

Memory (Part II):

Mechanisms for Memory Management: Paging & Segmentation

Professor Travis Peters
CSCI 460 Operating Systems
Fall 2019

Some slides & figures adapted from Stallings instructor resources.

Some slides adapted from Adam Bates's F'18 CS423 course @ UIUC
<https://courses.engr.illinois.edu/cs423/sp2018/schedule.html>

Goals for Today

Learning Objectives

- Understand basics of memory management, including
 - memory partitioning and common techniques
 - paging and segmentation — what they are, and their relative advantages and disadvantages
- Understand basics of loading and linking

Announcements

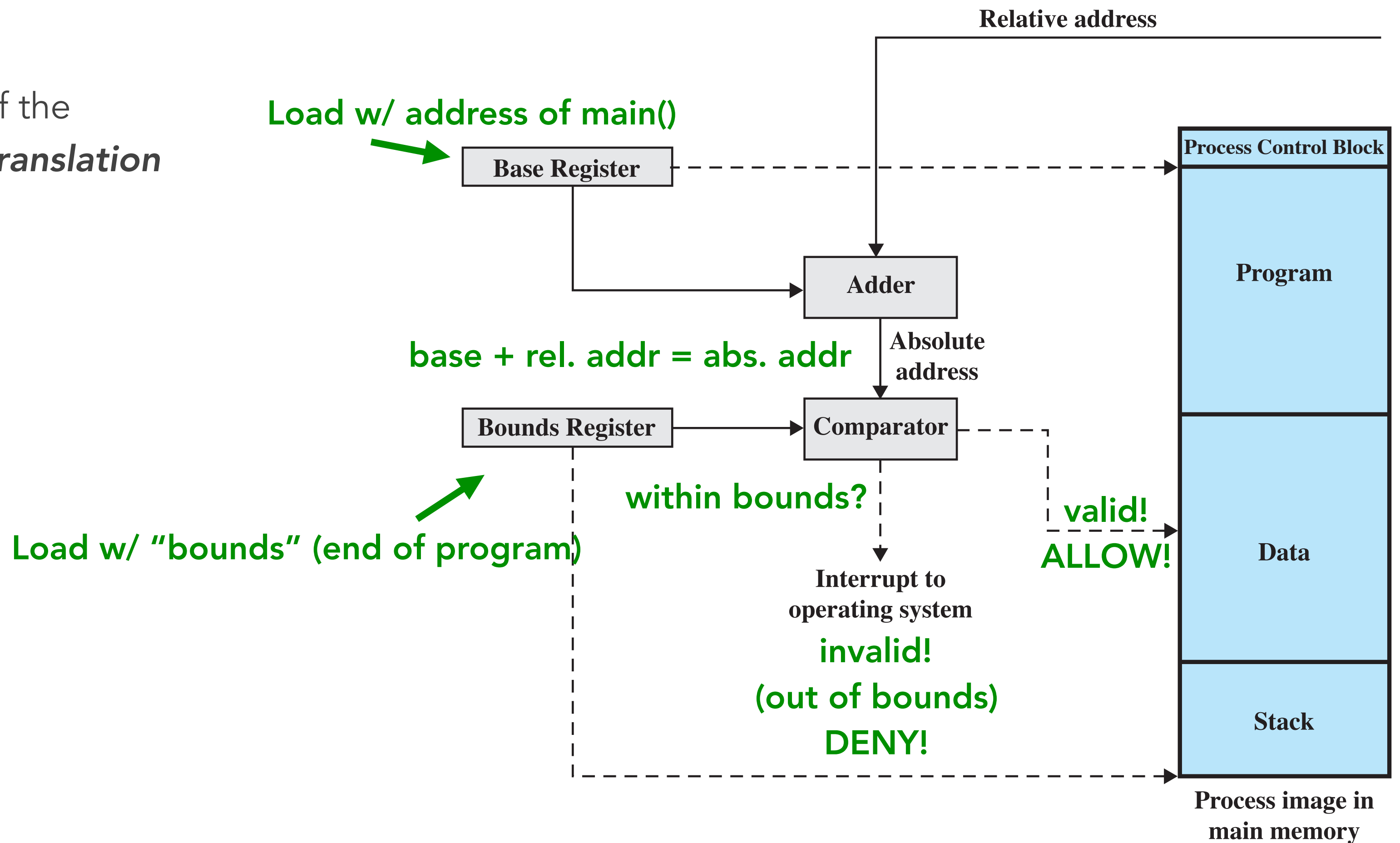
- Use Google Sheet to share info about your project:

https://docs.google.com/spreadsheets/d/1uMk0pcho_B2v8_7t_E3S-lpsdUfhjBKsT5XdBcfPOBI/edit?usp=sharing

- PA2 posted later this week...

Relocation of Processes into Partitions

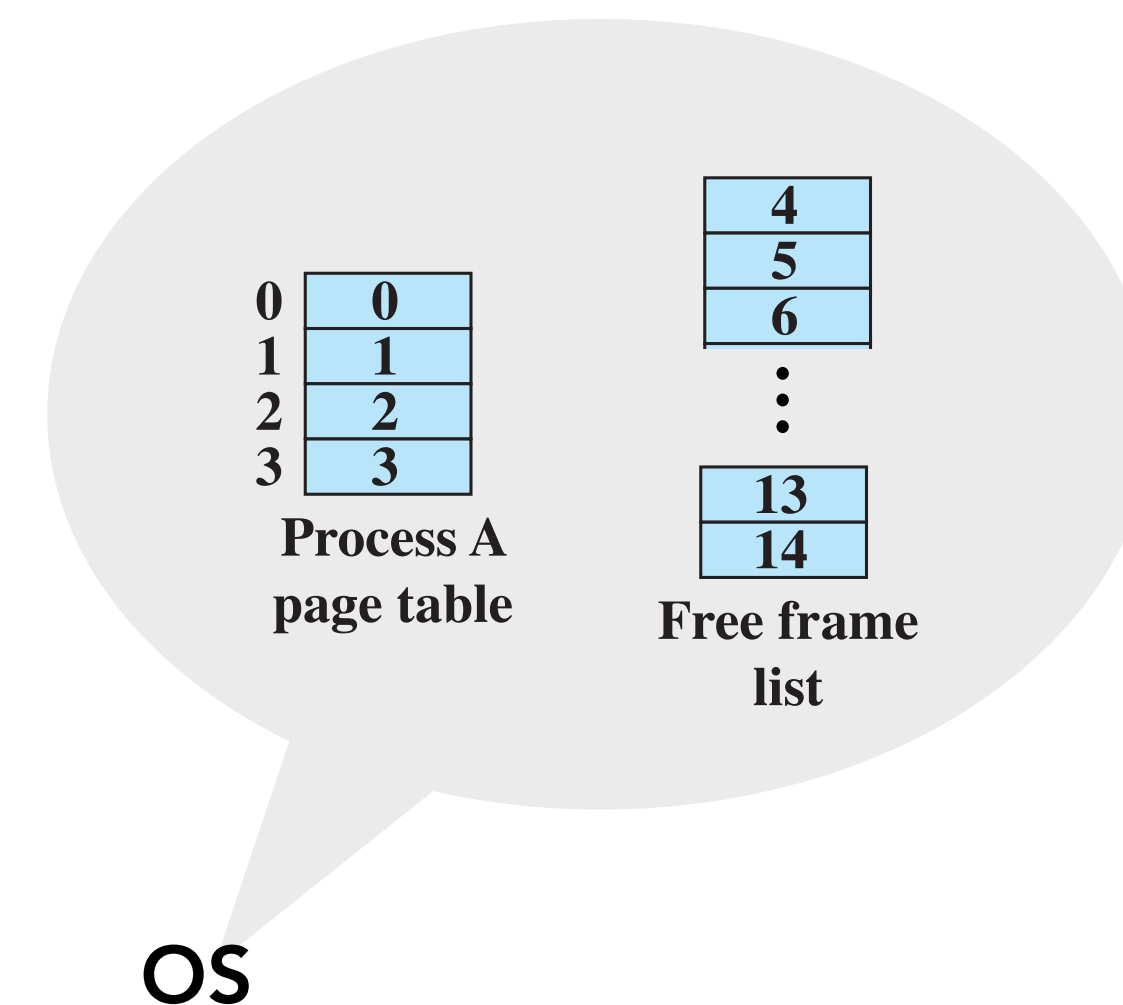
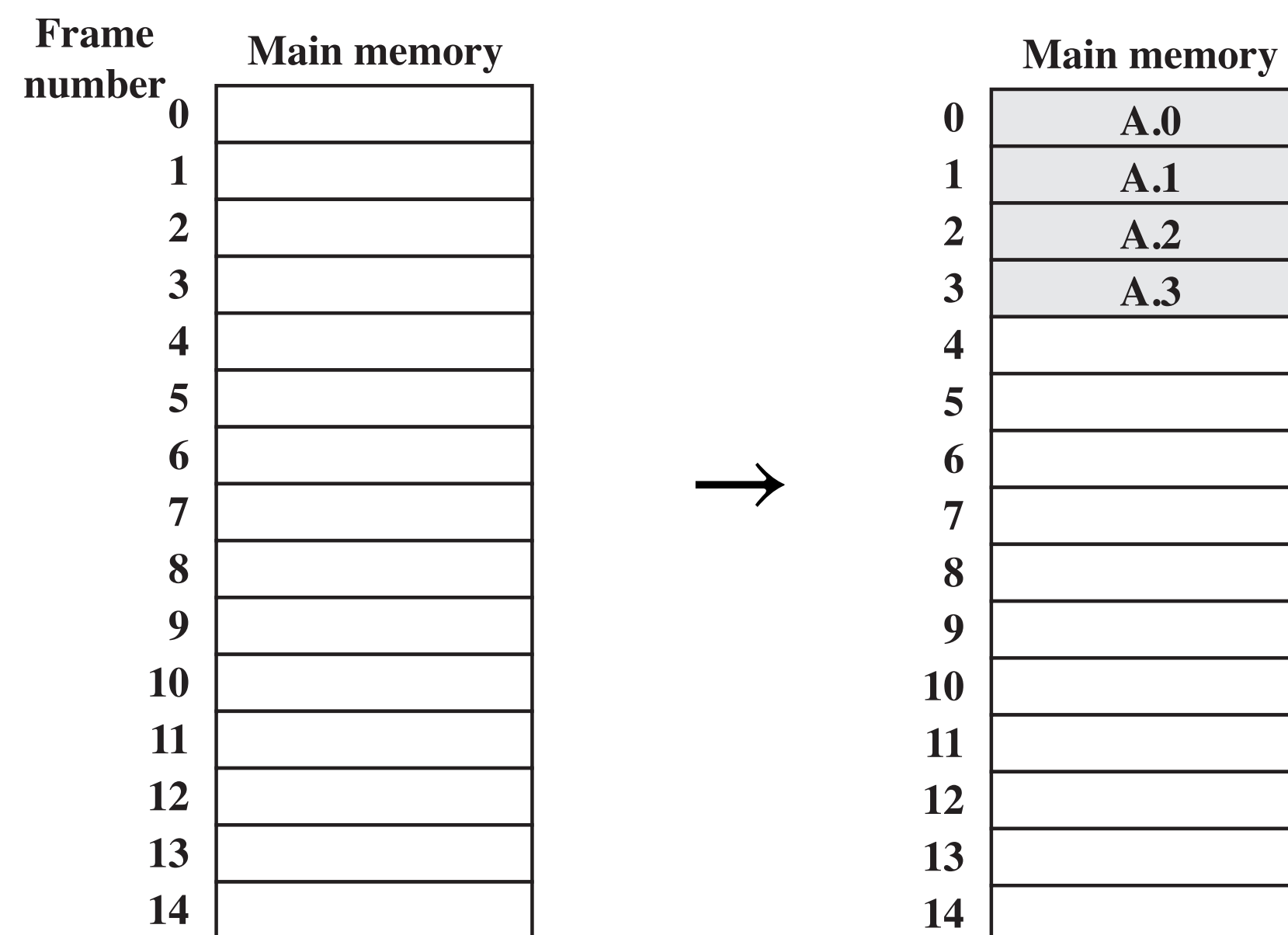
- **Logical Address**
a reference to a memory location independent of the current assignment of data to memory → need *translation*
- **Relative Address**
An example of a logical address.
Address = relative location to some known point (e.g., value in register)
- **Physical Address**
The actual location in main memory



Paging

Basic Idea

- Partition main memory into **small fixed-sized chunks** of the same size
- Assign chunks of processes (**pages**) into available chunks of main memory (**frames**)
- Small processes need fewer pages; larger processes need more pages
- No more external fragmentation
- Minimal internal fragmentation → *only part of the last page of a process*



Paging — Example: Assigning Process Pages to Free Frames

Frame number	Main memory
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(a) Fifteen Available Frames

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(b) Load Process A

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	
8	
9	
10	
11	
12	
13	
14	

(c) Load Process B

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(d) Load Process C

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(e) Swap out B

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

(f) Load Process D

0	0
1	1
2	2
3	3

Process A
page table

0	—
1	—
2	—

Process B
page table

0	7
1	8
2	9
3	10

Process C
page table

0	4
1	5
2	6
3	11
4	12

Process D
page table

13
14

Free frame
list

Paging — Logical Addressing

HW assists with logical addressing when using paging — HW must know how to access page table

$n + m$ bit addresses where

- n = # bits for page number (leftmost bits)
- m = # bits for offset within page (rightmost bits)
→ $\text{PAGESIZE} = 2^m$

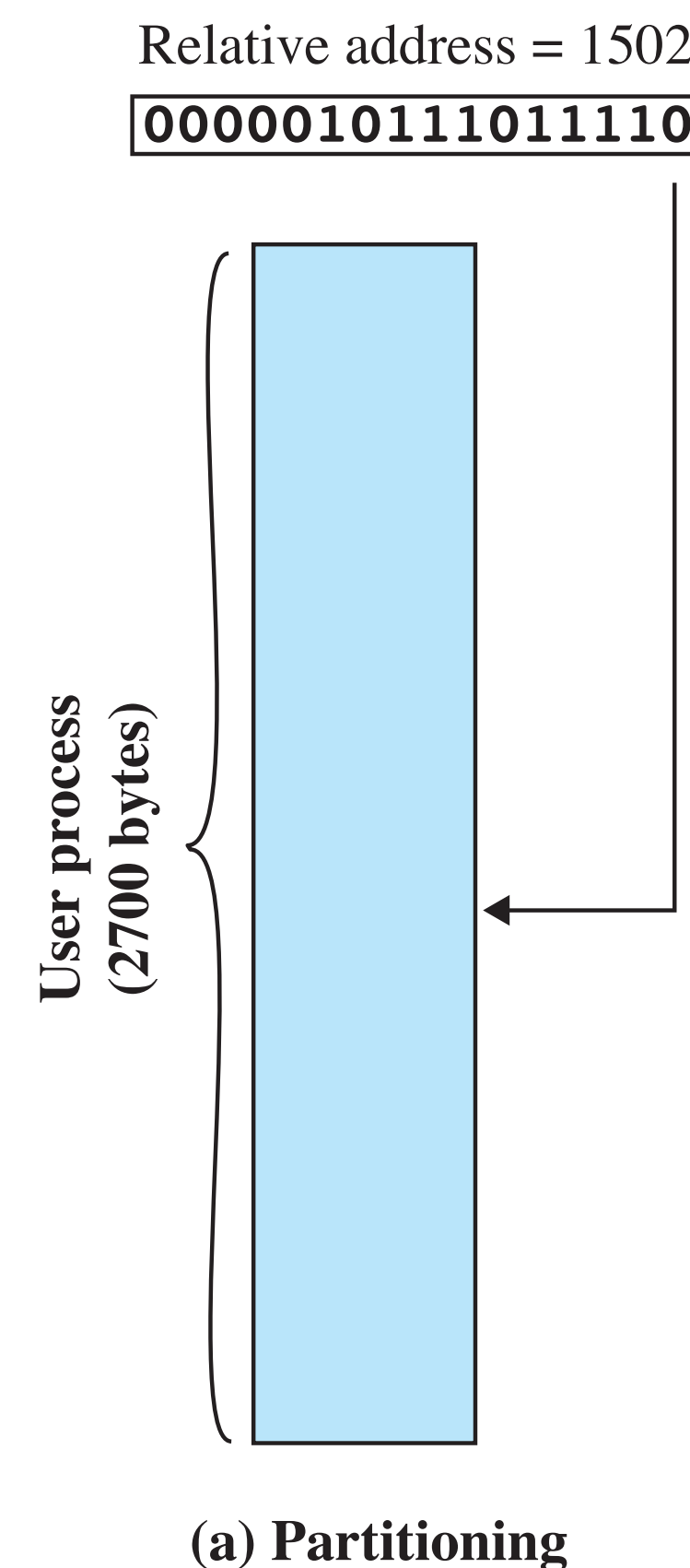
NOTE: In general, we set page/frame size to be a power of 2
→ relative address == logical address

Example:

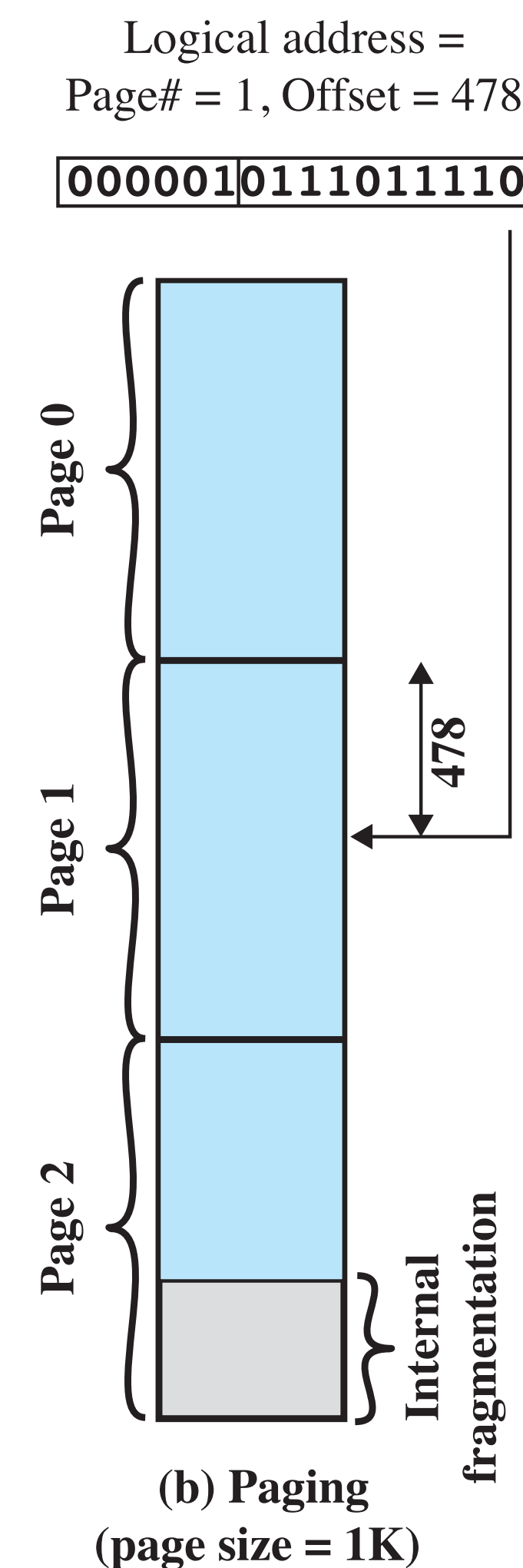
- 16-bit addresses
- Page Size = 1K (1024 Bytes)

Q: How many bits are needed to accommodate pages/frames of size 1K?
→ 10 bits needed for offset field → $1K = 2^{10}$

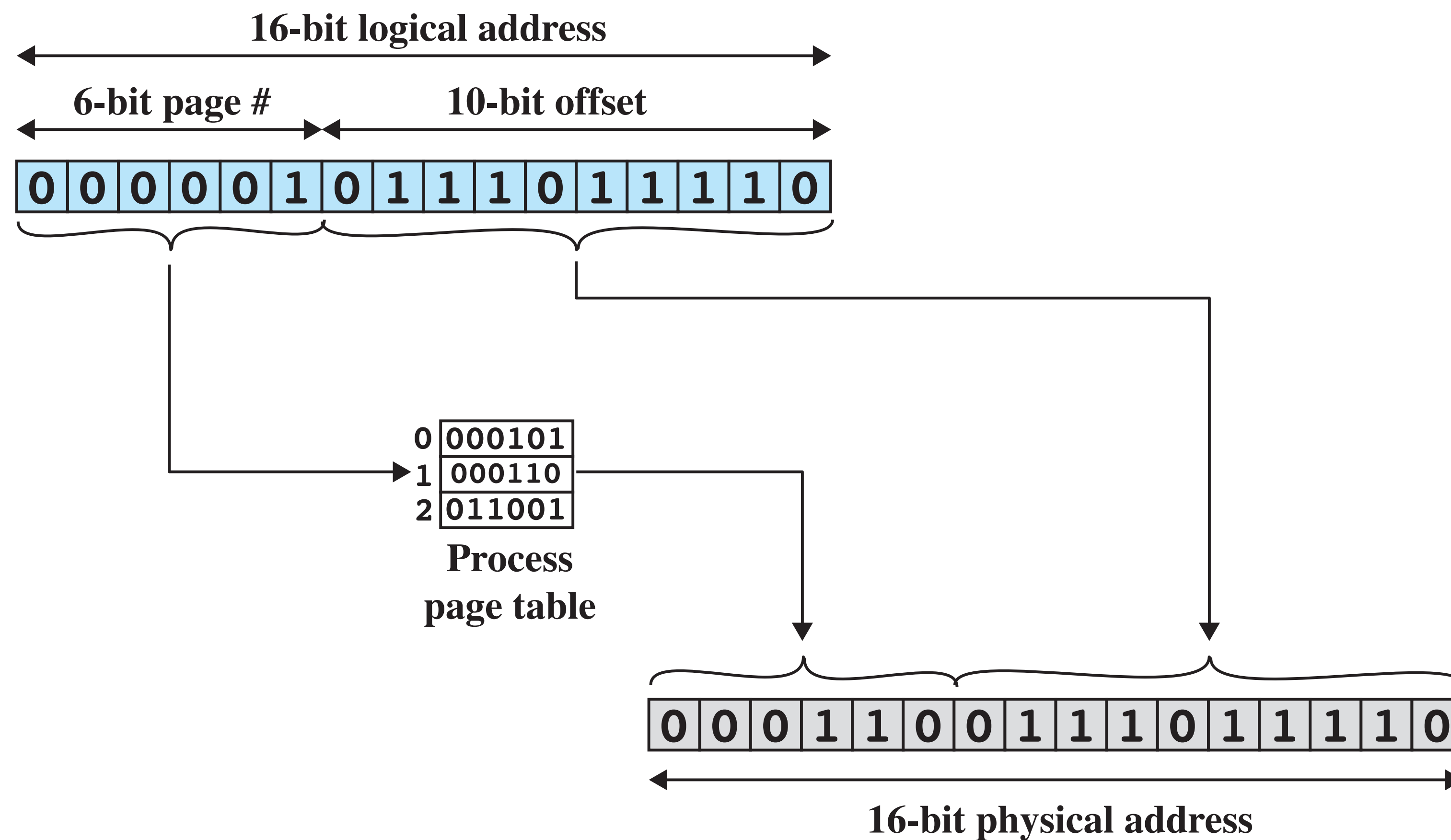
Q: How many pages are possible with 6-bits available for page numbers?
→ 6 bits left over for page # field → 64 pages



vs.



Paging — Example of Logical-to-Physical Address Translation



Segmentation

Basic Idea

- Programs can be broken up into **segments** that need not be in contiguous memory; may occupy more than one segment
- Partition main memory into **unequally-sized segments**
 - *Similar to dynamic partitioning... but not the same*
- Assign segments of processes into chunks of main memory allocated on demand
- No internal fragmentation
- Potential for external fragmentation

Segmentation — Logical Addressing

- Each segment needs to provide
 - starting address of segment
 - segment length
- Load address of segment table into register when process starts running
- $n + m$ bit addresses where
 - n = # bits for segment number (leftmost bits)
 - m = # bits for offset within segment (rightmost bits)

Example:

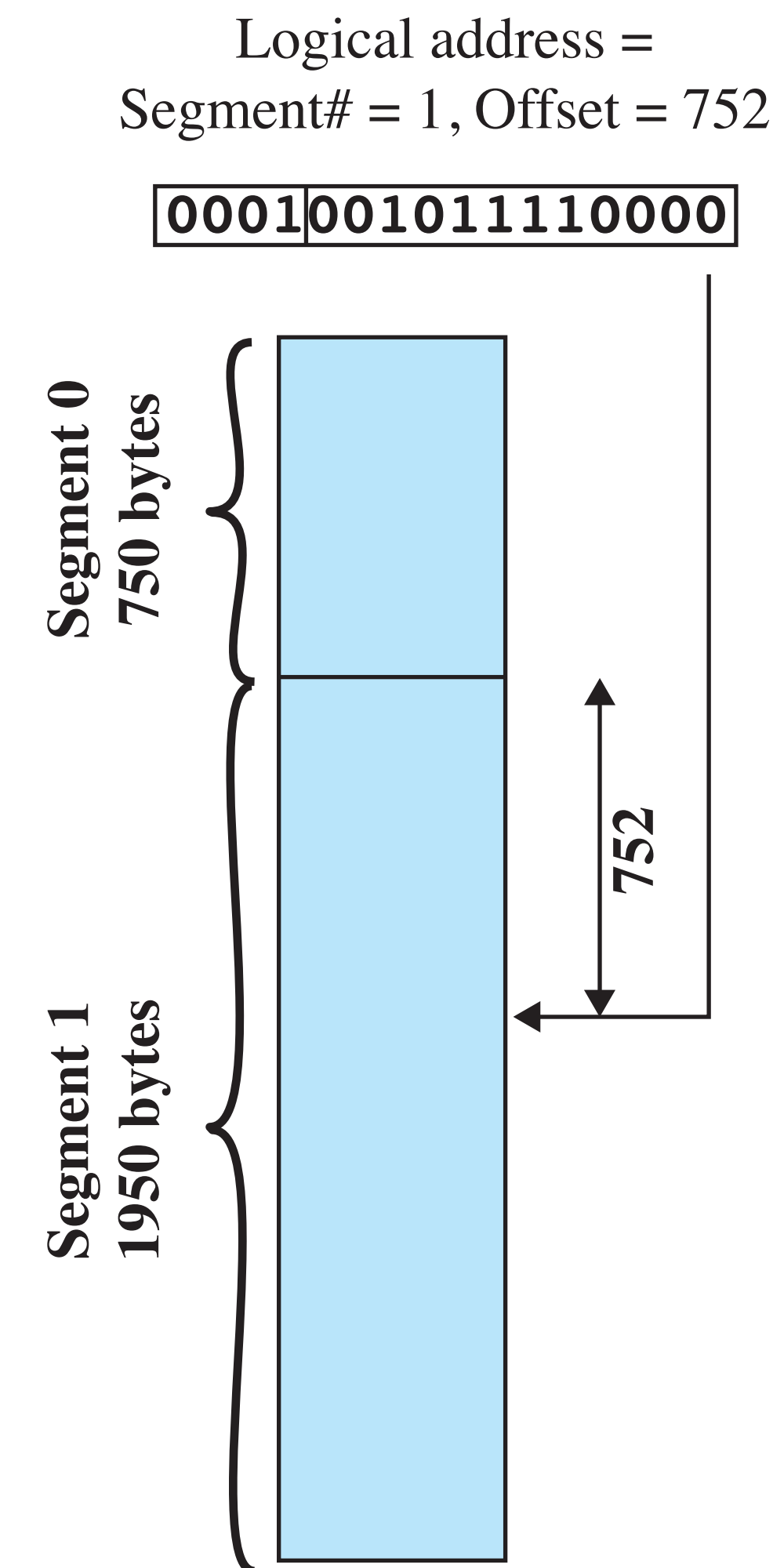
- 16-bit addresses
 - $n = 4$ bits
 - $m = 12$ bits

Q: What is the maximum size of a segment?

→ $4K = 2^{12}$

Q: How many segments are possible?

→ 4 bits used for segment # → 16 pages



Segmentation — Example of Logical-to-Physical Address Translation

