The Buddy System

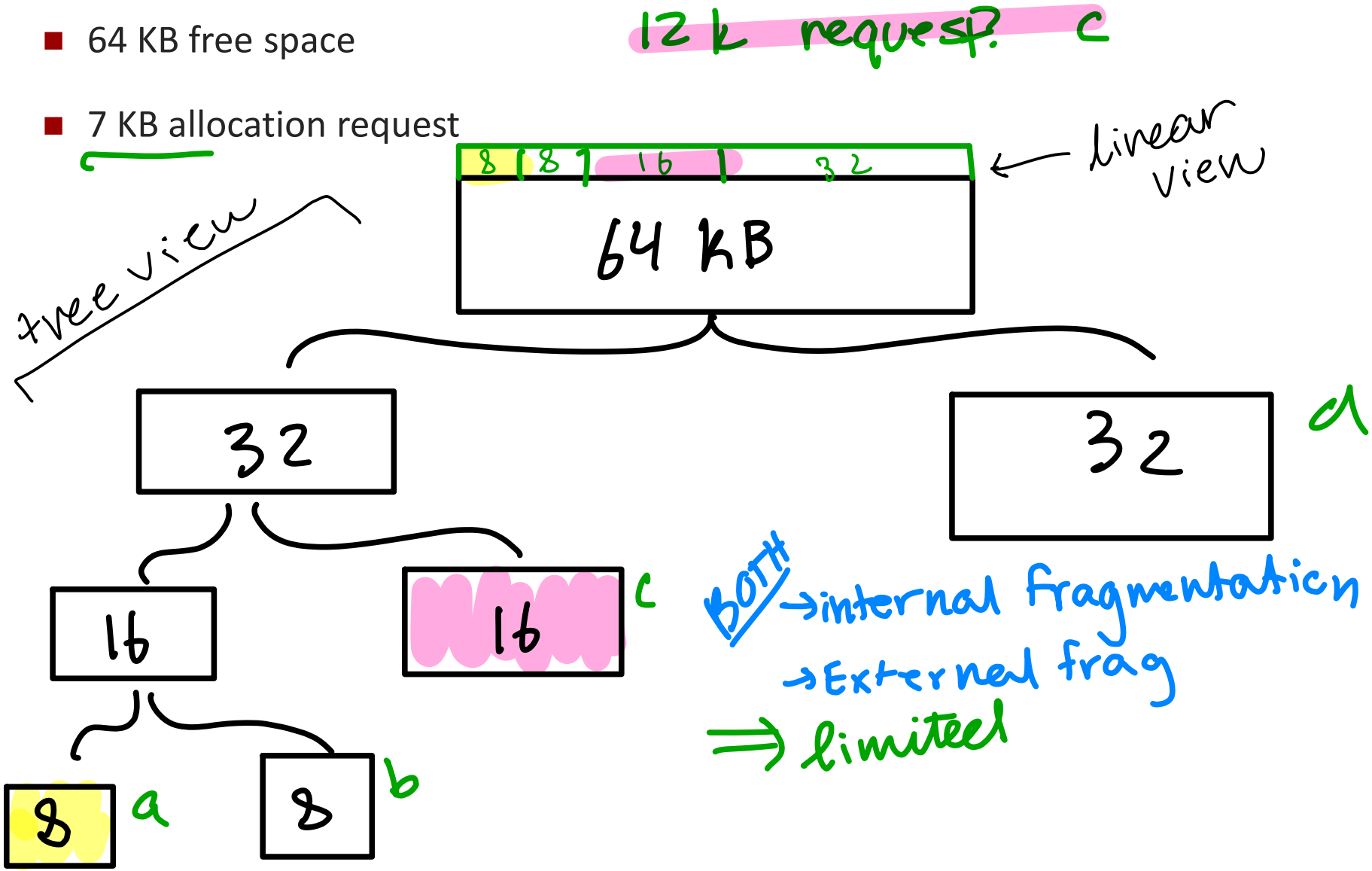
A *hybrid* of fixed and dynamic partitioning

1. start: all of physical memory is one large partition ↓ |size phys memory| = 2^N
2. allocation request: recursively split free mem in half until a further split is too small

3. de allocation request:
   - mark partition as free
   - when a parent has only freed children, compact/merge the children
     ↳ fast but limited

Buddy System Example

- 64 KB free space
- 7 KB allocation request

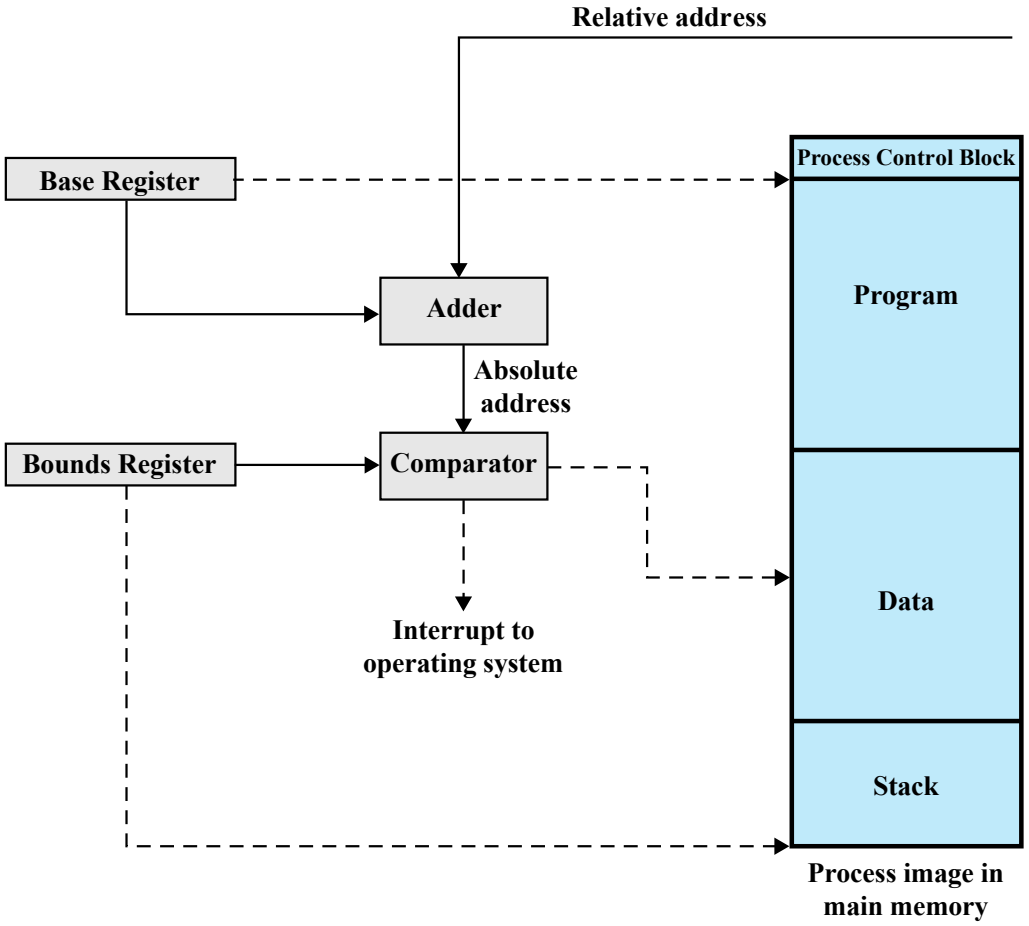


</Partitioning>

Different types of partitioning:

1. Fixed
2. Dynamic
3. Buddy System

address translation

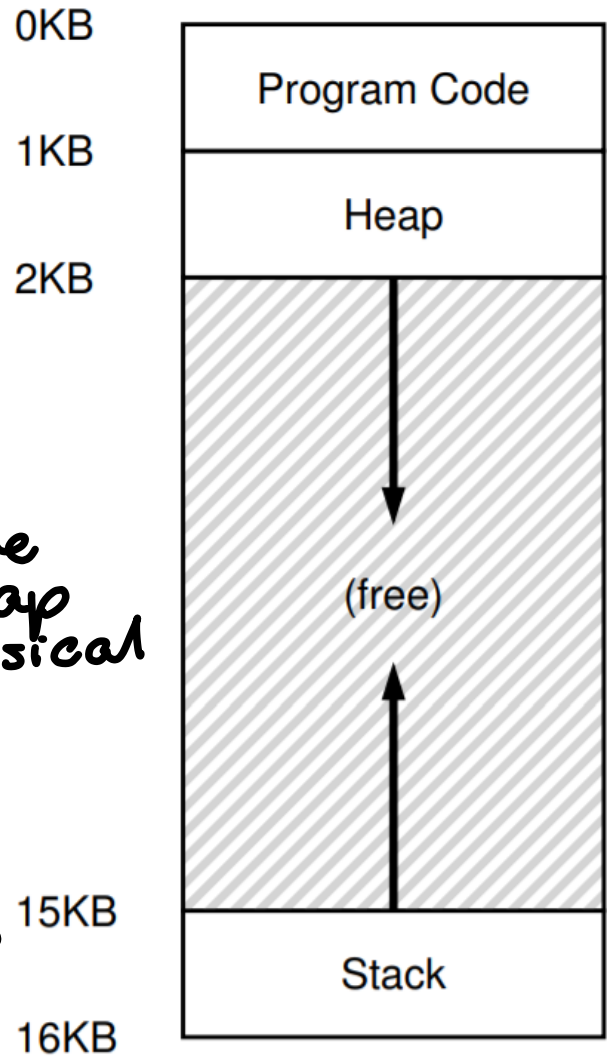


Part 2: Segmentation

Motivation

Partitioning:

- Transparent? ✓
- Process Isolation? ✓
- Sharing? ✗
- Efficiency?
  - speed: ✓
  - space: ✗
    1) fragmentation
    2) the gap between the stack and the heap is realized on physical mem



Crux of the problem:

how do we support a large but sparse address space efficiently?

Segmentation

Segment:

def: a contiguous portion of the processes address space

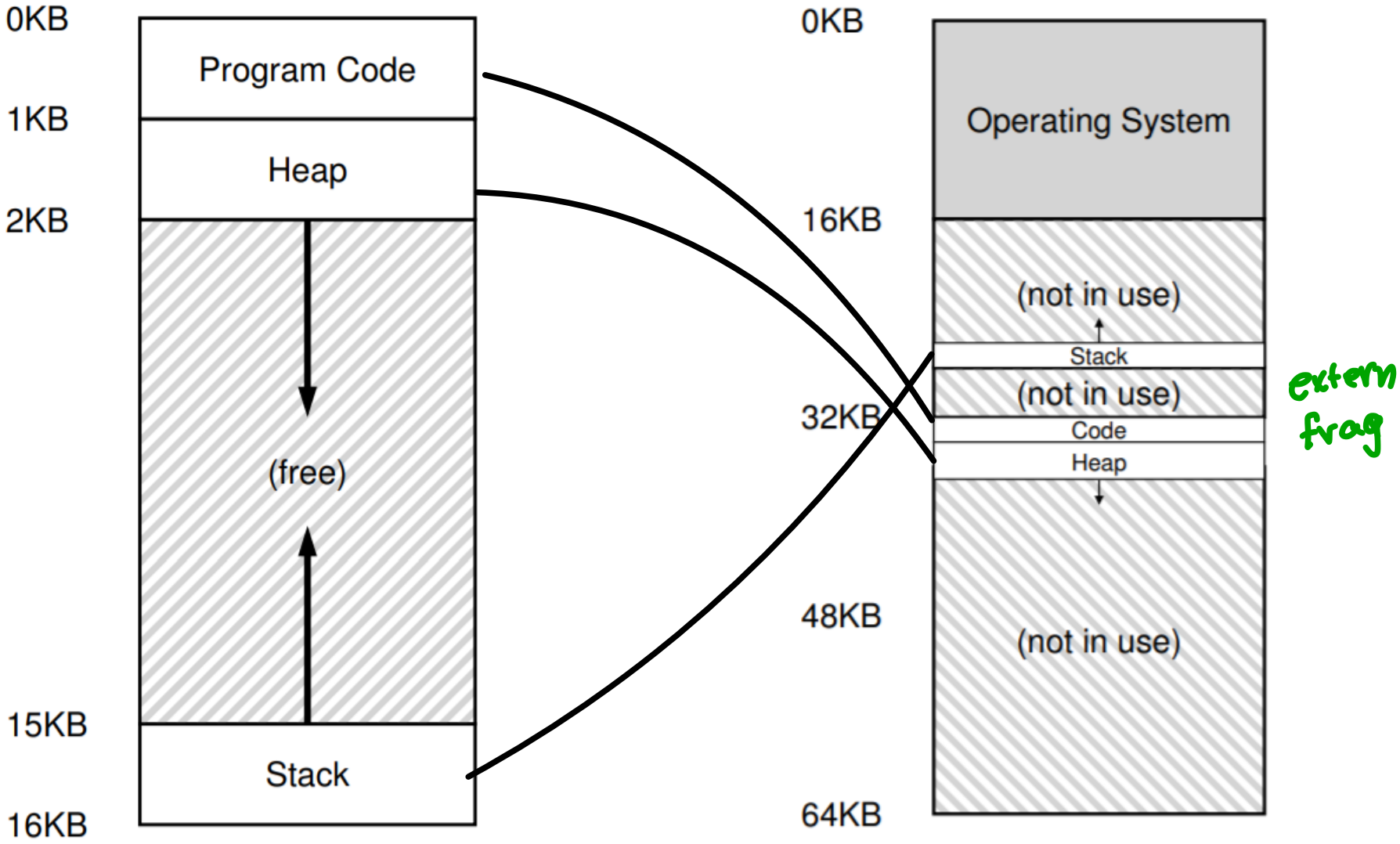
→ can be of diff lengths (up to a max)

→ ea segment has a diff base/bounds pair

→ 1: many relationship b/w procs & segs (dynamic partitioning 1:1)

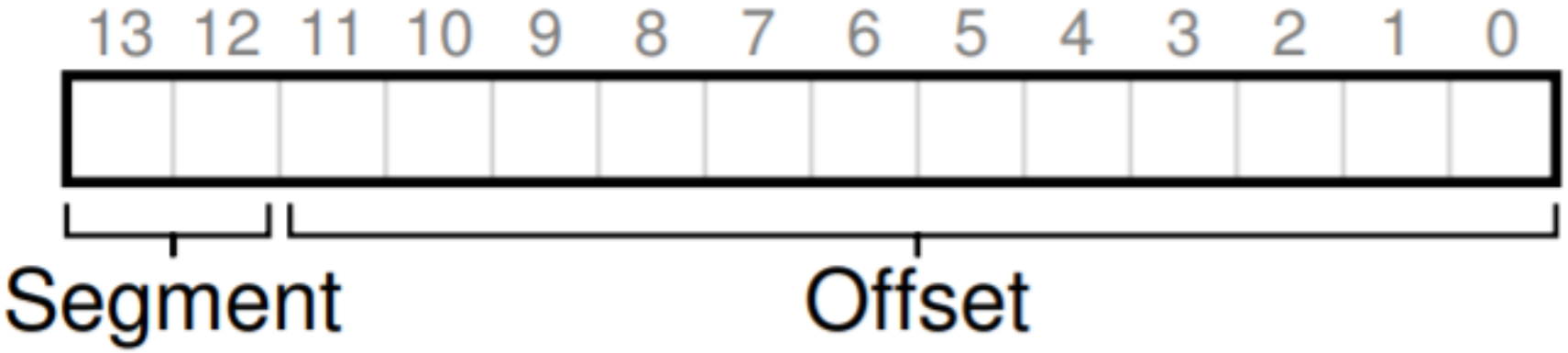
at least 3: code, stack, heap

High-Level Example

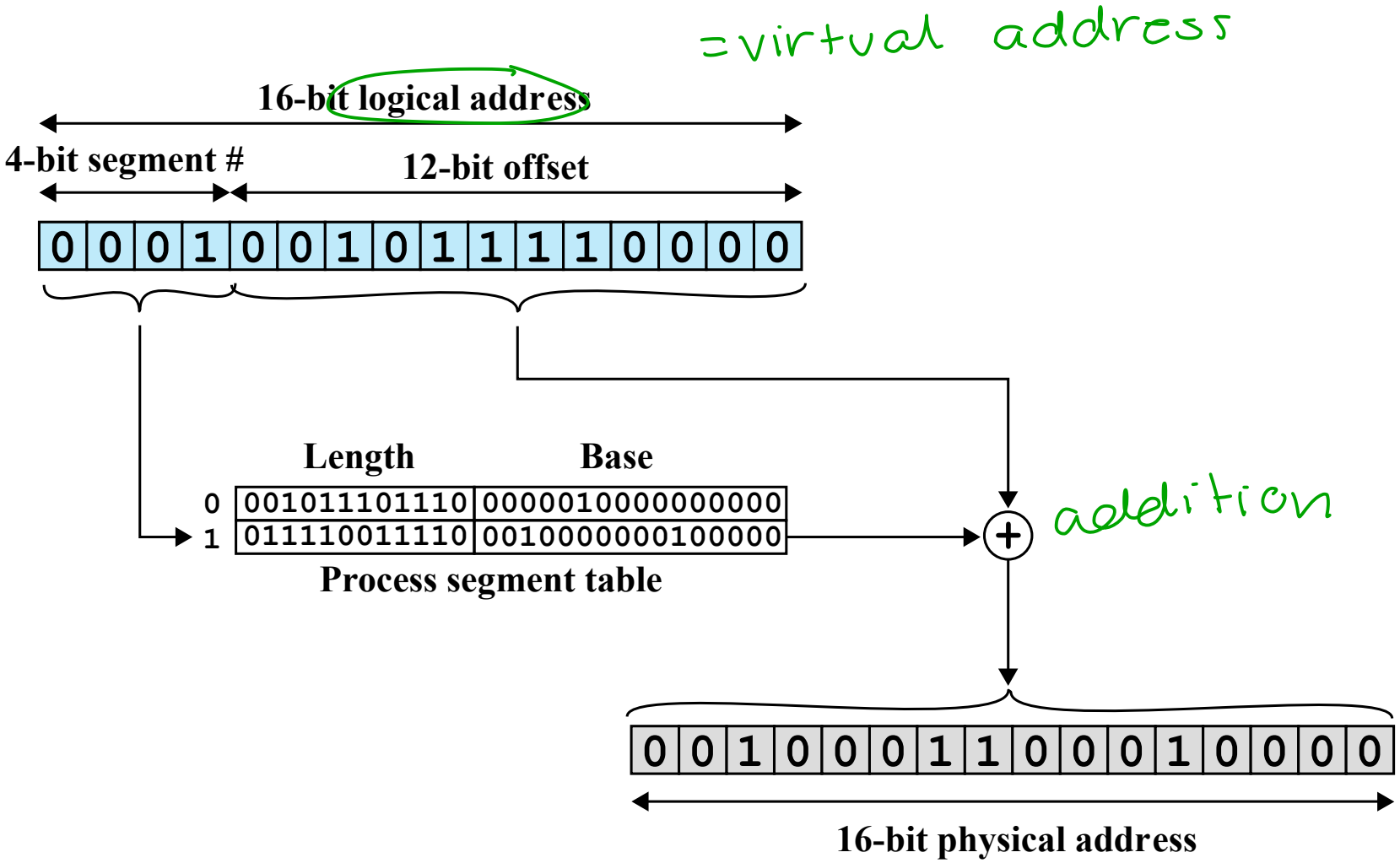


Address Translation

Which segment are we referring to?



Address Translation Figure



Segmentation Fault

What if you try to access an illegal address?

- E.g., beyond the end of the heap

1. Hardware detects out of bounds address
2. ... throw an interrupt...
3. OS kills the procs w/ error "seg fault"

Adding Support for Sharing

Protection Bits: control bits to indicate wether a process can read/write/exec a segment

Segment	Base	Size (Max 4K)	Protection
code	32k	2k	read/exec
stack	28k	2k	read/write
heap	34k	3k	read/write

seg table

Analysis

Segmentation:

+ no internal fragmentation
+ improved space efficiency
+ support sharing

- external fragmentation
- not flexible enough

One Last Term…

Simple Segmentation: all segs must be in RAM for the proc to run

Virtual Memory Segmentation:

NOT all segs must be in RAM for the proc to run

Part 3: Paging

Motivation

Dividing memory into *different*-sized pieces

=> External fragmentation

Fixed sized pieces => internal frag

limited by the size of partitions

Terms

Page: a contiguous chunk of a proc's address space

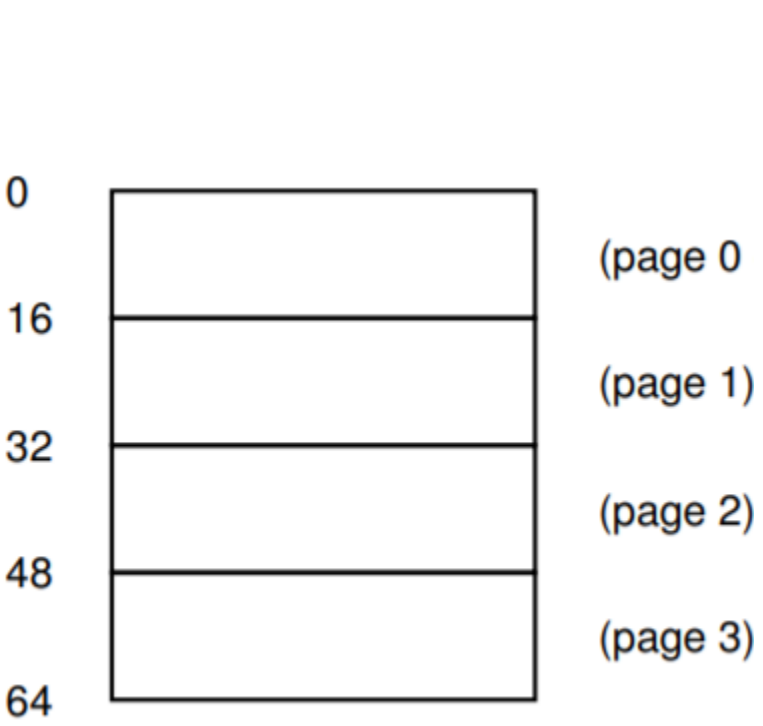
- → fixed size
- → typically very small
- → many pages per process

seg is typically
- dyn sized
- very large

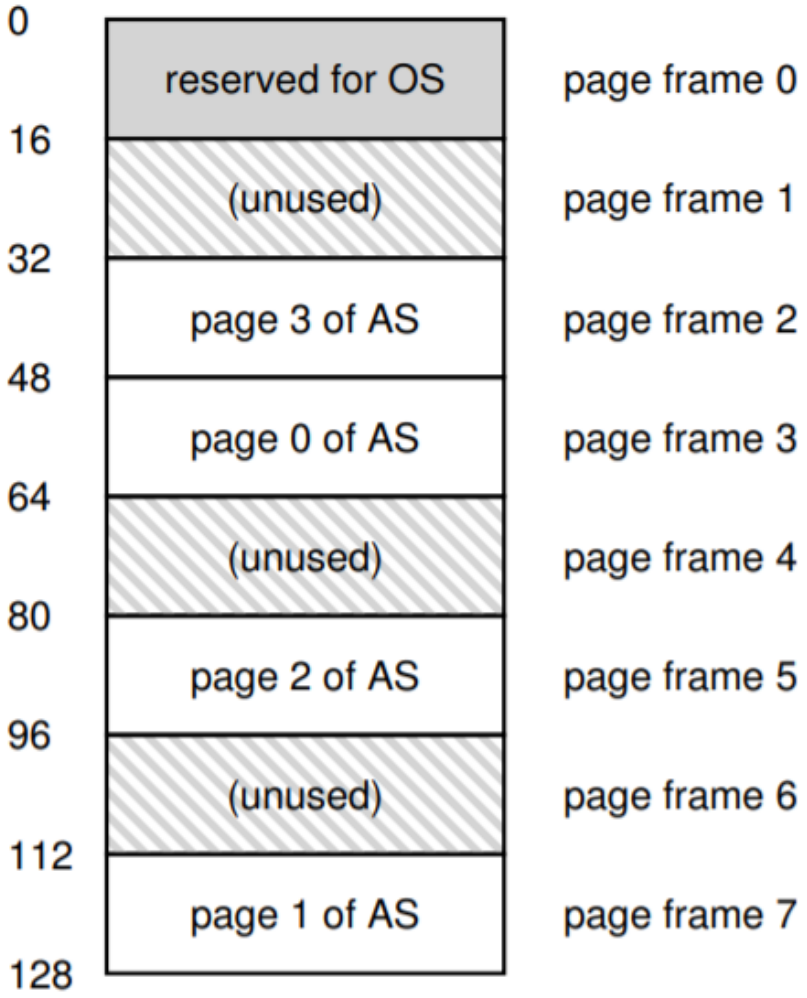
Page Frames: the slots in phys memory

- → only 1 page in a frame

Simple Example



Virtual Address Space

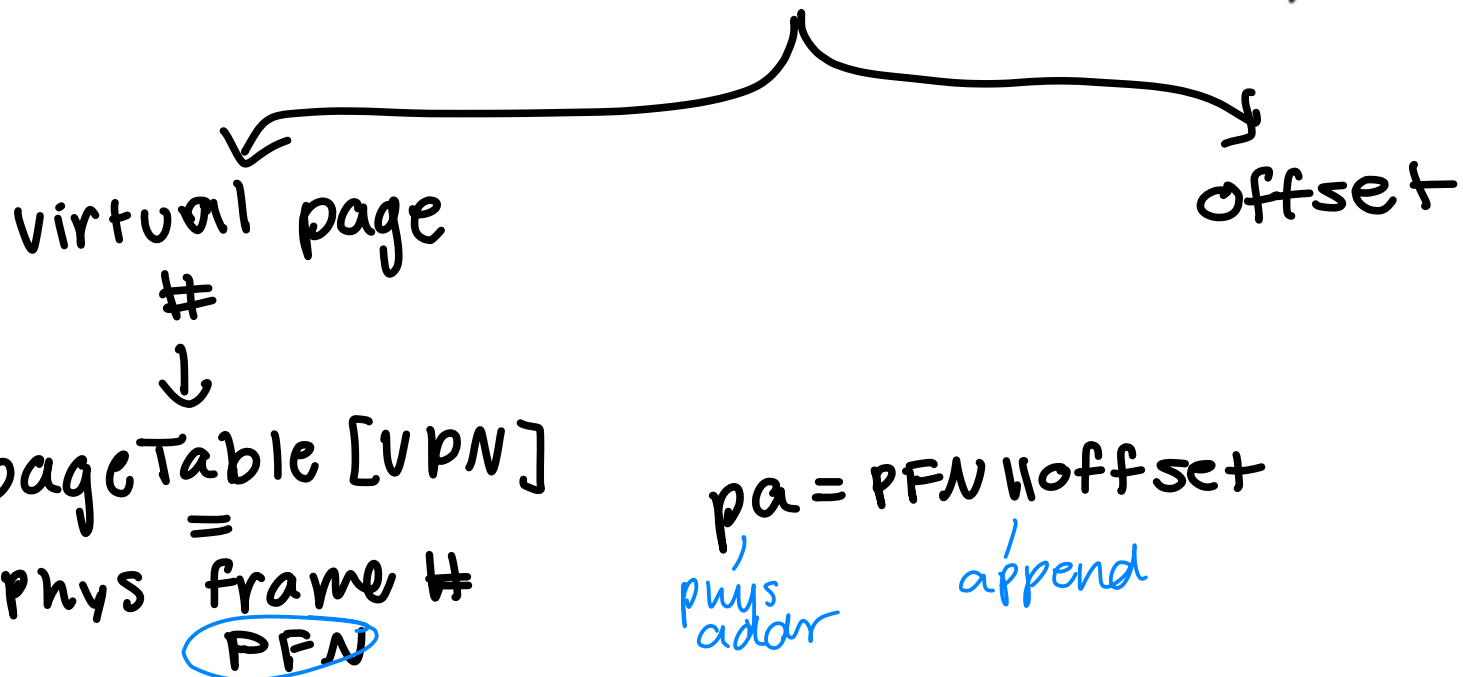


Physical Memory

Address Translation

Page Table: a per-proc structure which records the frame a page is located in along w/ some control bits

```
movl <virtual address>, %eax
```



Translation Example

- virt addr space sz = 64B = 2^6
- page sz: 16B = 2^4

virt addr

|VA| = 6 bits

|offset| = 4 bits

|VPN| = 6 − 4 = 2 b

```
movl 21, %eax
```

010101

VPN (2) | offset

page table [01] = 111

appended

111 0101 => 111 0101

Example in Fancy Graphics

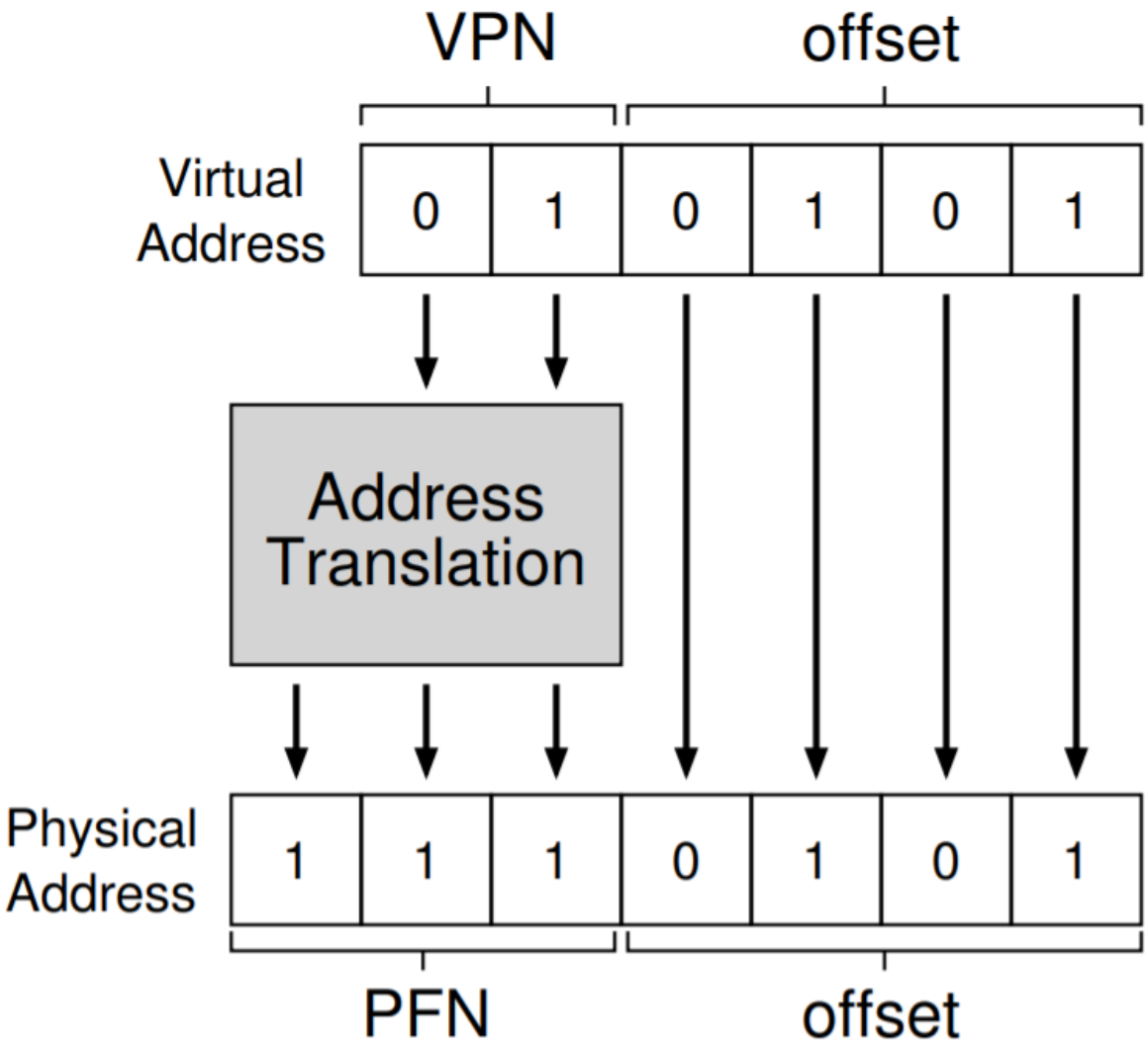


Figure 18.3: **The Address Translation Process**

Hw 7 p 3

Learning Group Time

HW 7, #1 and #2