Operating Systems

15. Address Translation

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.
  - The OS must manage memory to judiciously intervene.

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

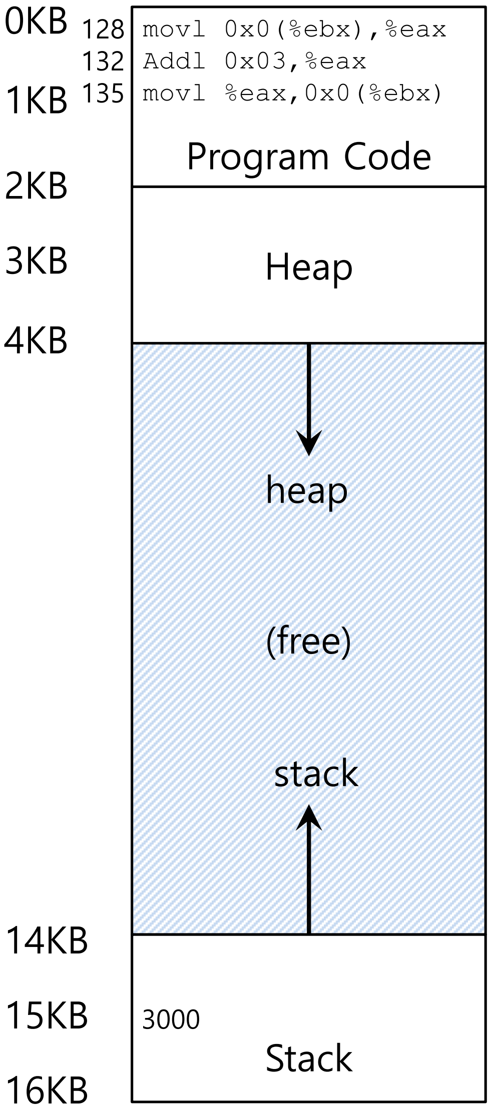
Example: Address Translation(Cont.)

- 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
```

- **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.)

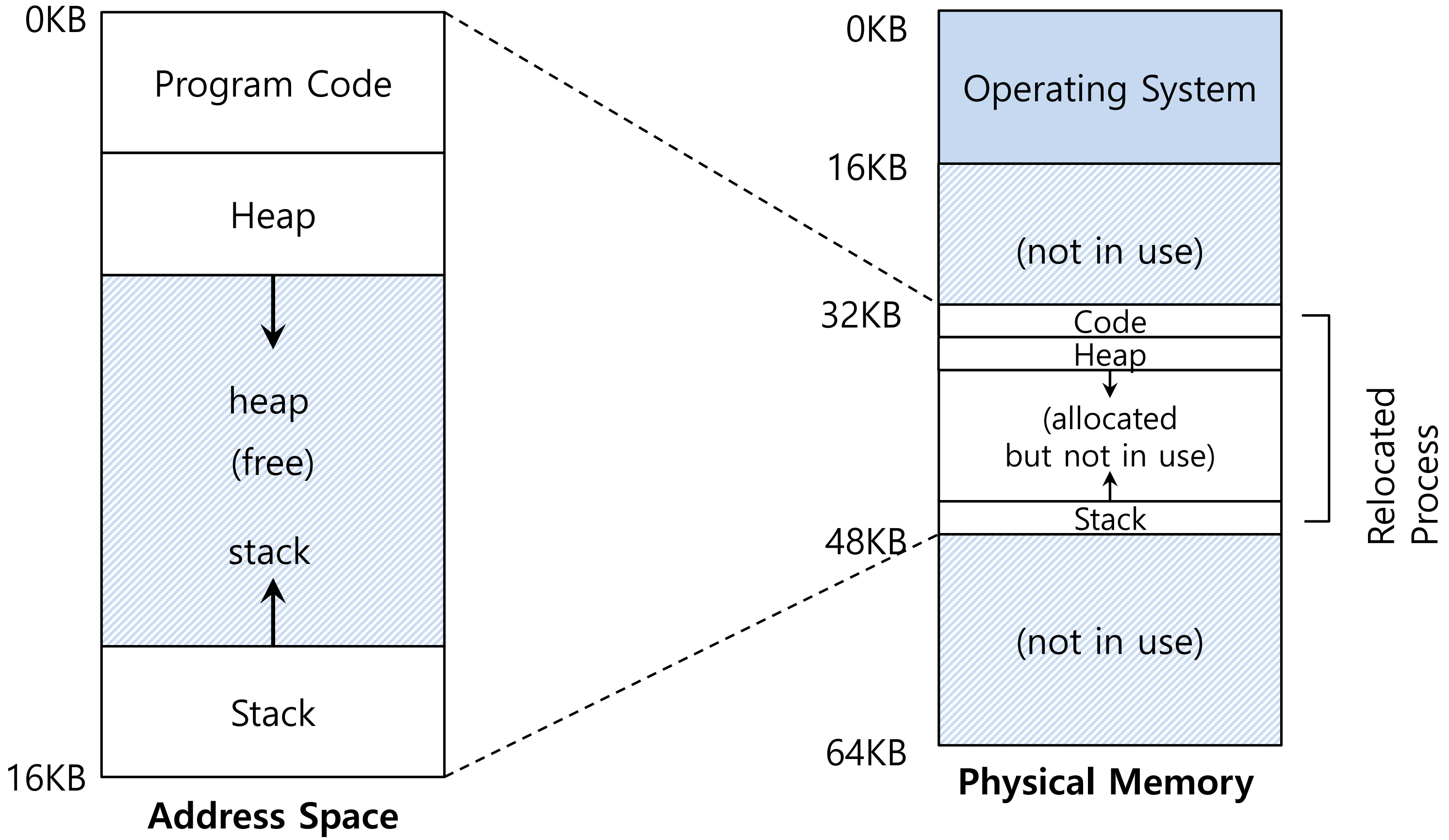


```
movl  0x0(%ebx),%eax
Addl  0x03,%eax
movl  %eax,0x0(%ebx)
```

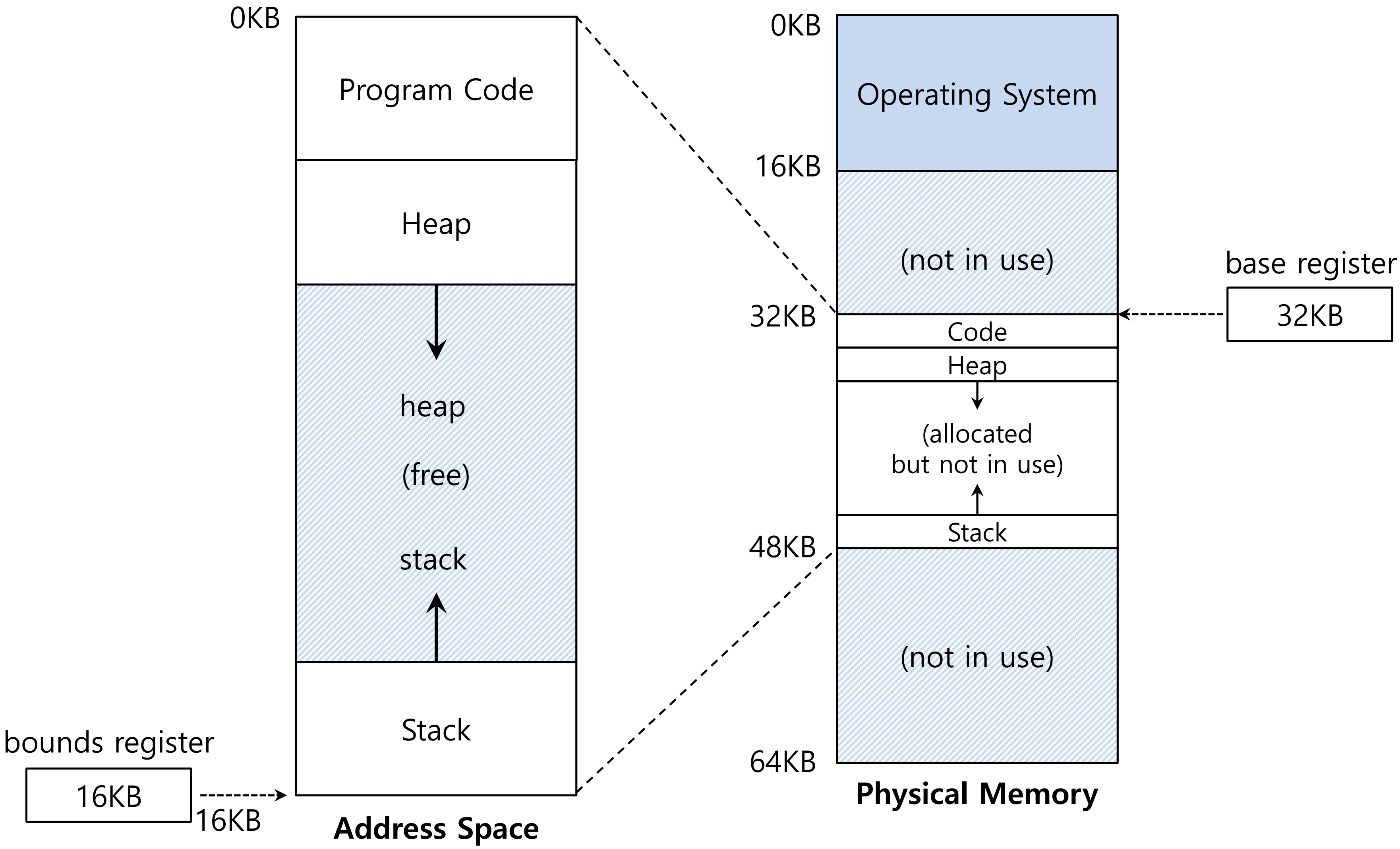
- Fetch instruction at address 128
- Execute this instruction (load from address 15KB)
- Fetch instruction at address 132
- Execute this instruction (no memory reference)
- Fetch the instruction at address 135
- Execute this instruction (store to address 15 KB)

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 Bounds 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

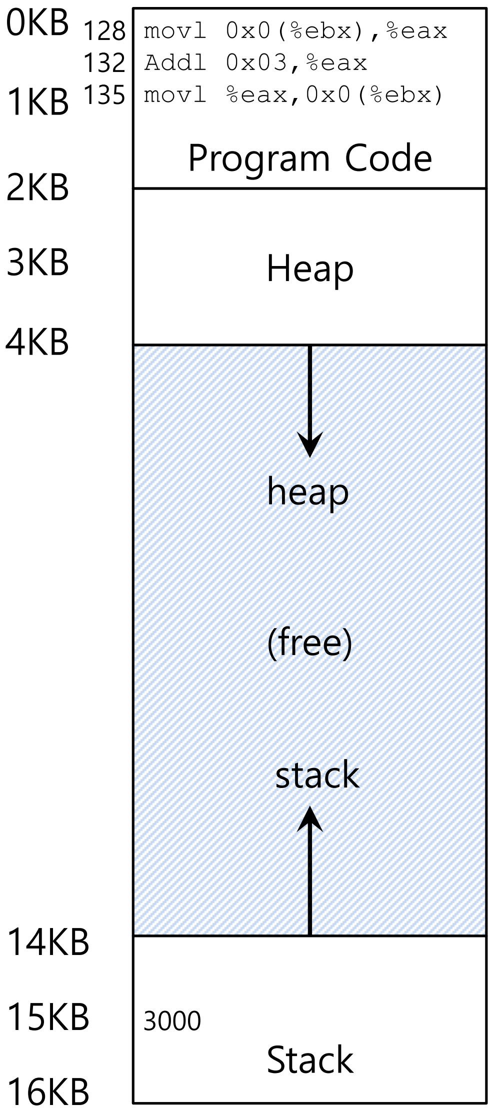
```
128 : movl 0x0(%ebx), %eax
```

- **Fetch** instruction at address 128

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

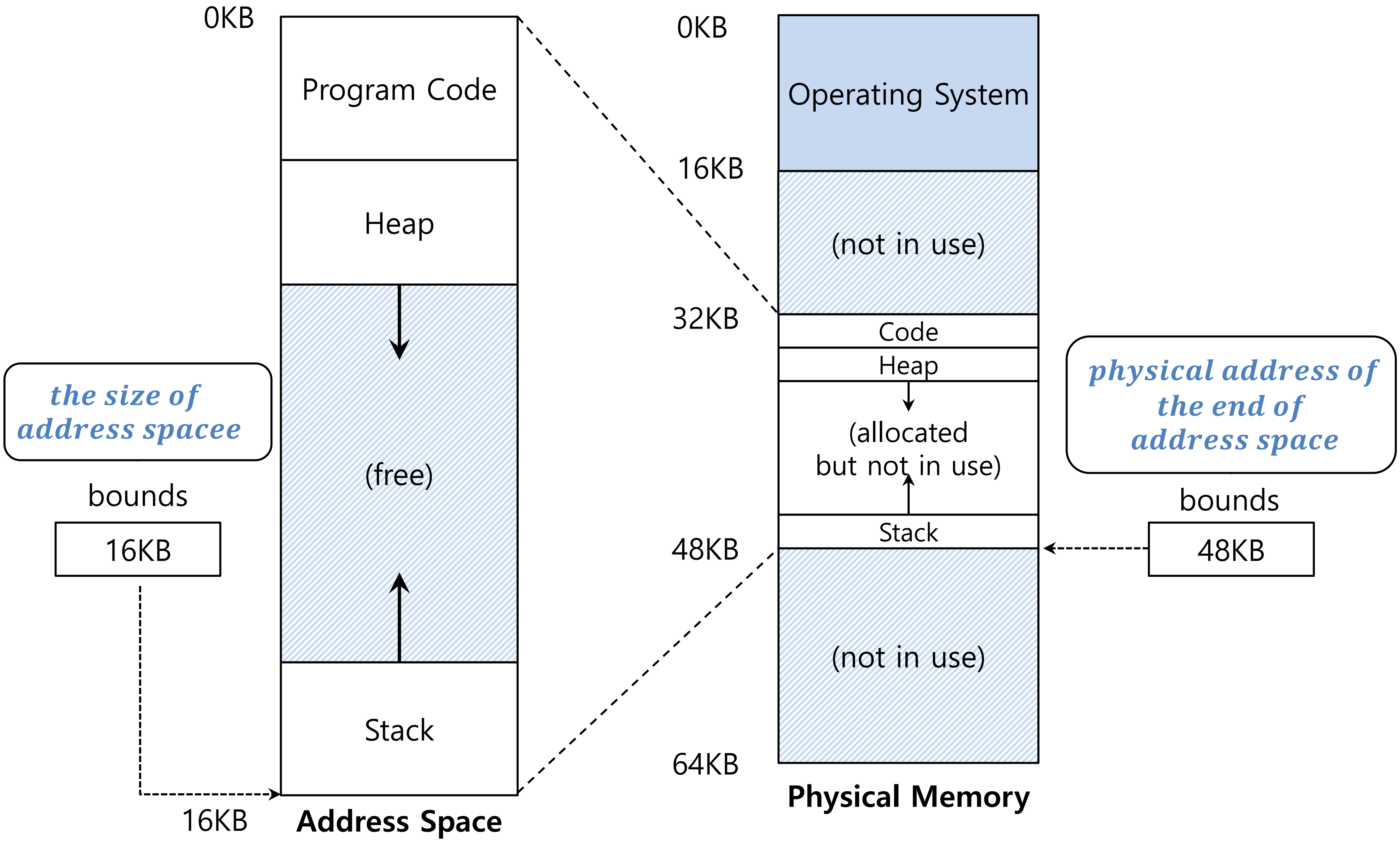
- **Execute** this instruction
  - Load from address 15KB

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



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

Two ways of Bounds Register



Hardware Requirements

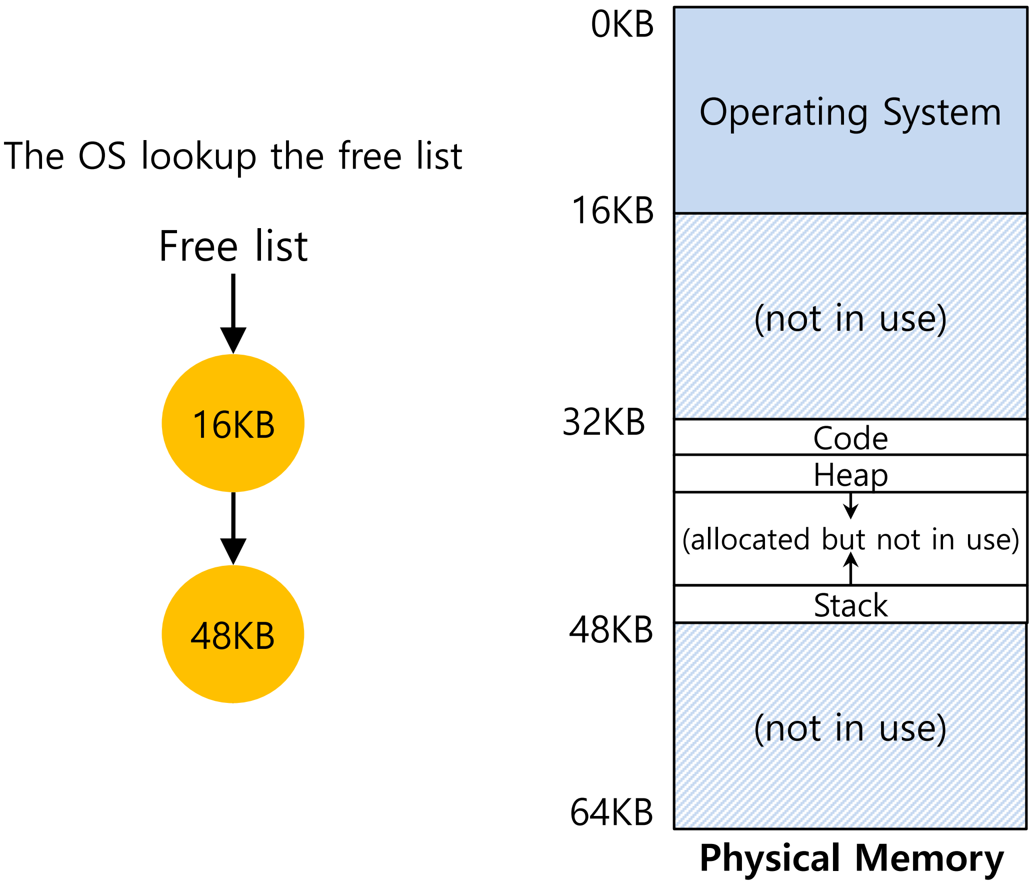
- Privileged mode: prevent user-mode processes from executing privileged operations
- Base/bounds registers: Need pair of registers per CPU to support address translation and bounds checks
- Ability to translate virtual addresses and check if within bounds limits; Circuitry to do translations.
- Privileged instruction(s) to update base/bounds: OS must be able to set these values before letting a user program run
- Privileged instruction(s) to register: OS must be able to tell hardware what exception handlers code to run if exception occurs
- Ability to raise exceptions when processes try to access privileged instructions or out-of-bounds memoryl

OS Issues for Memory Virtualizing

- The OS must **take action** to implement **base-and-bounds** approach.
- Three critical junctures:
  - 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

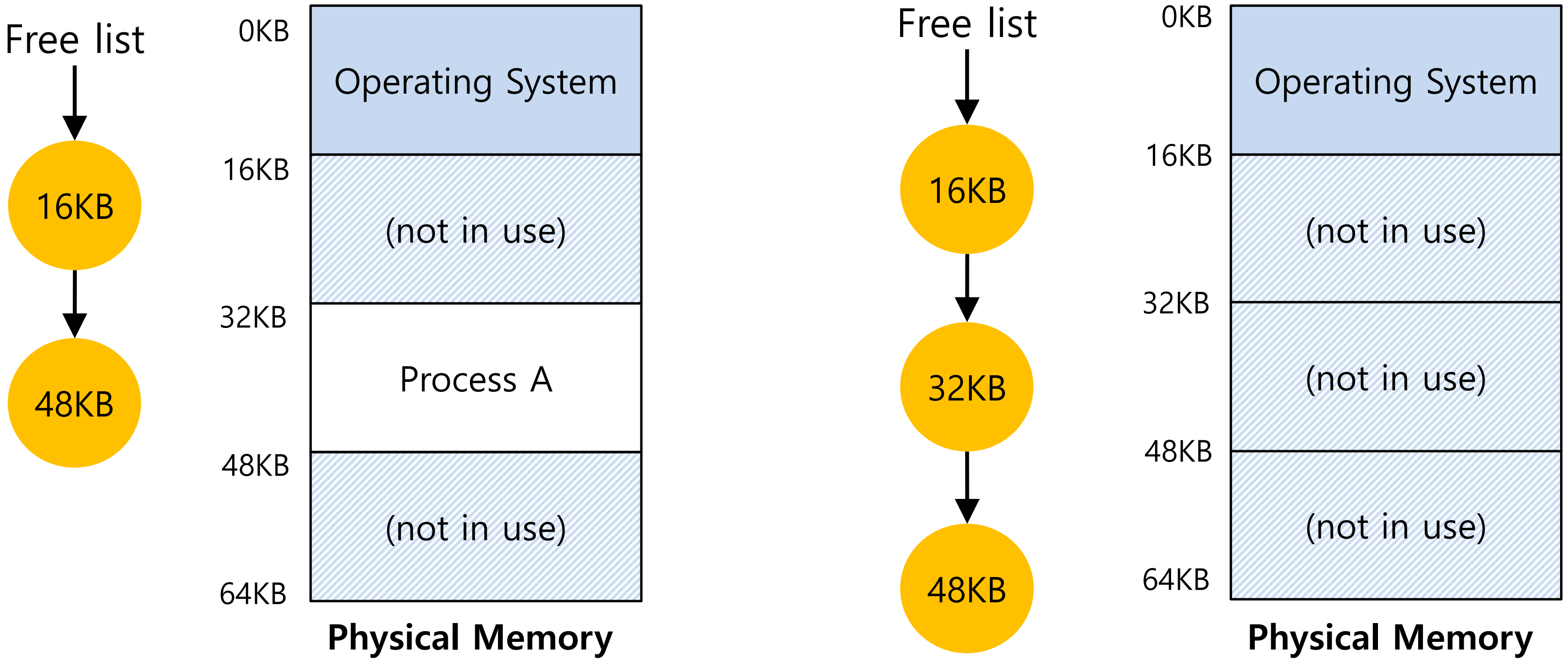
OS Issues: When a Process Starts Running

- 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.



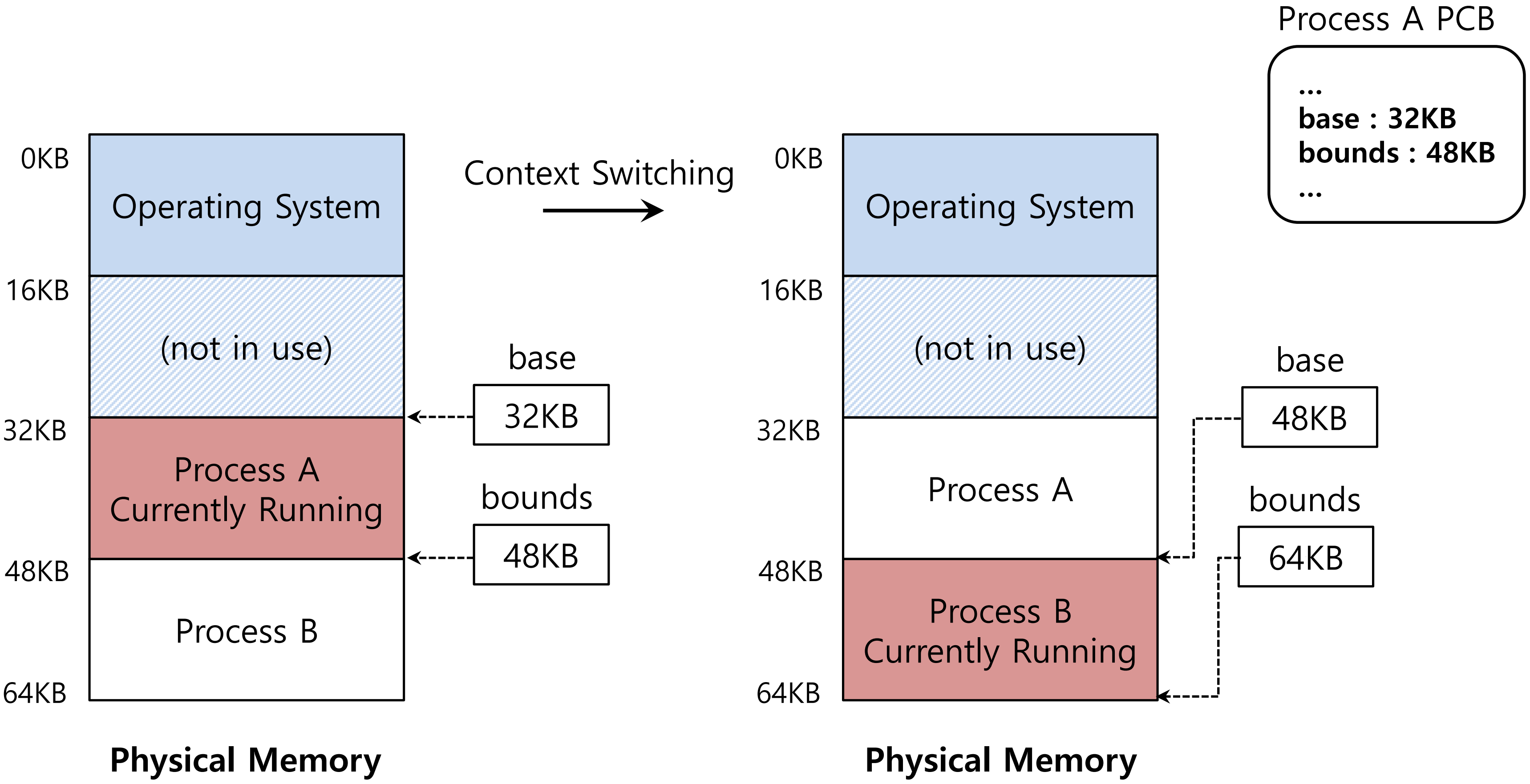
OS Issues: When a Process Is Terminated

- 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