Address Translation

Operating System (OS)
안인규

강의자료는 KAIST 원유집 교수님 OS 강의 자료를 참고했습니다.

Memory Virtualizing with Efficiency and Control

- Memory virtualizing takes a similar strategy known as **limited direct execution(LDE)** for efficiency and control.
- In memory virtualizing, efficiency and control are attained by hardware support
  - e.g., registers, TLB(Translation Look-aside Buffer)s, page-table

Address Translation

- Hardware transforms a **virtual address** to a **physical address**.
  - The desired information is actually stored in a physical address.

- The OS must get involved at key points to set up the hardware.
  - OS는 메모리를 관리하여 적절한 순간에 개입할 수 있어야 함

Example: Address Translation

- C - Language code

```
void func()
        int x=3000;
        ...
        x = x + 3; // this is the line of code we are interested in
```

- **Load** a value from memory
- **Increment** it by three
- **Store** the value back into memory

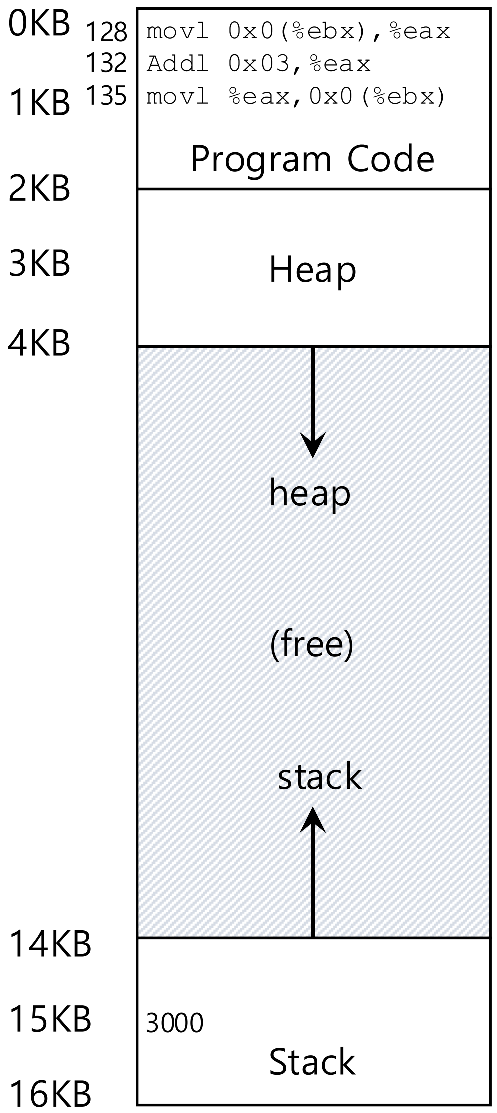
Example: Address Translation (Cont.)

- Assembly

```
128 : movl 0x0(%ebx), %eax     ; load 0+ebx into eax
132 : addl $0x03, %eax         ; add 3 to eax register
135 : movl %eax, 0x0(%ebx)     ; store eax back to mem
```

- Presume that the address of `'x'` has been place in `ebx` register.
- **Load** the value at that address into `eax` register.
- **Add** 3 to `eax` register.
- **Store** the value in `eax` back into memory.

Example: Address Translation (Cont.)

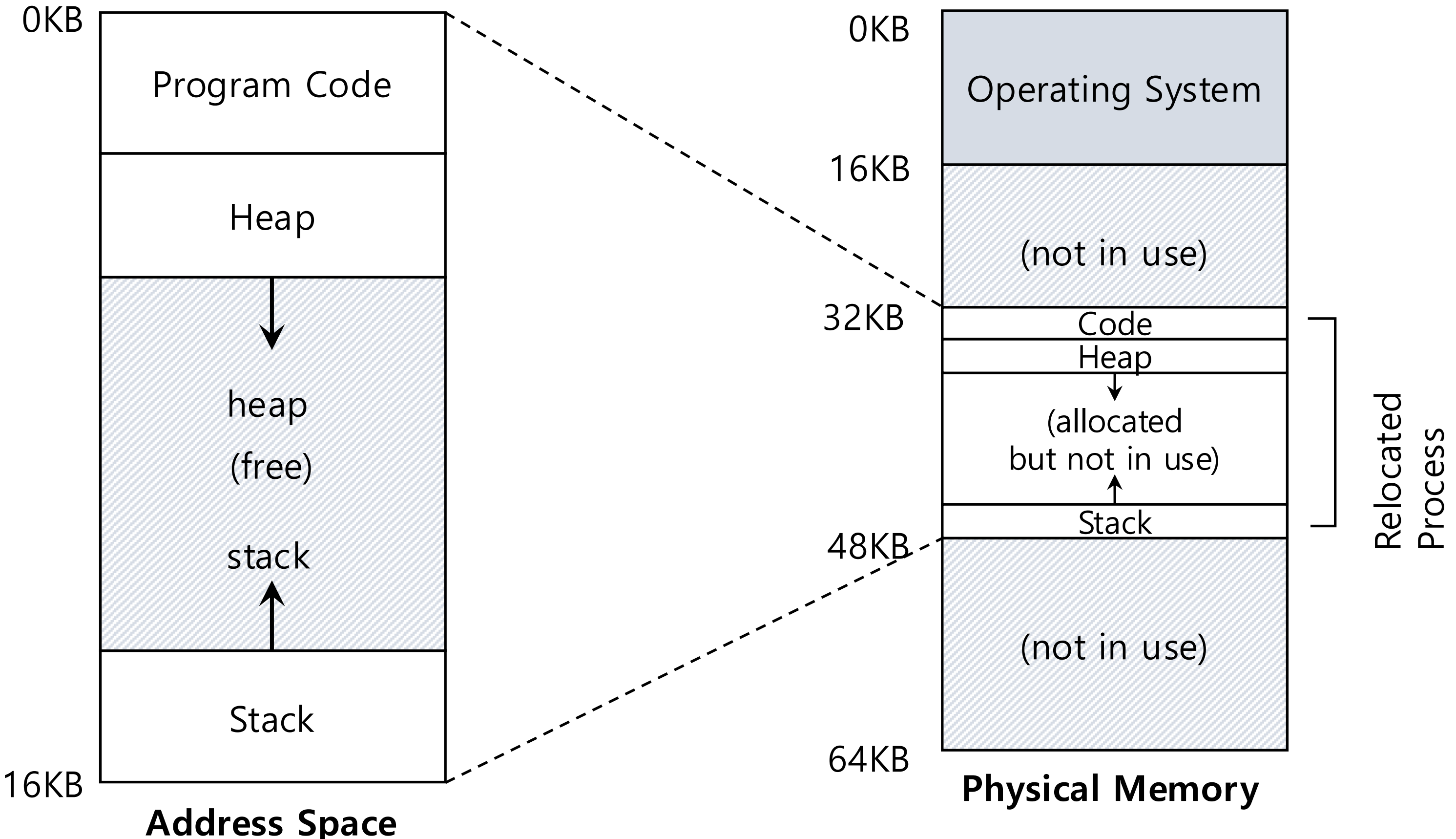


```
movl 0x0(%ebx),%eax
Addl 0x03,%eax
movl %eax,0x0(%ebx)
```

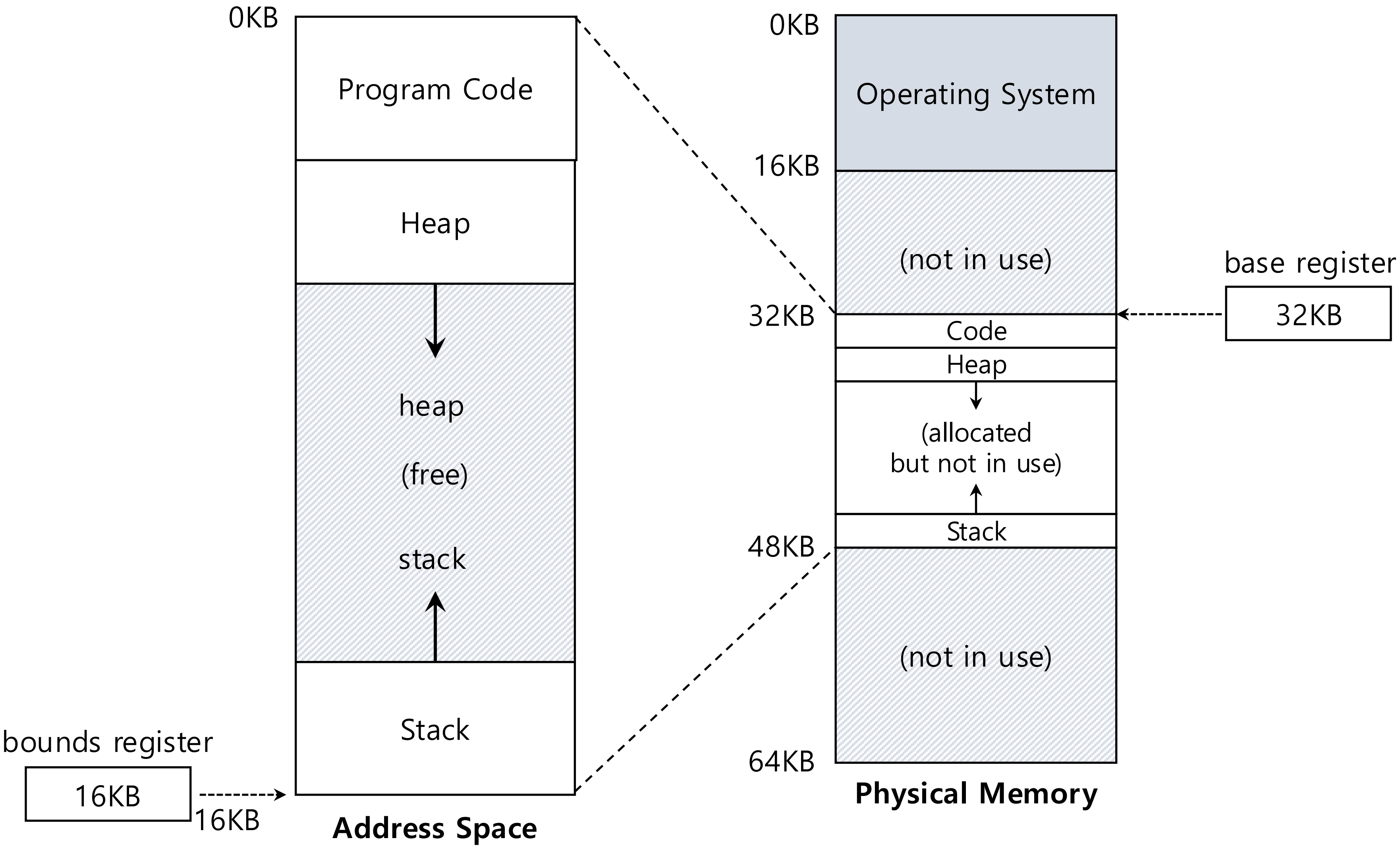
- Fetch instruction at address 128
- Execute this instruction (load from address 15KB)
- Fetch instruction at address 132
- Execute this instruction (no memory reference)
- Fetch the instruction at address 135
- Execute this instruction (store to address 15 KB)

Dynamic Relocation: Base and Bound Register

- The OS wants to place the process **somewhere else** in physical memory, not at address 0.
  - The address space start at address 0.



Base and Bound Register



Dynamic(Hardware base) Relocation

- When a program starts running, the OS decides **where** in physical memory a process should be **loaded**.
  - Set the **base** register a value.

$$phycal\ address = virtual\ address + base$$

  - Every virtual address must **not be greater than bound** and **negative.**

$$0 \leq \ virtual\ addressvirtual\ address < bounds$$

Relocation and Address Translation

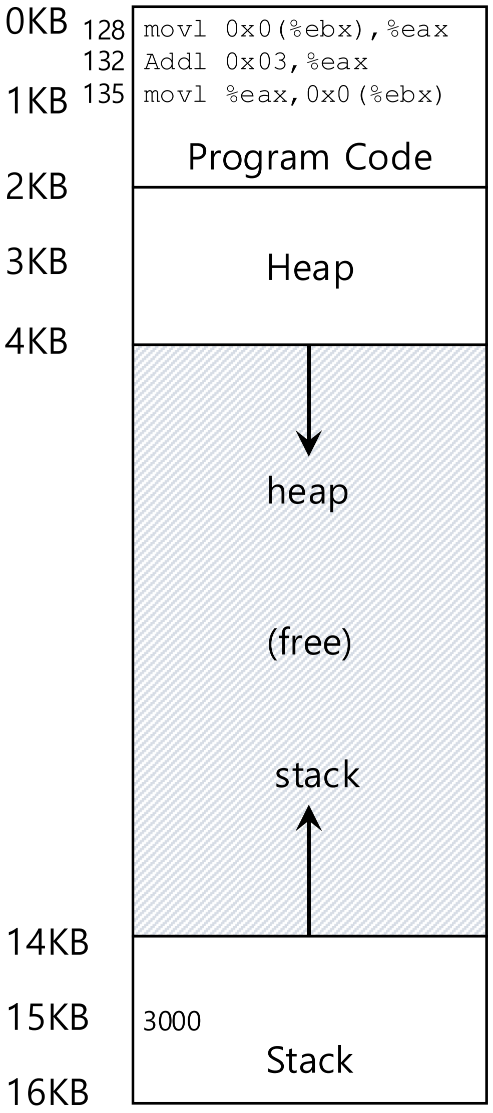
```
128 : movl 0x0(%ebx), %eax
```

- **Fetch** instruction at address 128

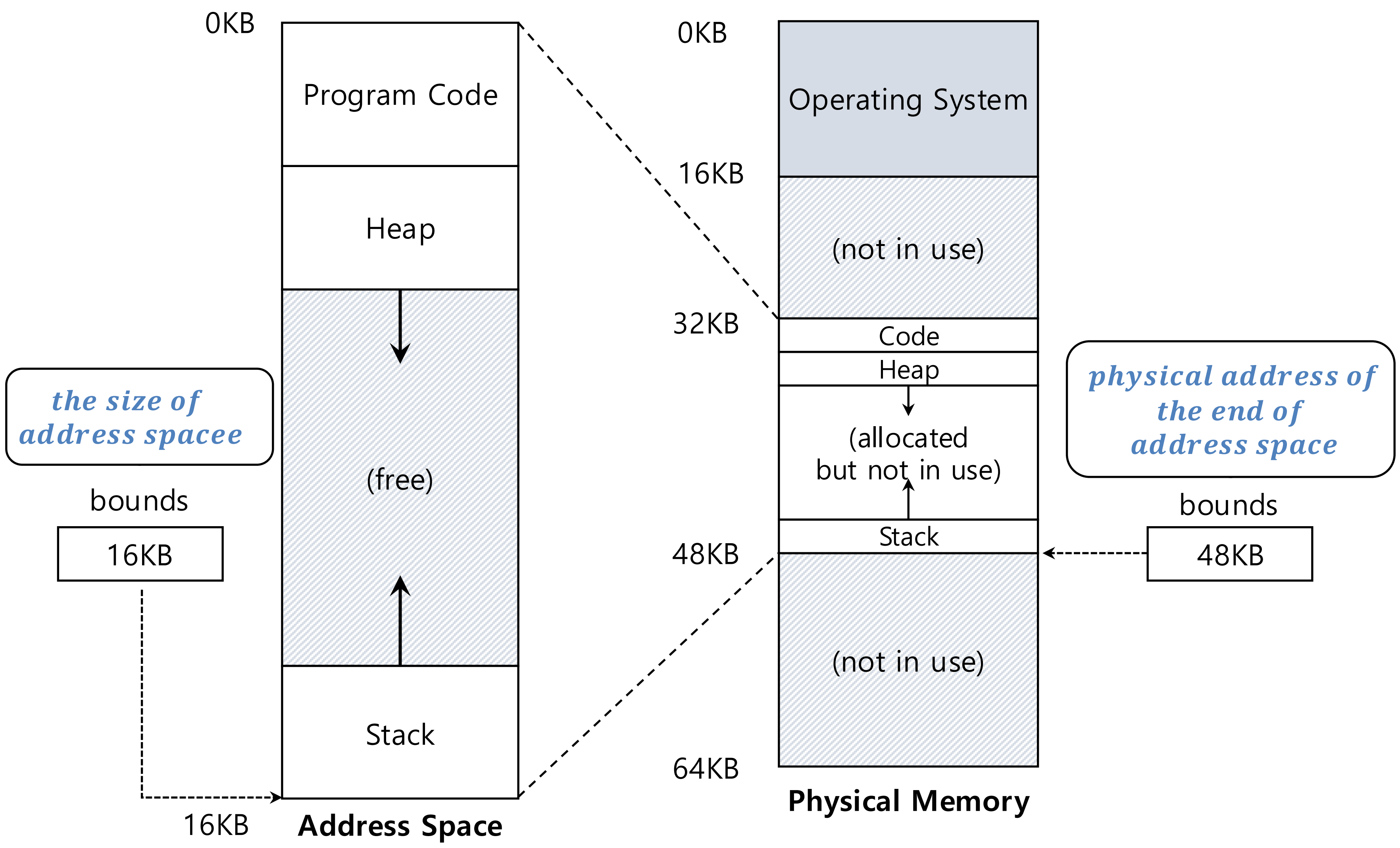
$$32896 = 128 + 32KB(base)$$

- **Execute** this instruction
  - Load from address 15KB

$$47KB = 15KB + 32KB(base)$$



Two ways of Bounds Register



Hardware Requirements

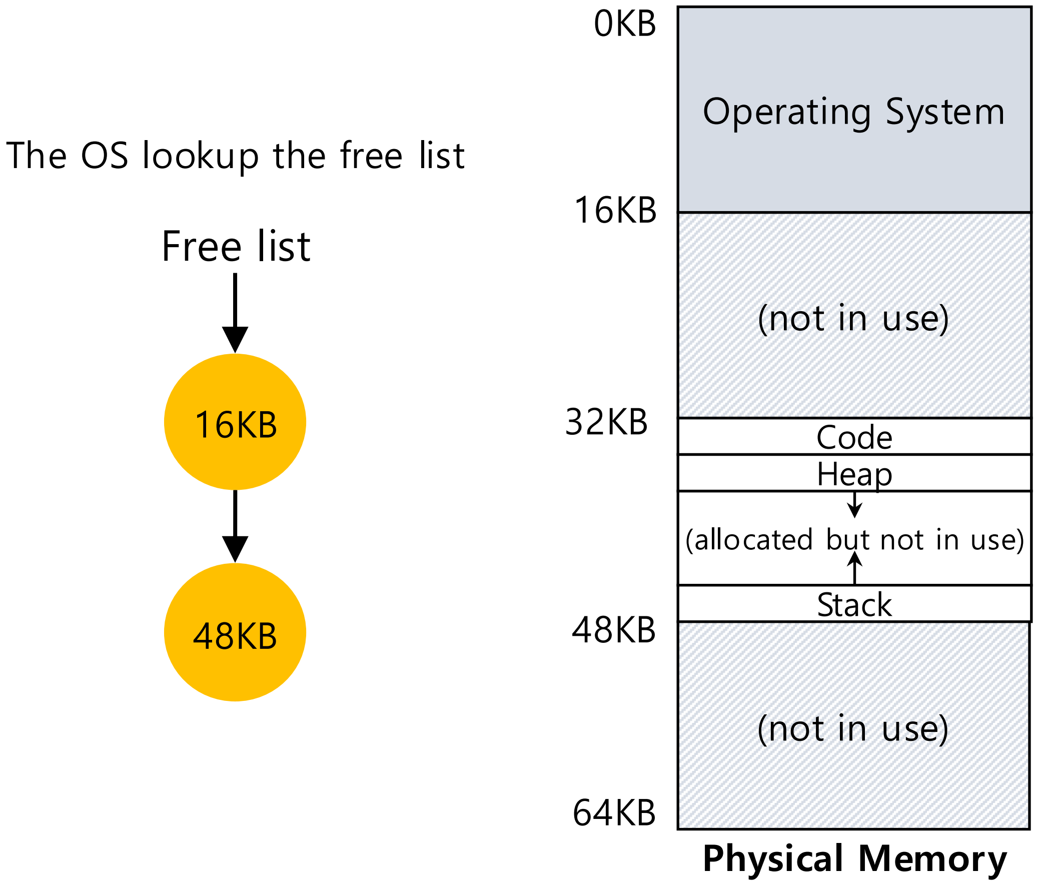
- **Privileged mode**: 사용자 모드의 프로세스가 권한이 필요한 작업(Privileged operations) 을 수행하지 못하도록 막는다.
- **Base/Bounds Registers**: 주소 변환과 범위 검사(Bounds Check)를 지원하려면 CPU당 한 쌍의 레지스터가 필요하다.
- **가상 주소를 물리 주소로 변환**하고, **주어진 범위 내에 있는지 확인**할 수 있어야 하며, 이를 위한 회로(Circuitry) 가 필요하다.
- **Privileged instruction(s) to update base/bounds**: 운영체제는 사용자 프로그램을 실행하기 전에 이 값들을 설정할 수 있어야 한다.
- **Privileged instruction(s) to register**: 운영체제는 예외(Exception) 발생 시 어떤 코드(예: 핸들러)를 실행할지 하드웨어에 알려줄 수 있어야 한다.
- **Ability to raise exceptions**: 사용자 프로세스가 특권 명령어를 실행하거나, 메모리 경계를 벗어난 접근을 시도할 경우, 예외를 발생시킬 수 있어야 한다.

OS Issues for Memory Virtualizing

- The OS must **take action** to implement **base-and-bounds** approach.
- Three critical junctures (중요 시점):
  - When a process **starts running:**
    - Finding space for address space in physical memory
  - When a process is **terminated:**
    - Reclaiming the memory for use
  - When context **switch occurs:**
    - Saving and storing the base-and-bounds pair

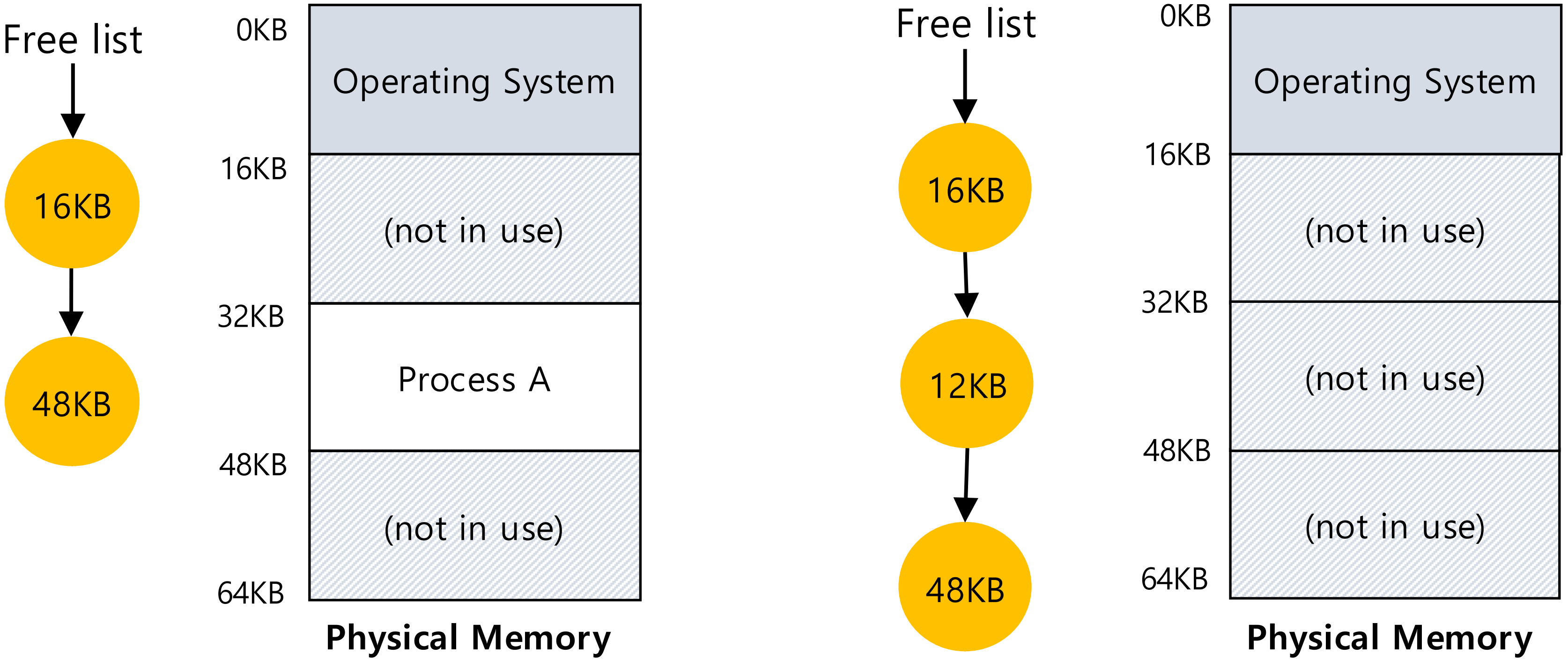
OS Issues: When a Process Starts Running

- The OS must **find a room** for a new address space.
  - free list : A list of the range of the physical memory which are not in use.



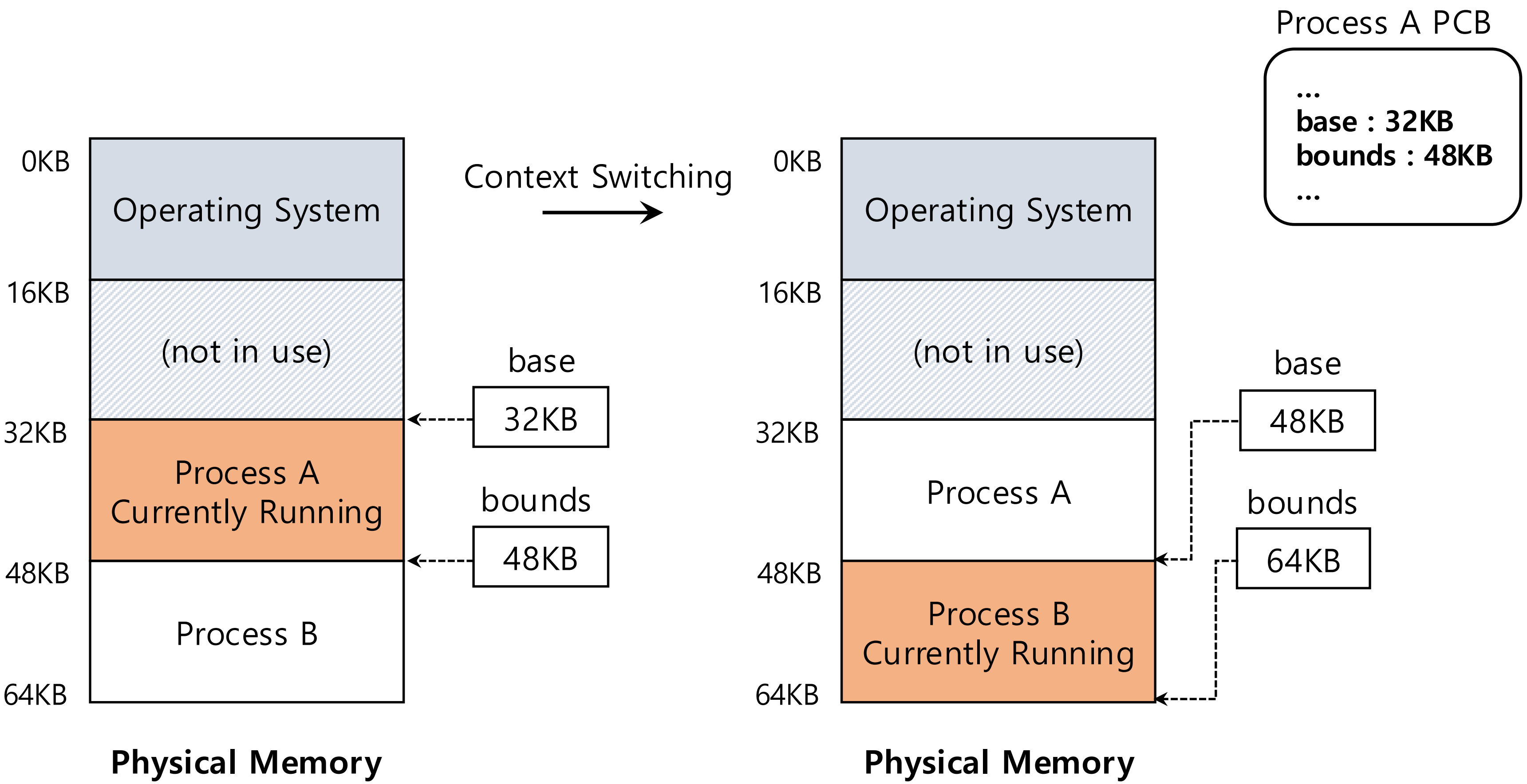
OS Issues: When a Process is Terminated

- The OS must **put the memory back** on the free list.



OS Issues: When Context Switch Occurs

- The OS must **save and restore** the base-and-bounds pair.
  - In **process structure** or **process control block(**PCB)



OS Issues: provide exception handlers

- the OS must provide exception handlers,
- the OS installs these handlers at boot time (via privileged instructions
    - Exception handler for segmentation fault

Summary

- Address translation: hardware support and OS support
- Basic form: base and bound
- Fragmentation issue