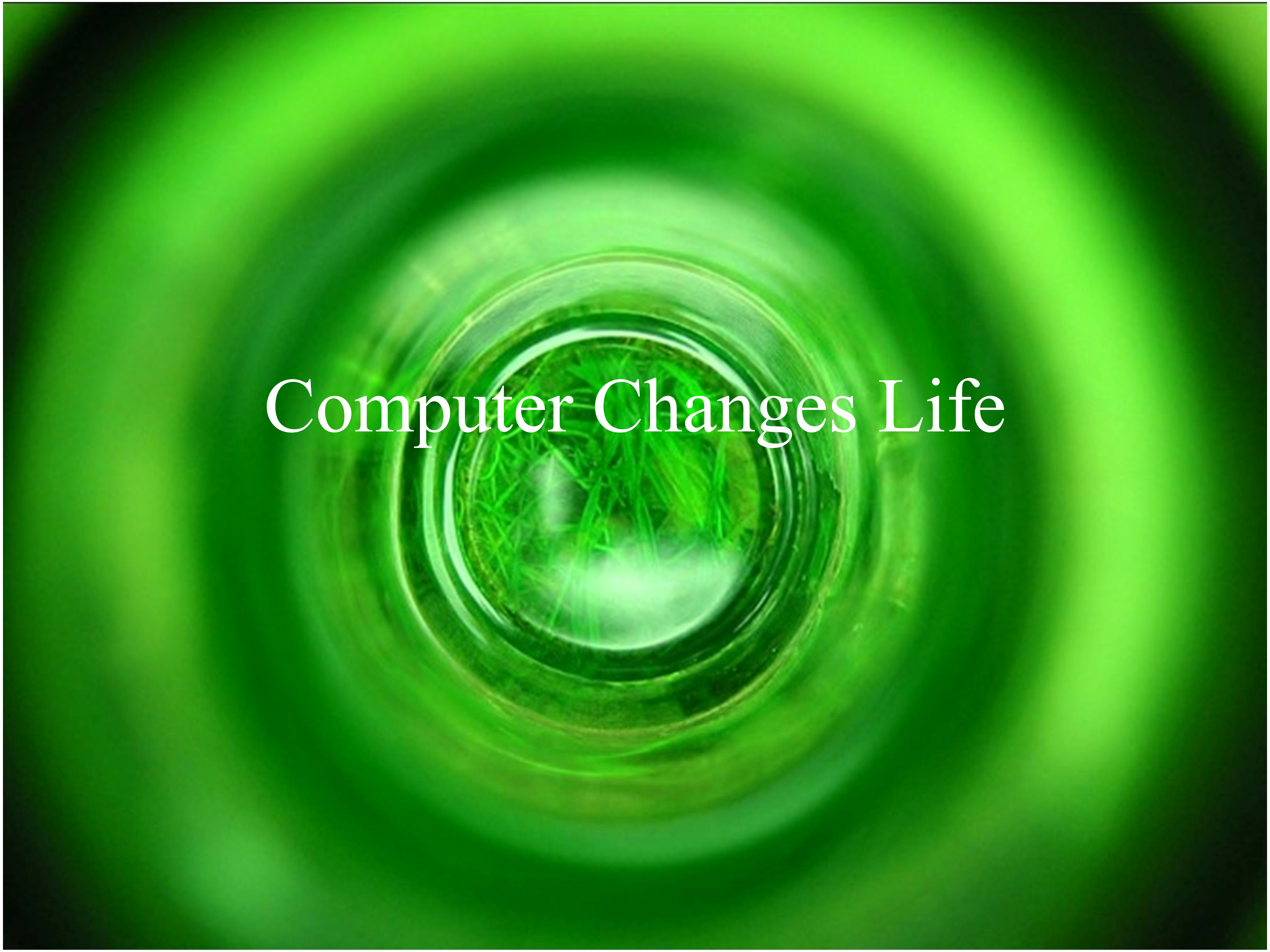
(a)



(b)

Protection on the Pentium




Computer Changes Life