IOWA STATE UNIVERSITY
Department of Electrical and Computer Engineering

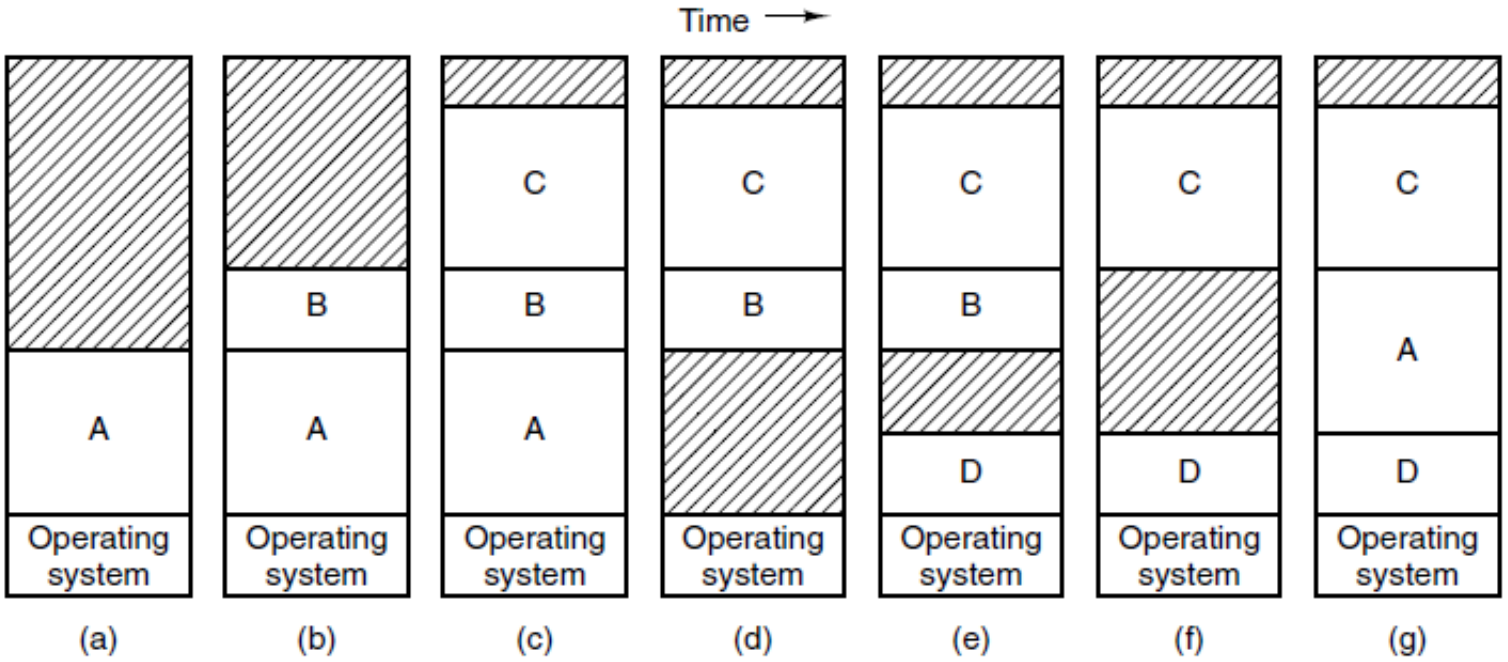
Lecture 18: Segmentation & Free-Space Management

Agenda

- Recap
- Segmentation
- Free-Space Management

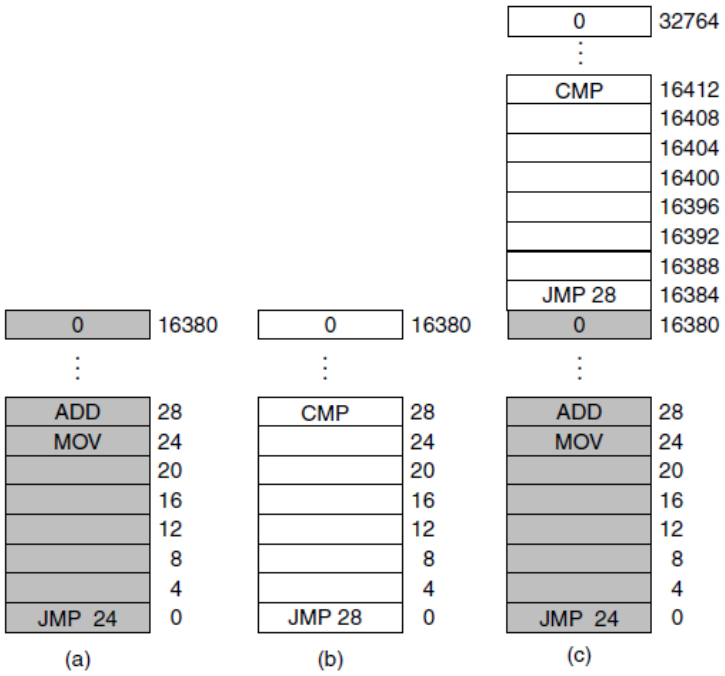
Recap

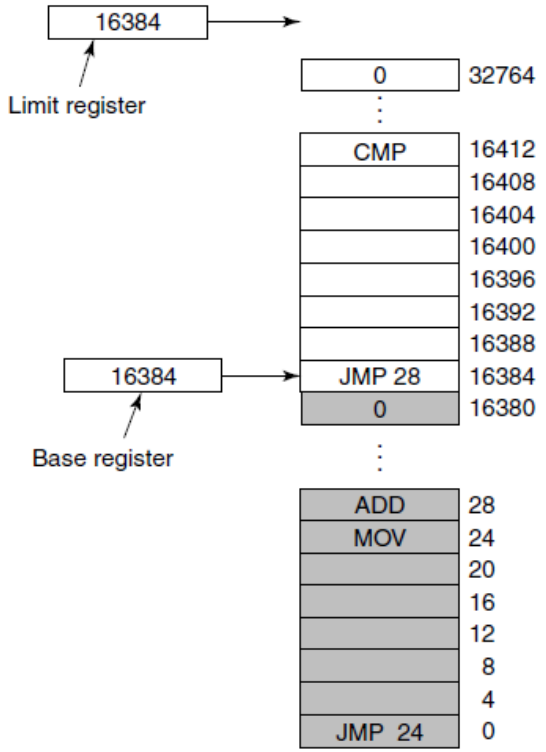
- Memory management in early systems
  - Treat the memory space of a process as a whole



Recap

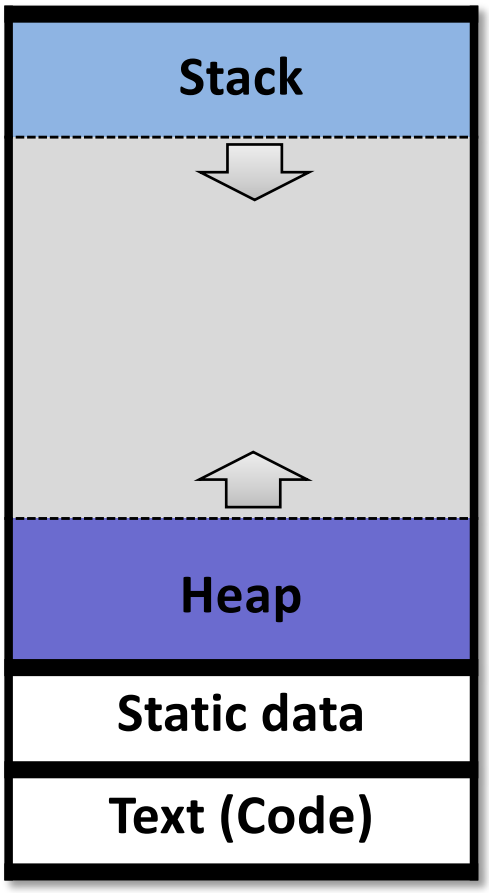
- Memory management in early systems
  - Relocation using a pair of *base* and *limit* registers
  - Basic protection: checking if address within the range





Recap

- Modern OS provides a virtual memory for each process
  - An illusion that each process uses the whole memory itself
  - Improve memory efficiency, isolation & protection
- Address space
  - An abstraction of physical memory for a process
  - the set of all virtual addresses visible to a program



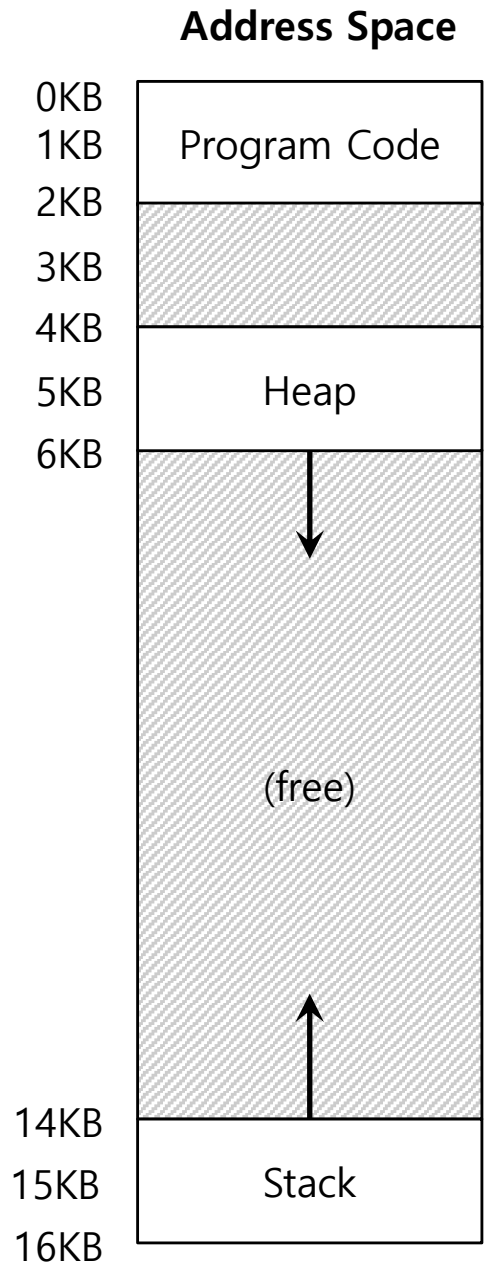
Address Space

Agenda

- ~~Recap~~
- Segmentation
- Free-Space Management

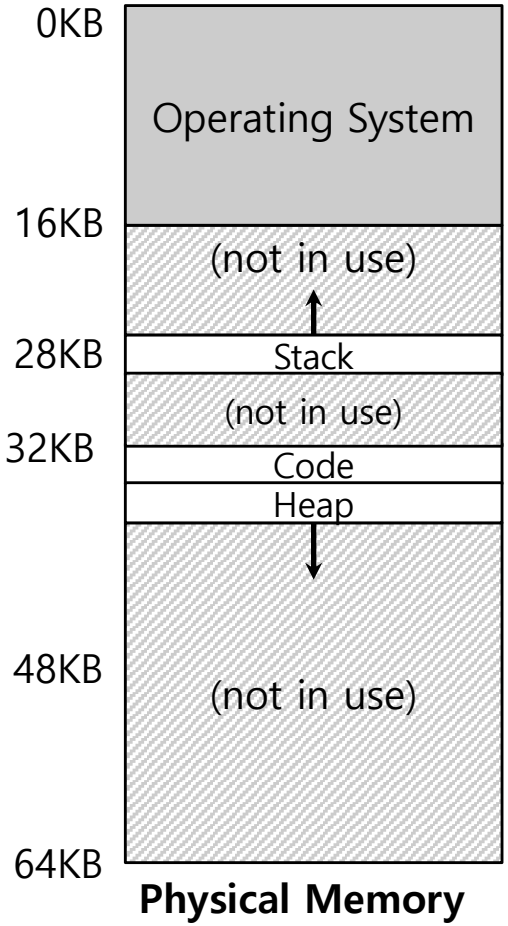
Segmentation

- 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



Segmentation

- Placing segments in physical memory

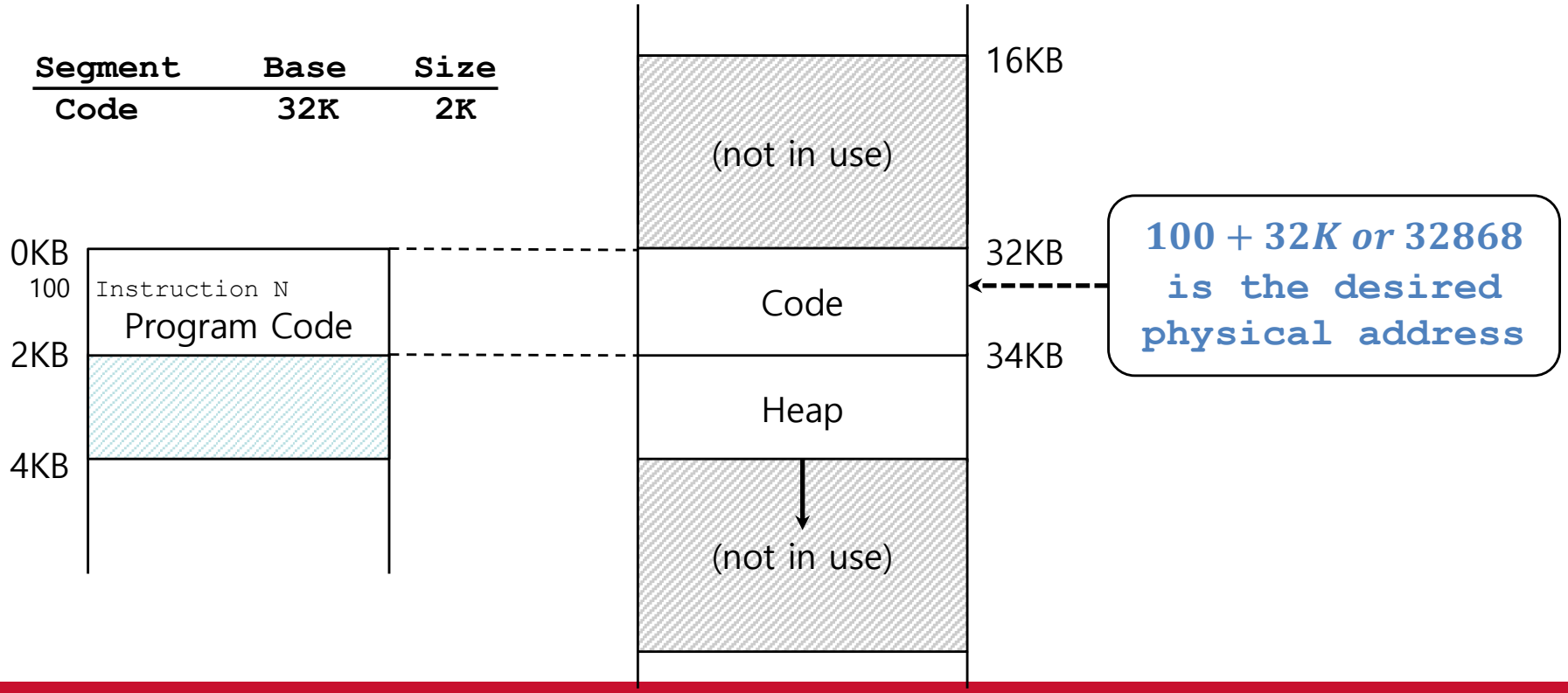


Segment	Base	Size
Code	32K	2K
Heap	34K	2K
Stack	28K	2K

Address Translation of Segmentation

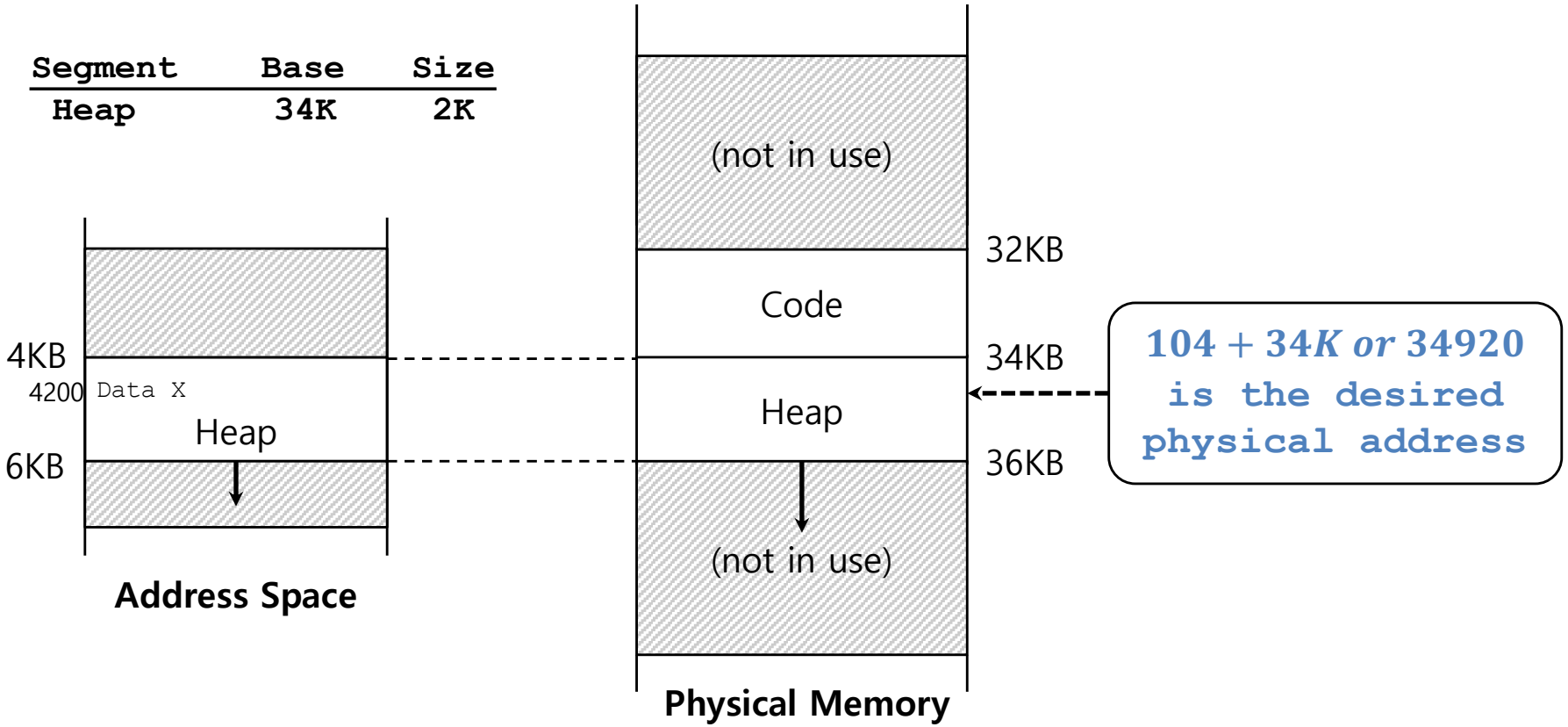
$$physical\ address = offset + base$$

- The code segment starts at virtual address 0 in address space
  - The `offset` of virtual address 100 in code region is `100`.



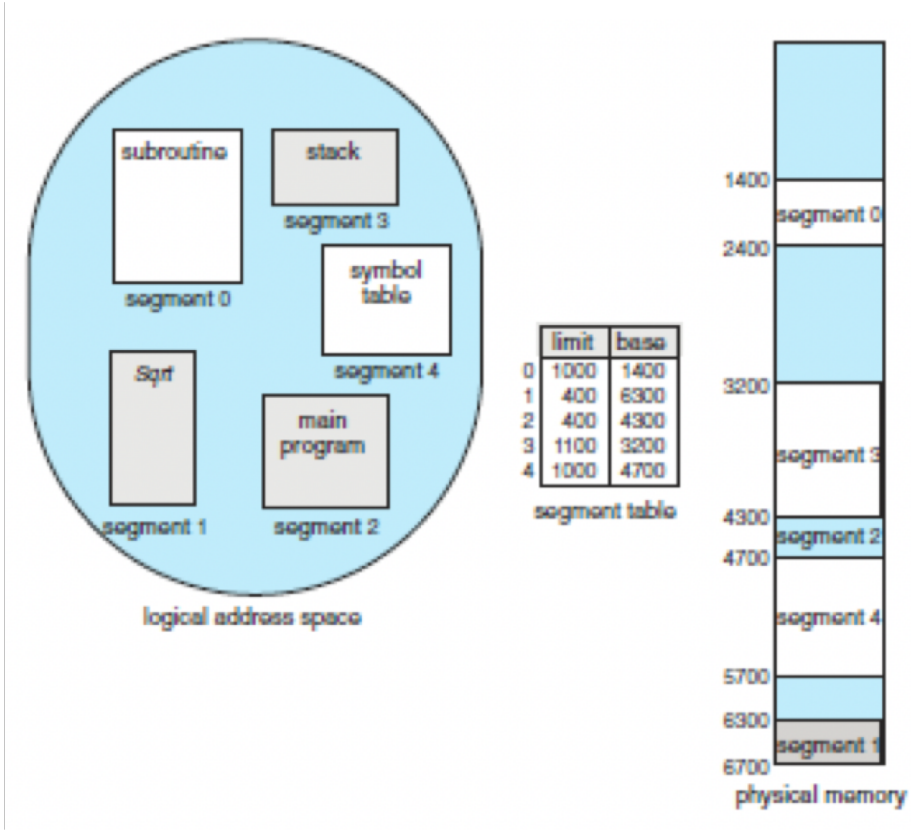
Address Translation of Segmentation

- ***Virtual address+base*** is not always the correct physical address.
  - The heap segment starts at virtual address 4K in address space.
    - The `offset` of virtual address 4200 is `104`.



Address Translation of Segmentation

- To support multiple segments, a segment table is needed



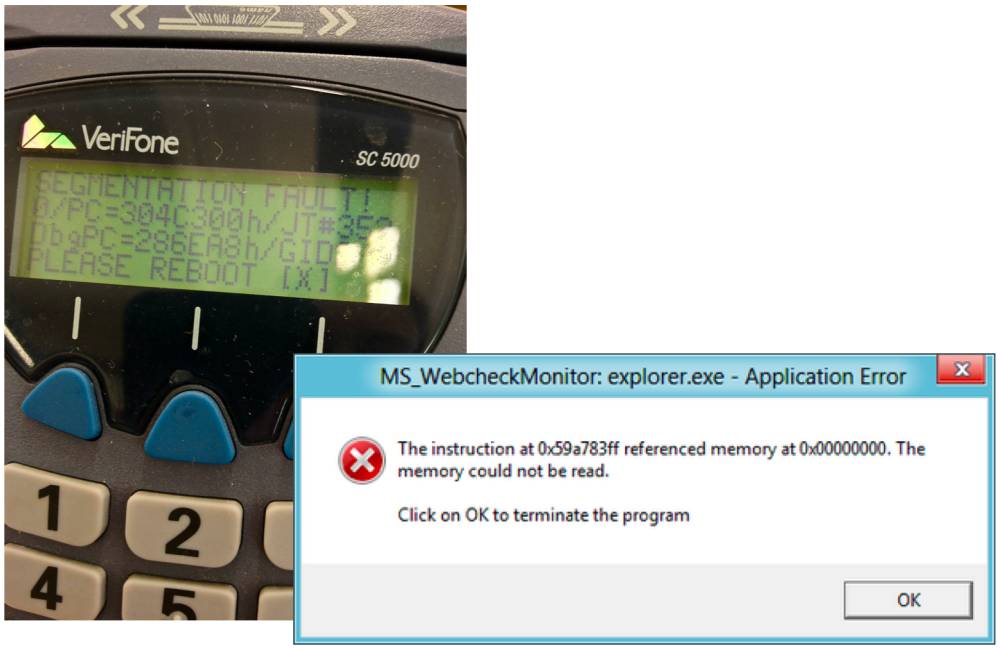
Segmentation Fault

- In case of illegal 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

```
diego@cryptos:/tmp$ cat segfault.c
int main(int argc, const char *argv[]){

        int *ptr;
        *(ptr + 1000000) = 10;

        return 0;
}
diego@cryptos:/tmp$ gcc segfault.c -o segfault
diego@cryptos:/tmp$
diego@cryptos:/tmp$ ./segfault
Segmentation fault (core dumped)
diego@cryptos:/tmp$
diego@cryptos:/tmp$ :D
```



Sharing & Protection

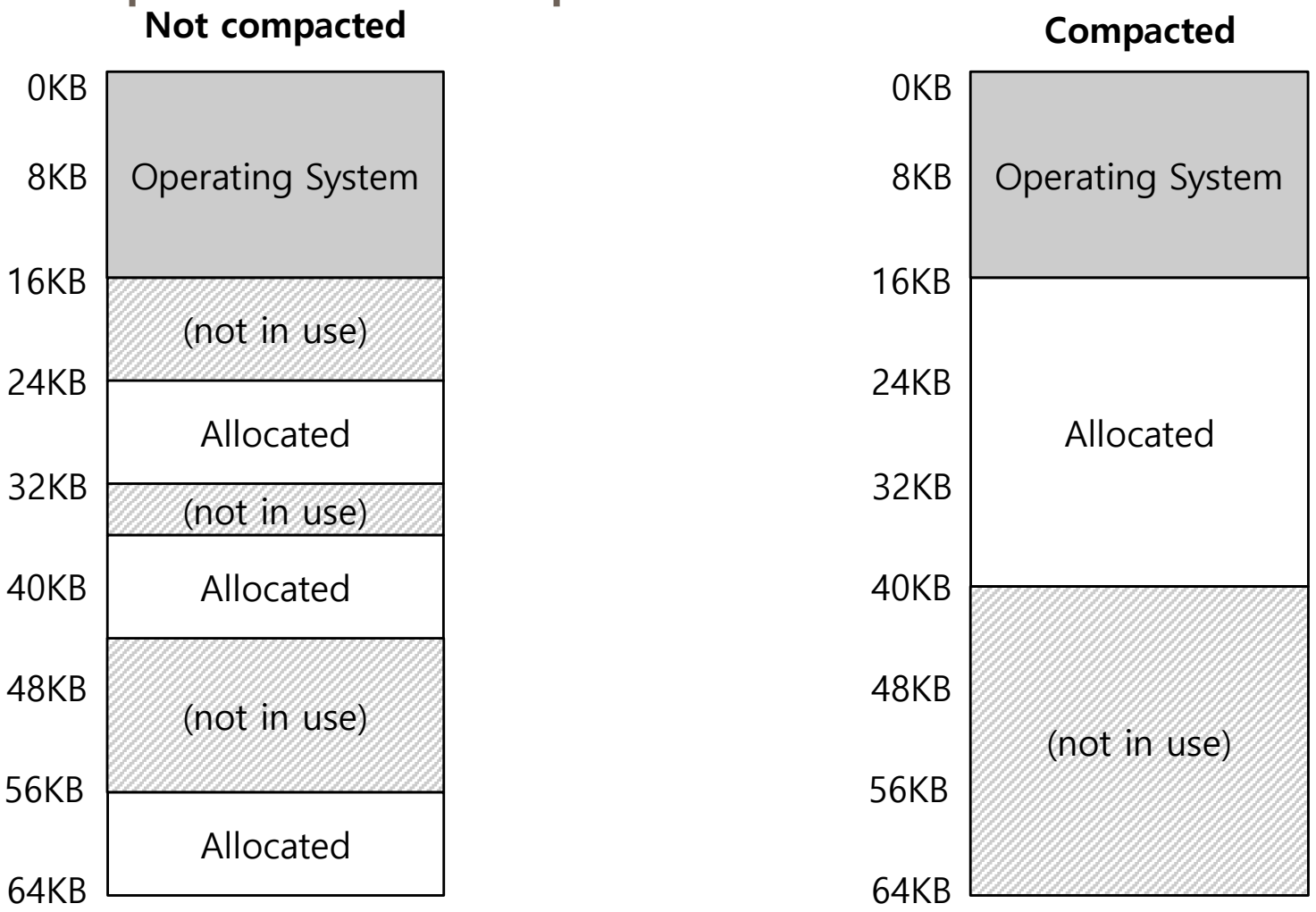
- Segment can be shared between address spaces
  - Code sharing is still in use in systems today
- Extra hardware support is needed for protection
  - A few more bits in the segment registers to indicate permissions of read, write and execute

Fragmentation & Compaction

- External Fragmentation
  - little holes of free space in physical memory that make difficulty to allocate new segments
  - E.g., there is 24KB free space, but not in one contiguous segment.
    - the OS cannot satisfy a 20KB request.
- Compaction
  - rearranging the exiting segments in physical memory.
  - Compaction is costly.
    - Stop running process.
    - Copy data to somewhere.
    - Change segment register value.

Fragmentation & Compaction

- Compaction example



Agenda

- ~~Recap~~
- ~~Segmentation~~
- Free-Space Management

Free-Space Management

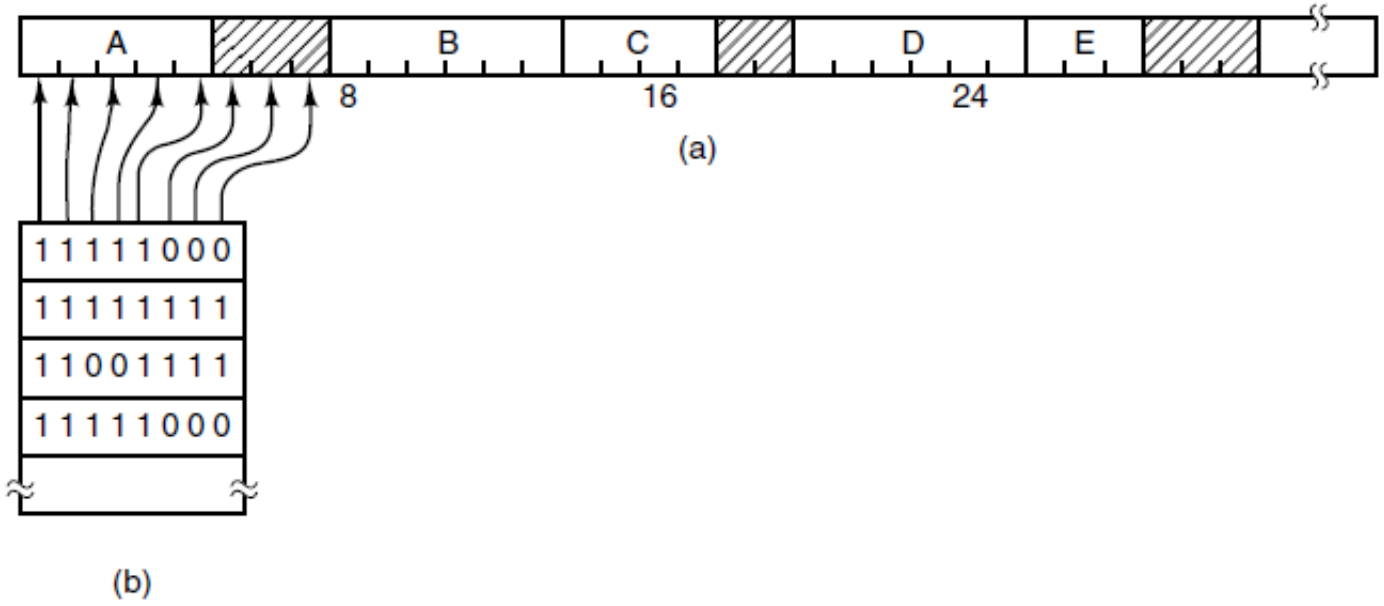
- How to find a free chunk of memory that can satisfy the user request?

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

Free-Space Management

- Bitmap
  - Divide memory into blocks
  - Use a bit 0/1 for every block to indicate whether it is free or not



Free-Space Management

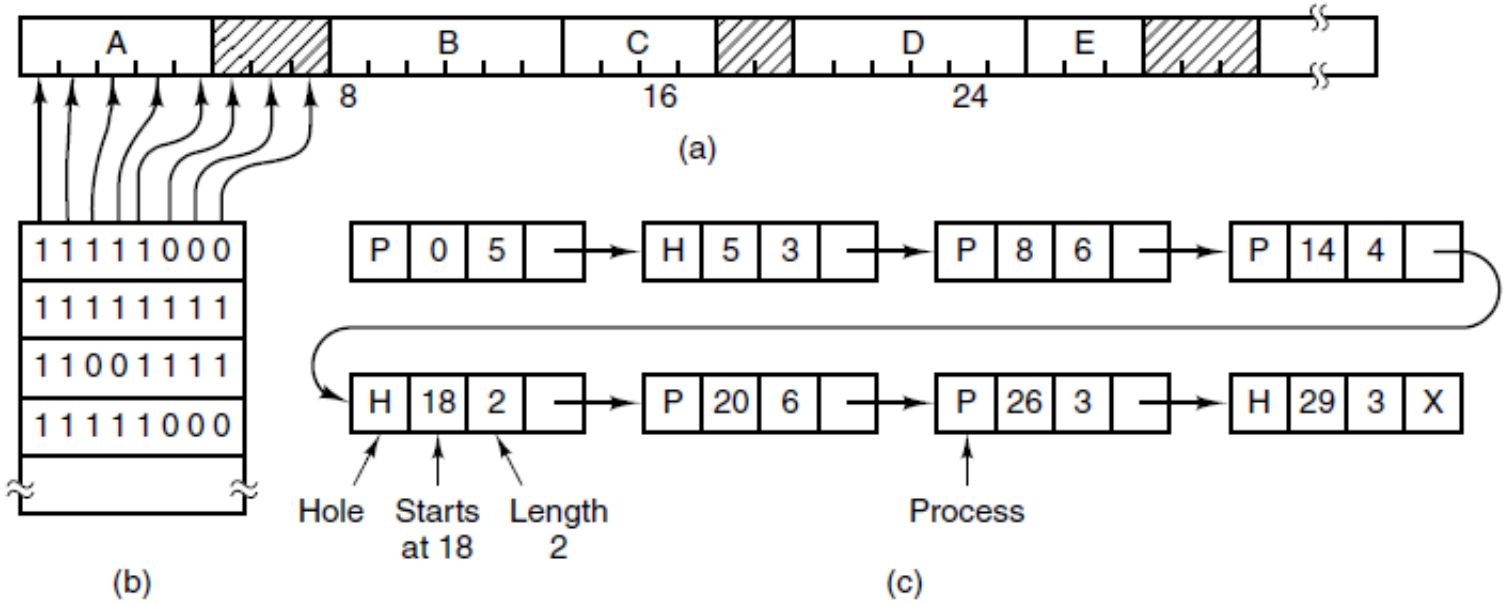
- Bitmap
  - Memory allocation
    - Suppose a process needs k blocks
    - Find k consecutive 0 bits in the bitmap
  - Memory deallocation
    - Reset the corresponding blocks in bitmap to 0

Free-Space Management

- Linked list
  - Version 1:
    - Put all memory blocks in a linked list
    - Each node in this list has four fields
      - First field: indicate free or not
      - Second field: indicate the start address
      - Third field: length of the memory segment
      - Fourth field: a pointer points to the next node

Free-Space Management

- Linked list
  - Version 1: e.g.



Agenda

- ~~Recap~~
- ~~Segmentation~~
- ~~Free-Space Management~~

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.