IOWA STATE UNIVERSITY
Department of Electrical and Computer Engineering

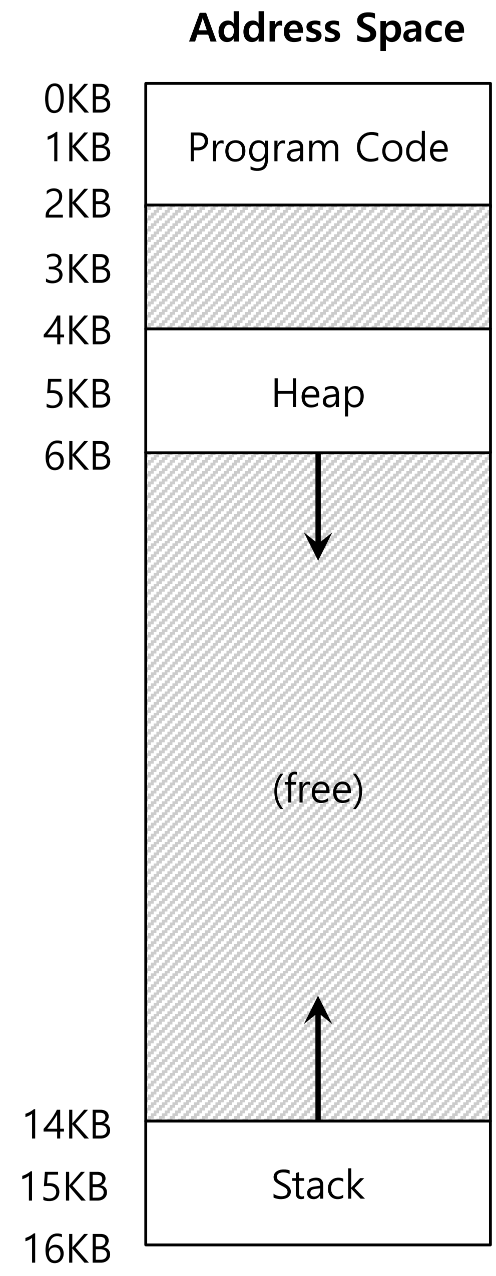
Lecture 19:
Free-Space Management

Agenda

- Recap
- Free-Space Management (cont')
- Paging Concepts

Recap

- Segmentation
  - Break the full address space into a few segments
    - Segment: a contiguous portion of the address space of a particular length
    - Logically-different segment:
      - code, stack, heap, data
    - Each segment can be placed in different part of physical memory
      - Base and limit exist for each segment

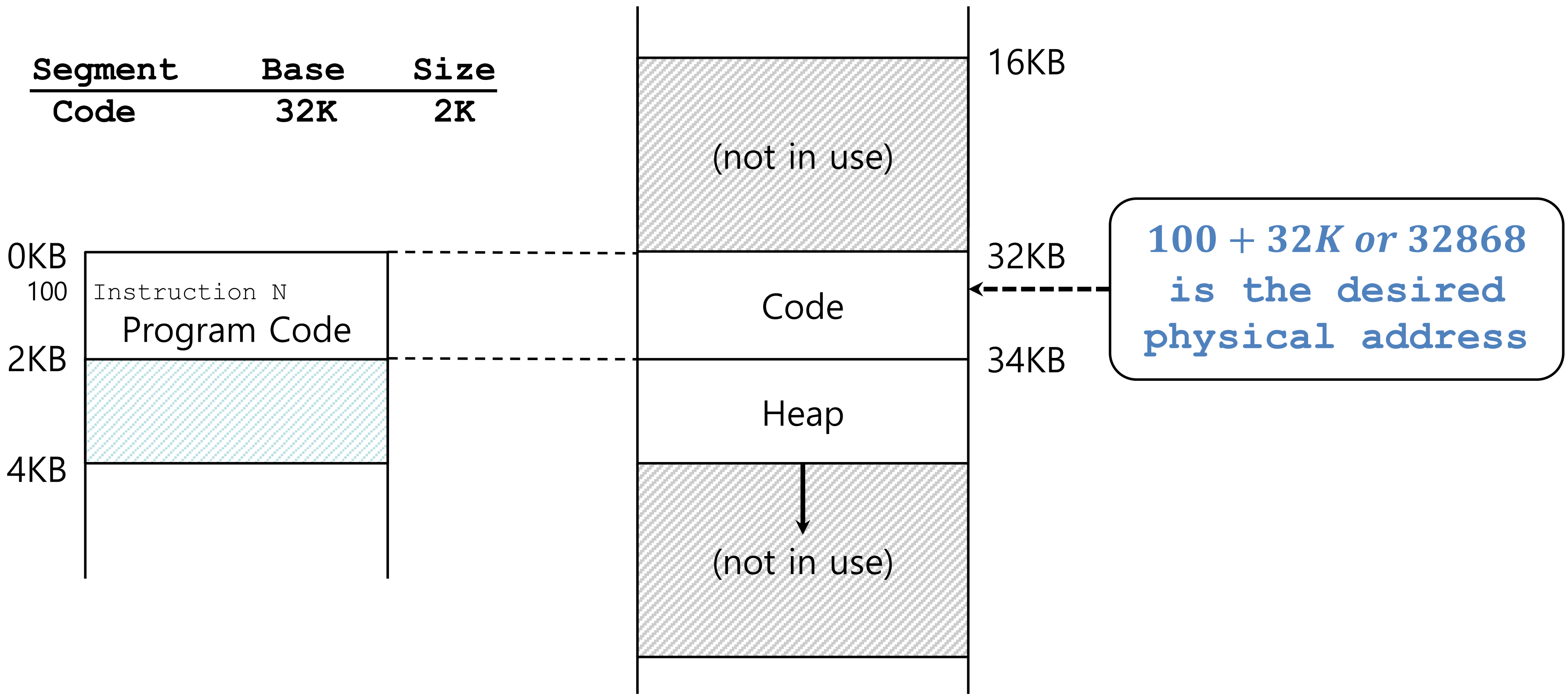


Recap

- Address Translation of Segmentation

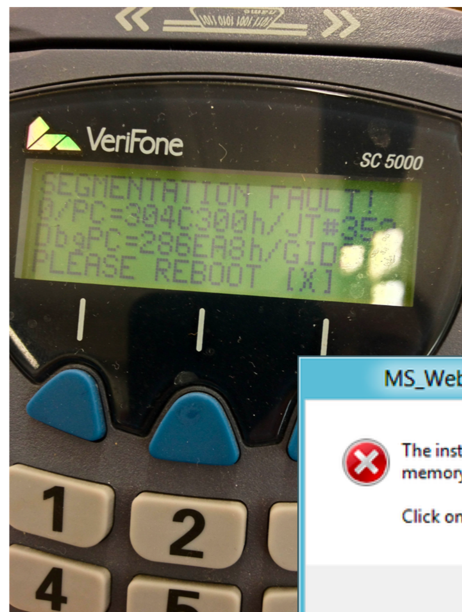
$$physical\ address = offset + base$$

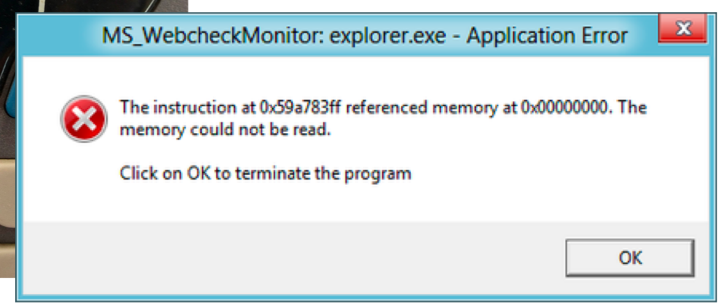
Segment	Base	Size
Code	32K	2K



Recap

- Segmentation Fault
  - accessing illegal memory address
    - Hardware detects the illegal access and raises an exception
    - CPU executes a fault handler (part of the OS)
      - E.g., kill the process and throws a *segmentation fault* message to user



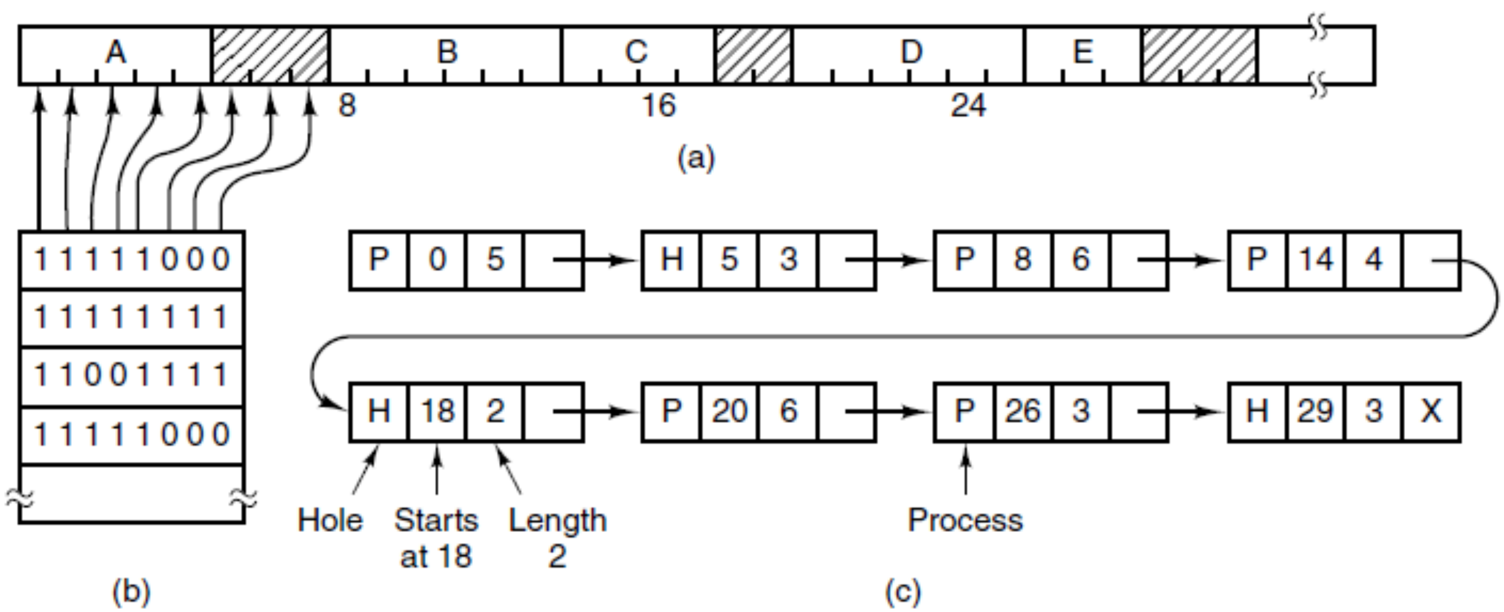


Recap

- Free-Space Management
  - How to find a free chunk of memory that can satisfy the user request?
    - Need to keep track of the free/used space
    - Often needs to split a free chunk into two
      - one sub-chunk is allocated to the requesting process
      - the rest of the chunk remains free
    - Two Basic Approaches
      - Bitmap
      - Linked list

Recap

- Free-Space Management
  - Bitmap
  - Linked list (version 1)

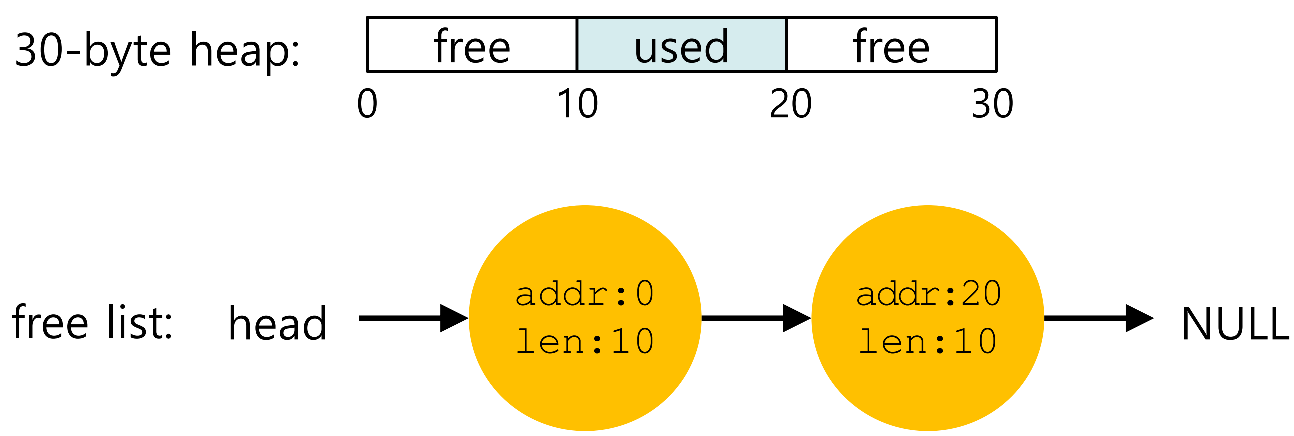


Agenda

- ~~Recap~~
- Free-Space Management (cont')
- Paging Concepts

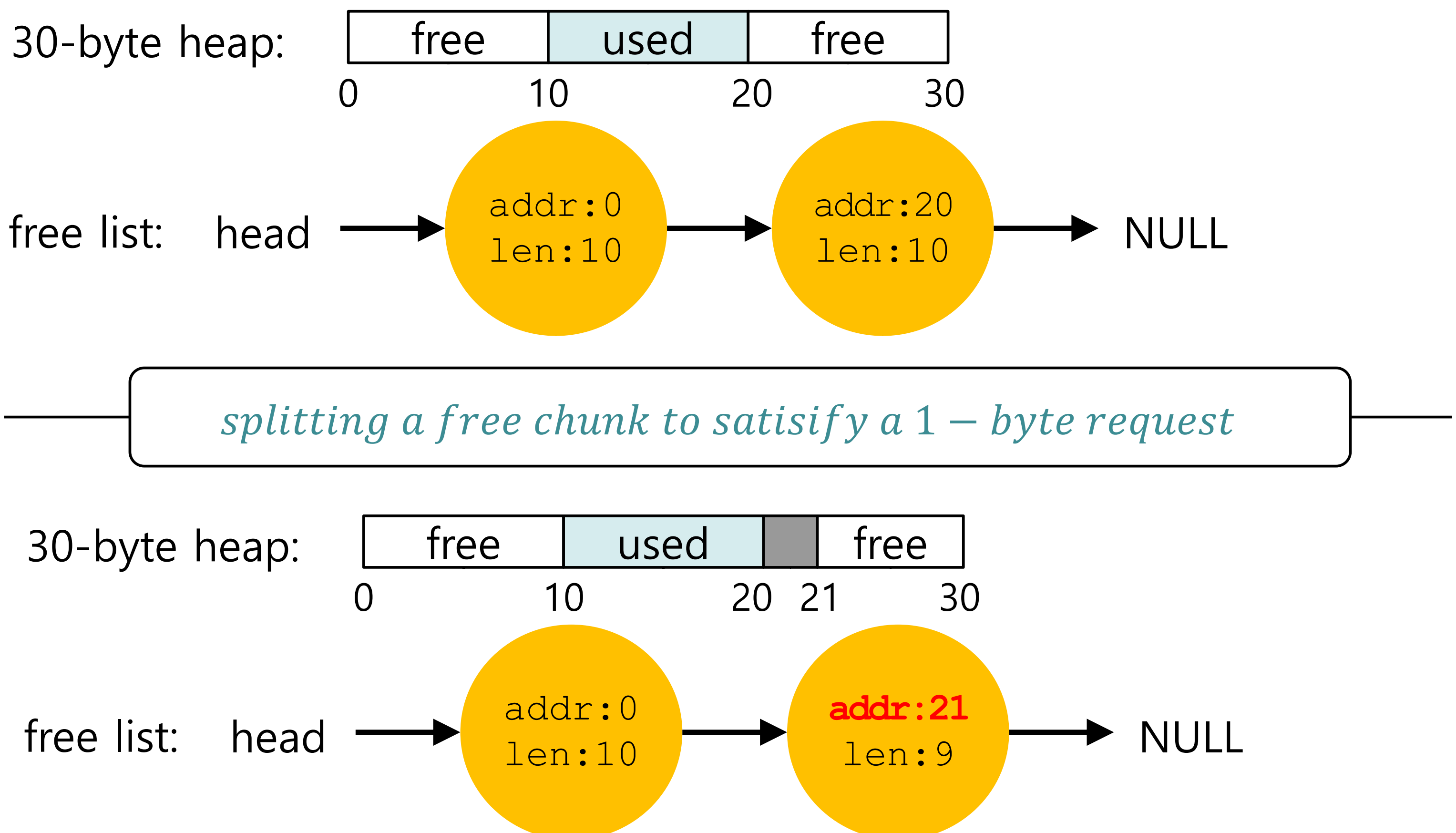
Free-Space Management

- Linked list (version 2)
  - link free chunks



Free-Space Management

- Linked list (version 2)
  - link free chunks



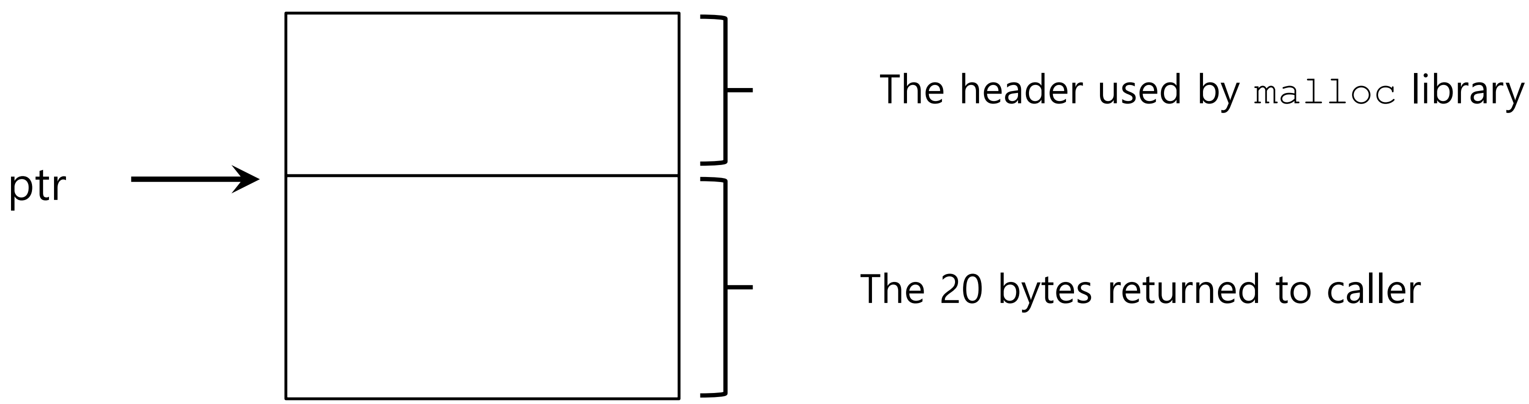
Free-Space Management

- Linked list (version 2)
  - link free chunks
  - tracking the size of allocated regions
    - The interface to `free(void *ptr)` does not take a size parameter
    - How does the library know the size of memory region that will be back into free list?

Free-Space Management

- Linked list (version 2)
  - link free chunks
  - tracking the size of allocated regions
    - Most allocators store extra information in a header block

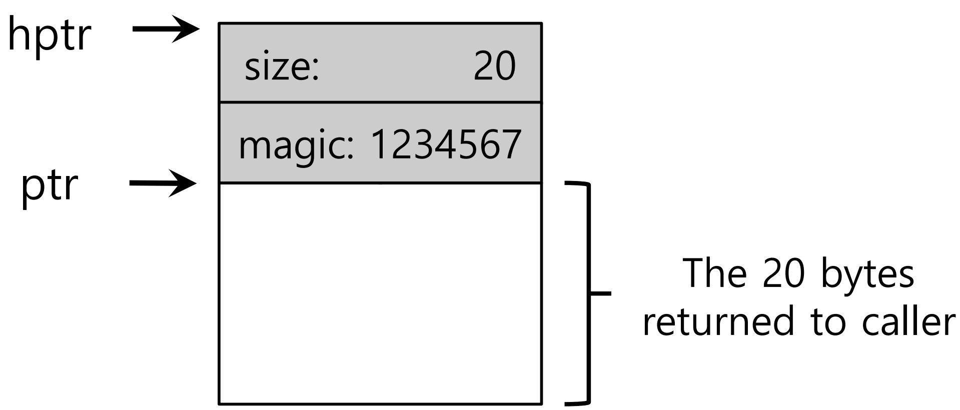
```
ptr = malloc(20);
```



An Allocated Region Plus Header

Free-Space Management

- Linked list (version 2)
  - link free chunks
  - tracking the size of allocated regions
    - The header minimally contains the size of the allocated memory region
    - The header may also contain other info
      - e.g., a magic number for integrity checking



Specific Contents Of The Header

```
typedef struct __header_t {
        int size;
        int magic;
} header_t;
```

A Simple Header

Free-Space Management

- Linked list (version 2)
  - link free chunks
  - tracking the size of allocated regions
    - The size of the allocated region is the size of the header plus the size of the space allocated to the user.
      - If a user request N bytes, the library searches for a free chunk of size N **plus the size of the header**
    - Simple pointer arithmetic to find the header pointer.

```
void free(void *ptr) {
        header_t *hptr = (void *)ptr - sizeof(header_t);
        ...
}
```

Free-Space Management

- Linked list
  - Memory allocation
    - Best Fit:
      - Find free chunks that are big or bigger than the request
      - Return the smallest one in the group of candidates
    - Worst Fit:
      - Find the largest free chunks and allocate the amount of the request (so that the remaining chunk may still be usable)
      - Keep the remaining chunk on the free list
    - First Fit:
      - Find the first chunk that is big enough for the request
      - Return the requested amount and keep the remaining chunk on the free list

Free-Space Management

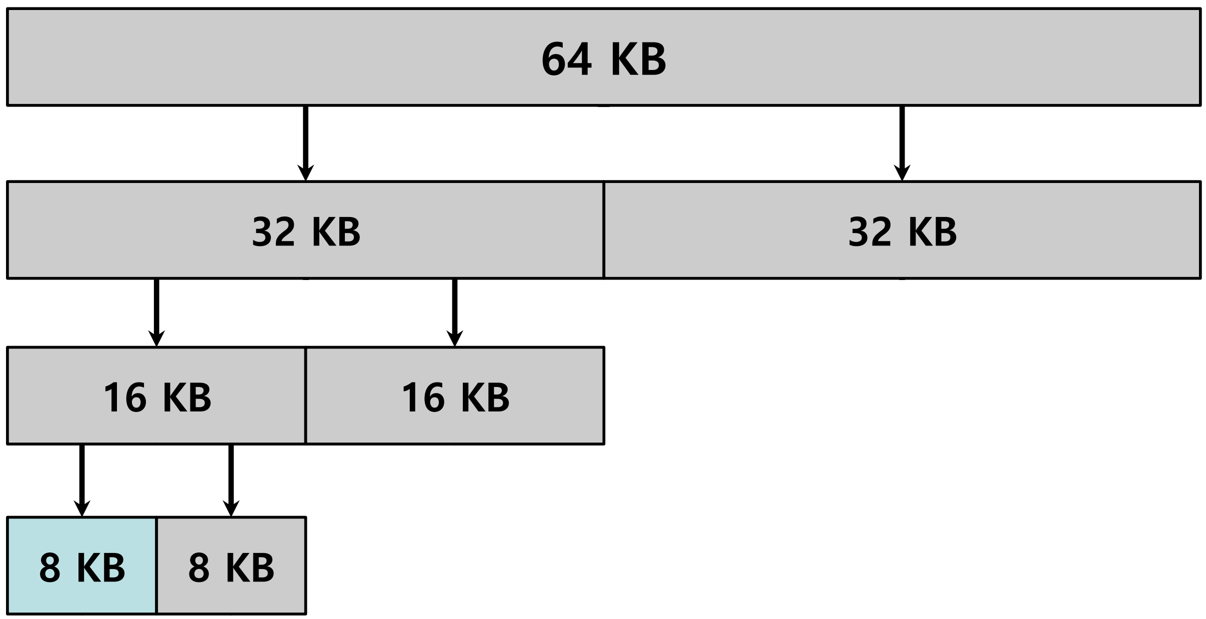
- Other approaches
  - Segregated List
    - Keeping free chunks in different size in a separate list for the size of popular request.
    - New Complication:
      - How much memory should dedicate to the pool of memory that serves specialized requests of a given size?

Free-Space Management

- Other approaches
  - Segregated List
    - Keeping free chunks in different size in a separate list for the size of popular request.
    - New Complication:
      - How much memory should dedicate to the pool of memory that serves specialized requests of a given size?
    - **Slab allocator** handles this issue
      - Allocate a number of object caches
        - The objects are likely to be requested frequently.
        - e.g., locks, etc.
      - Request some memory from a more general memory allocator when a given cache is running low on free space.

Free-Space Management

- Other approaches
  - Buddy Allocation
    - The allocator divides free space by two until a block that is big enough to accommodate the request is found.



64KB free space for a 7KB request

Free-Space Management

- Other approaches
  - Buddy Allocation
    - The allocator divides free space by two until a block that is big enough to accommodate the request is found
    - Buddy allocation can suffer from **internal fragmentation**.
    - Buddy system makes **coalescing** simple.
      - Coalescing two blocks in to the next level of block.

Agenda

- ~~Recap~~
- ~~Free-Space Management (cont’)~~
- Paging Concepts

Paging

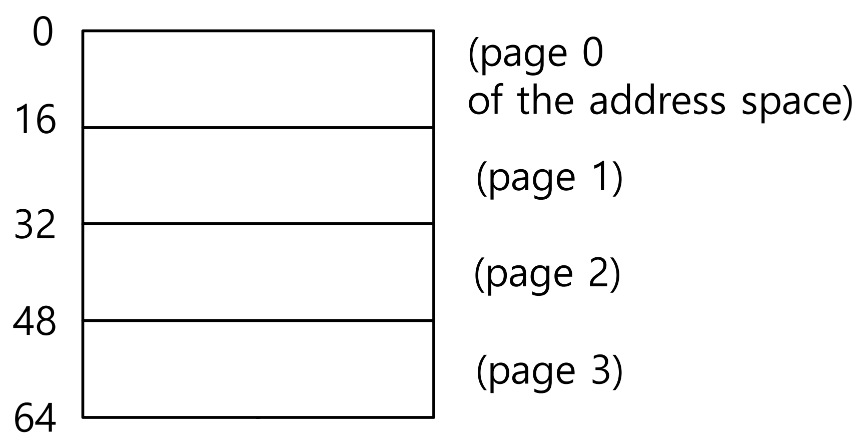
- Paging splits up address space into fixed-size units called ***pages***
  - typically 4KB each
  - Segmentation: variable size of logical segments (code, stack, heap, etc.)
- With paging, physical memory is also split into fixed-size units called ***page frames***
  - typically 4KB each

Advantages of Paging

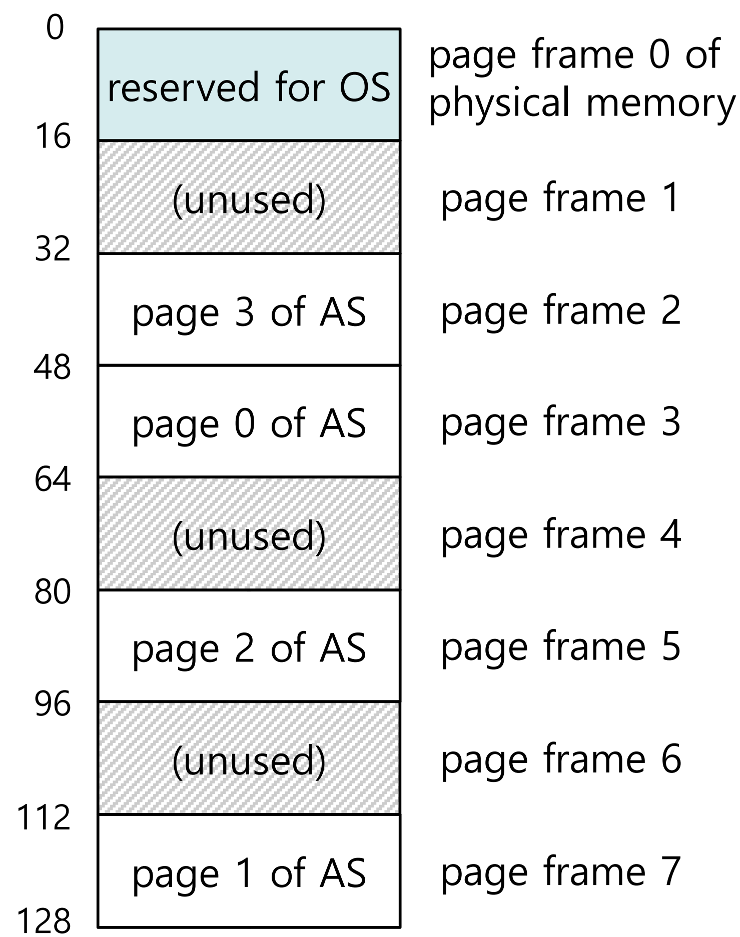
- Flexibility
  - Supporting the abstraction of address space effectively
    - Don't need assumption on how heap and stack grow or are used
- Simplicity
  - ease of free-space management
    - the page in address space and the page frame are the same size
    - easy to allocate and keep a free list

One Simple Example

- 64-byte address space
  - 16-byte pages
- 128-byte physical memory
  - 16-byte page frames



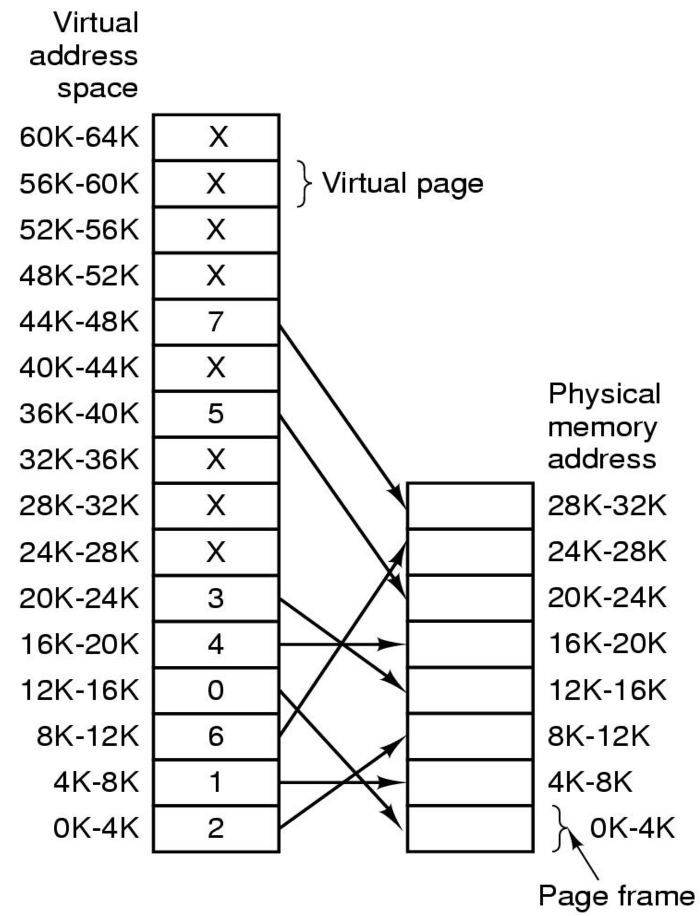
A Simple 64-byte Address Space



64-Byte Address Space Placed In Physical Memory

Another Example

- 64KB Address Space
  - divided into 16 pages
  - 4KB each page
- 32KB Physical Memory
  - 8 page frames
  - 4KB each



Agenda

- ~~Recap~~
- ~~Free-Space Management (cont')~~
- ~~Paging Concepts~~

Questions?

*acknowledgement: slides include content from "Modern Operating Systems" by A. Tanenbaum, "Operating Systems Concepts" by A. Silberschatz etc., "Operating Systems: Three Easy Pieces" by R. Arpaci-Dusseau etc., and anonymous pictures from internet.