Operating System: Address Translation

Sang Ho Choi (shchoi@kw.ac.kr)
School of Computer & Information Engineering
KwangWoon University

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

메모리 가상화에도 limited direct execution 전략이 사용되는데 이는 cpu 가상화때와 마찬가지로 자원을 효율적으로 제어하기 위함임
효율적으로 제어하기 위해서는 하드웨어의 지원이 필요한데 위와 같은 것들이 있음

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
  - Keeping track of which locations are free and which are in use
- Assumption 이와같은가정을두고 설명
  - The user's address space must be placed contiguously in physical memory
  - The size of the address space is less than the size of physical memory
  - Each address space is exactly the same size

Example: Address Translation

- C – Language code

```
void func()
        int x;
        ...
        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

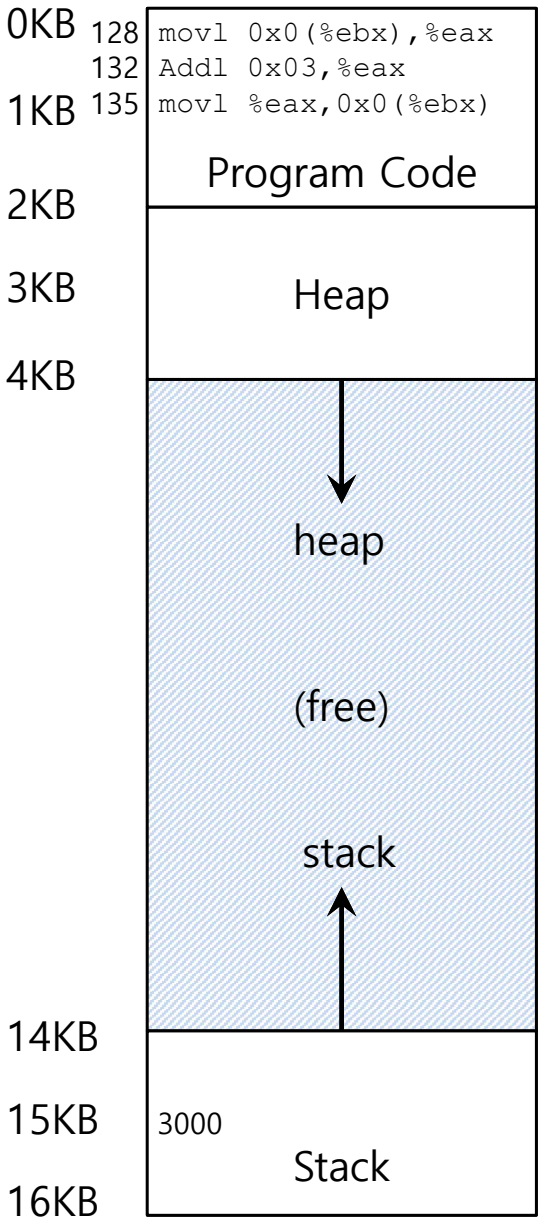
Example: Address Translation (Cont.)

- 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.)

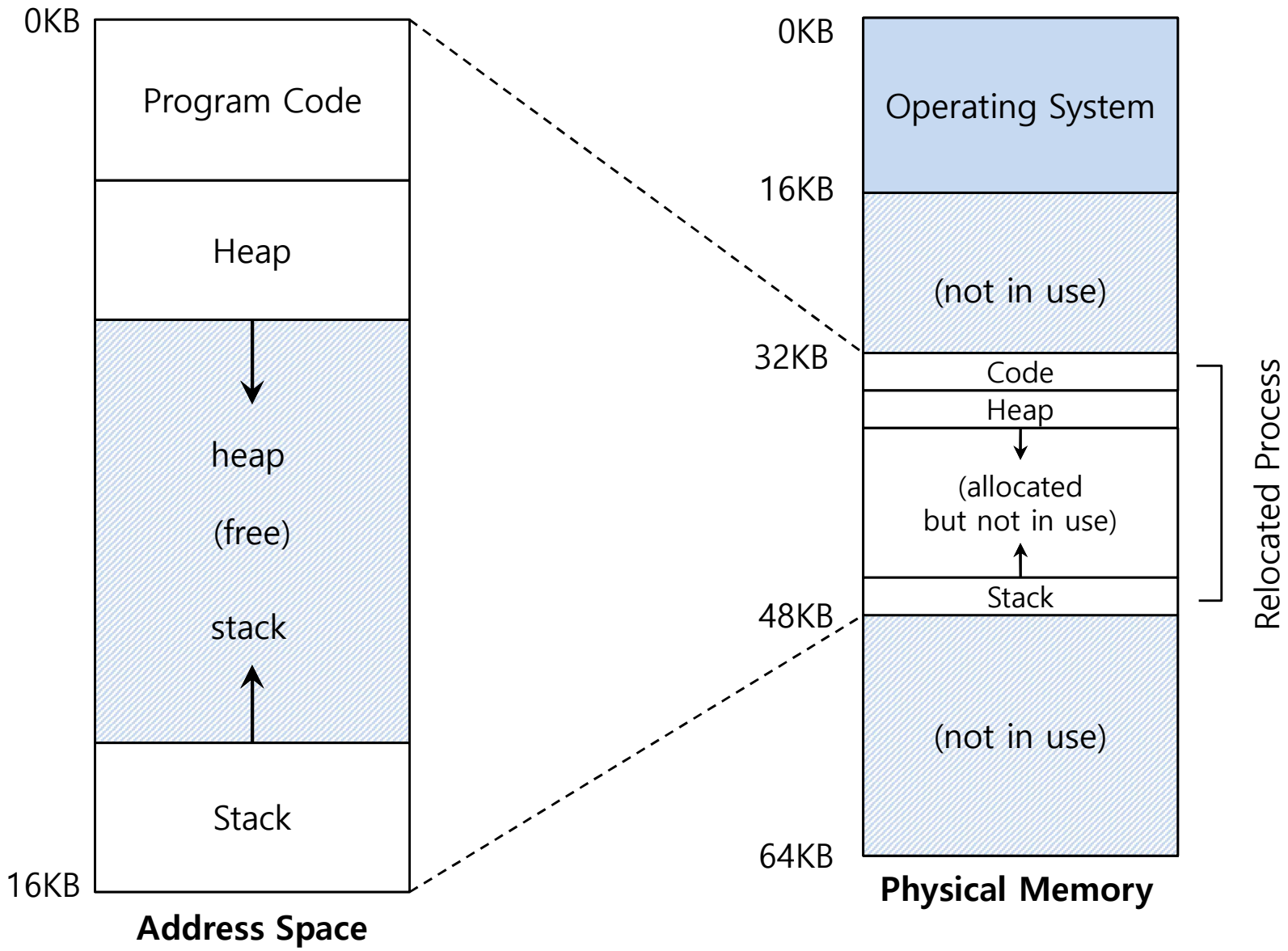


- Fetch instruction at address 128
- Execute this instruction (load from address 15KB)
- Fetch instruction at address 132
- Execute this instruction (no memory reference)
- Fetch the instruction at address 135
- Execute this instruction (store to address 15 KB)

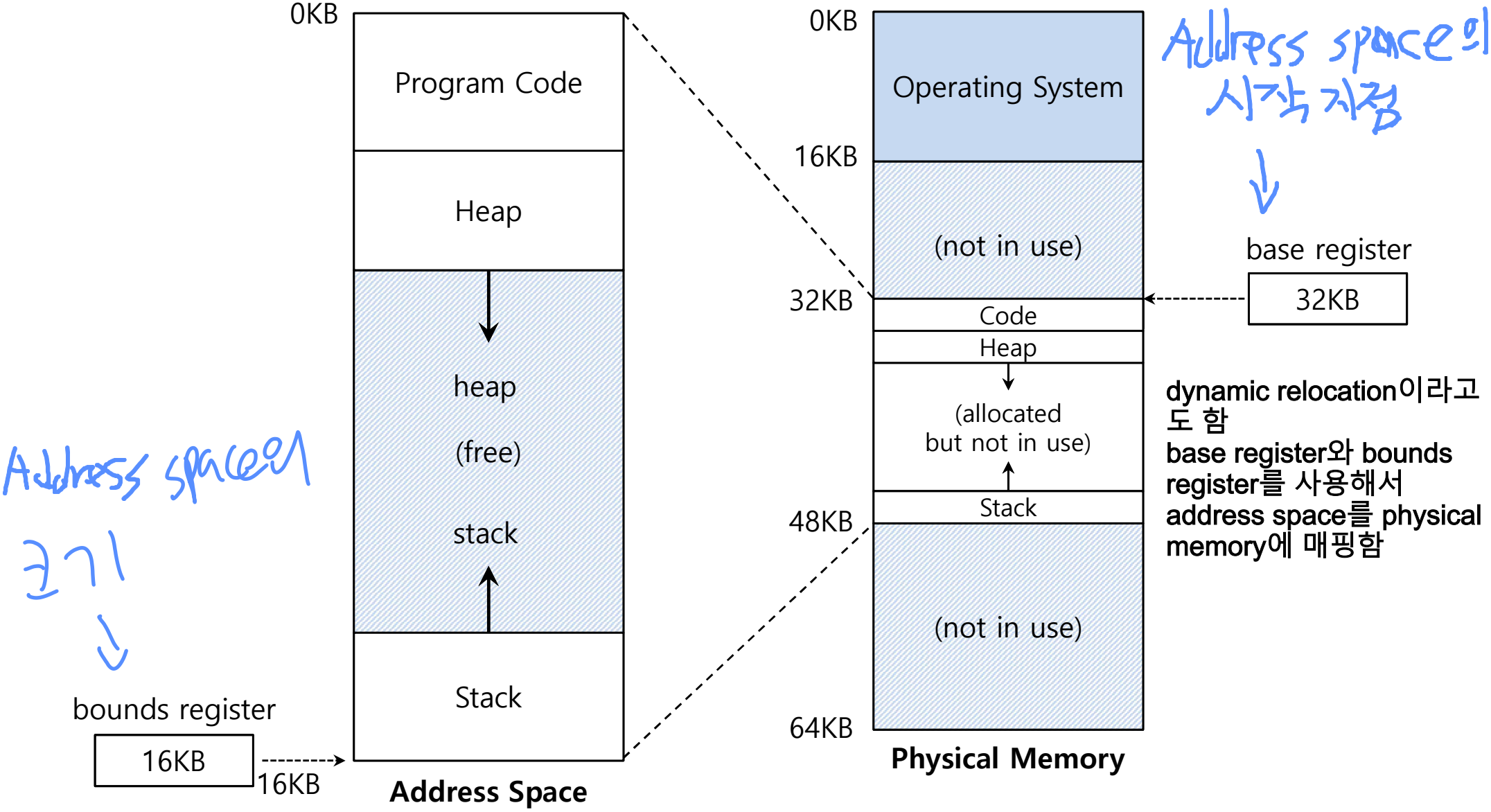
Relocation Address Space

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

A Single Relocated Process



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

$$physical\ address = virtual\ address + base$$

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

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

bound register는 virtual address가 bound크기보다 작아야하므로 현재 프로그램이 다른 프로그램 메모리공간을 침해하지 않는지를 체크할 수 있음
즉, protection 역할

base, bound는 cpu칩내에 존재하는 하드웨어 한 쌍

Relocation and Address Translation

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

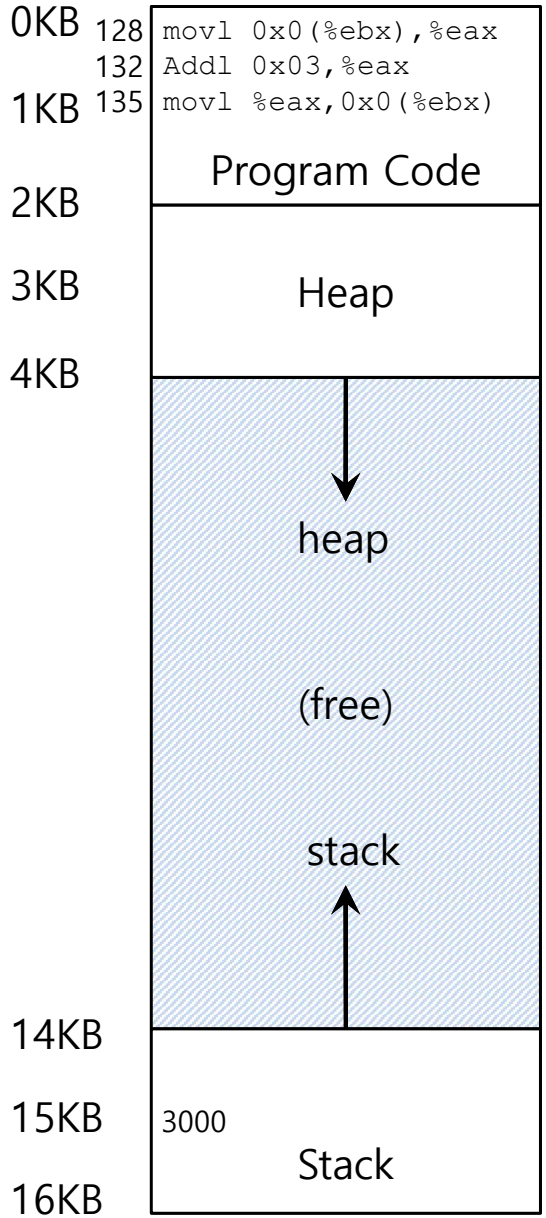
– Fetch instruction at address 128

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

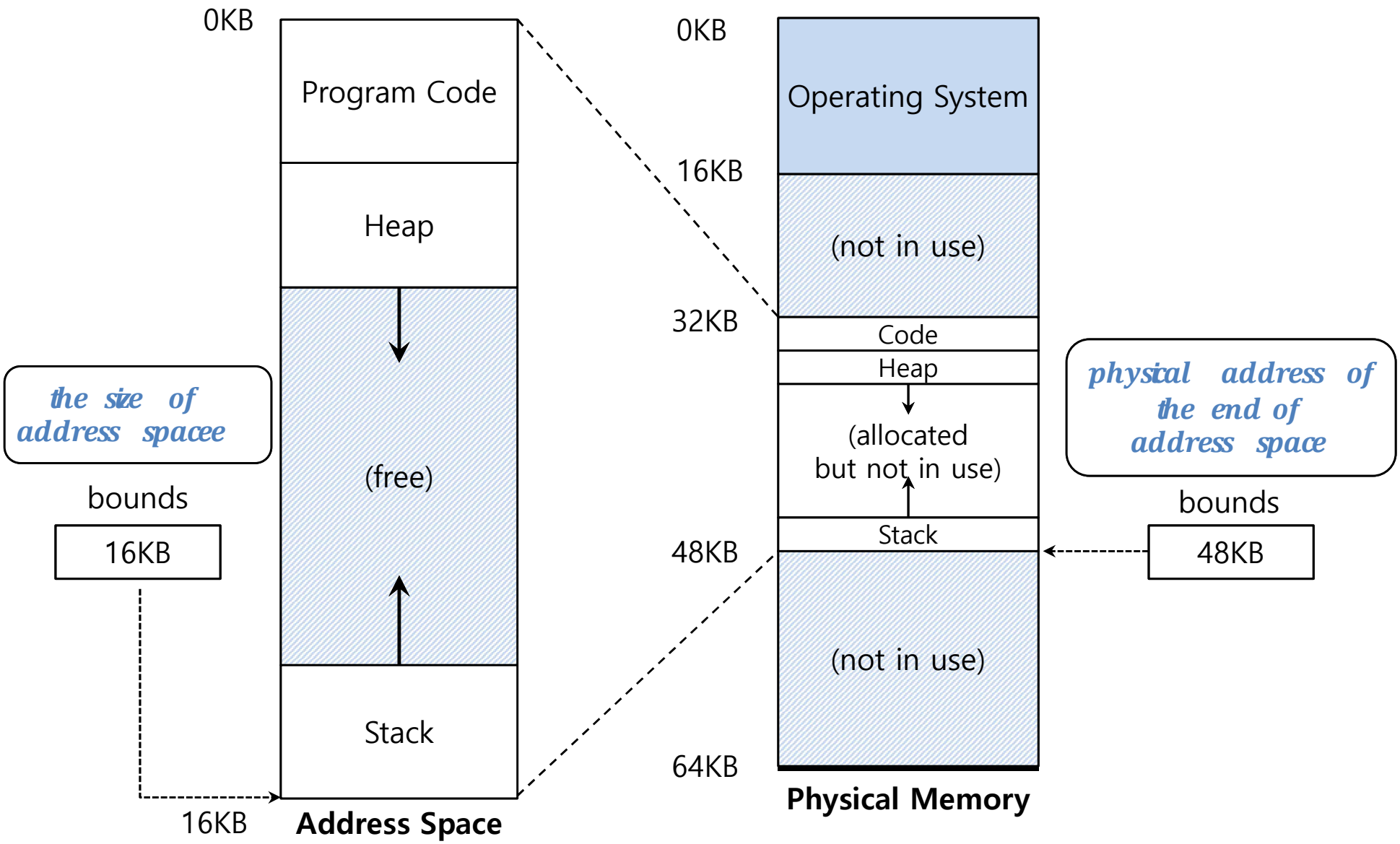
– Execute this instruction

➢Load from address 15KB

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



Two ways of Bounds Register



OS Issues for Memory Virtualizing

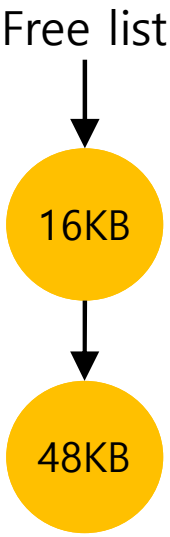
- The OS must take action to implement **base-and-bounds** approach
- Three critical junctures:
  - When a process **starts running:**
    - Finding space for address space in physical memory
  - When a process is **terminated:**
    - Reclaiming the memory for use
  - When context **switch occurs:**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

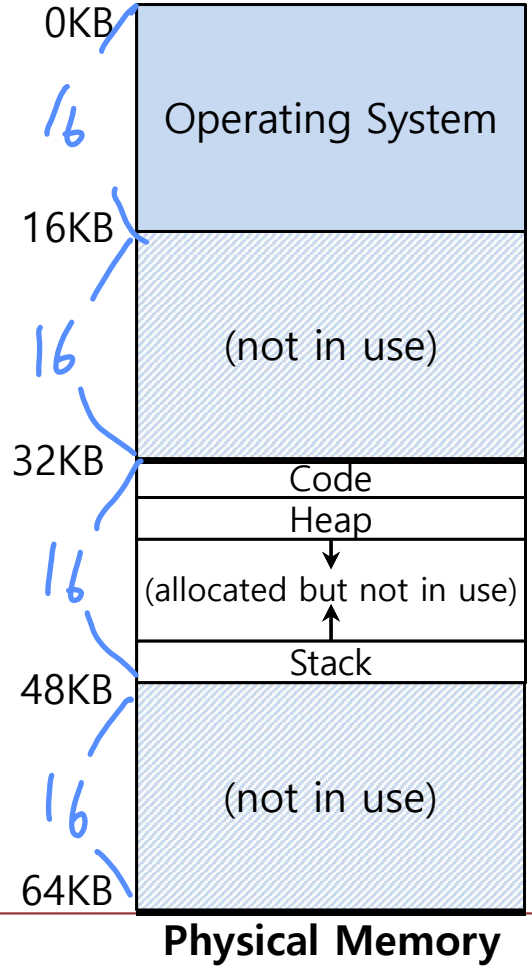
OS Issues: When a Process Starts Running

- 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

초기 가정에서 블록들끼리 크기가 다 같다고 했었음 그래서 free list에 여유 메모리의 시작지점만 명시되어 있는 것
블록의 사이즈가 다 개별적이라면 여유메모리의 시작점과 크기도 함께 명시했을 것

The OS lookup the free list

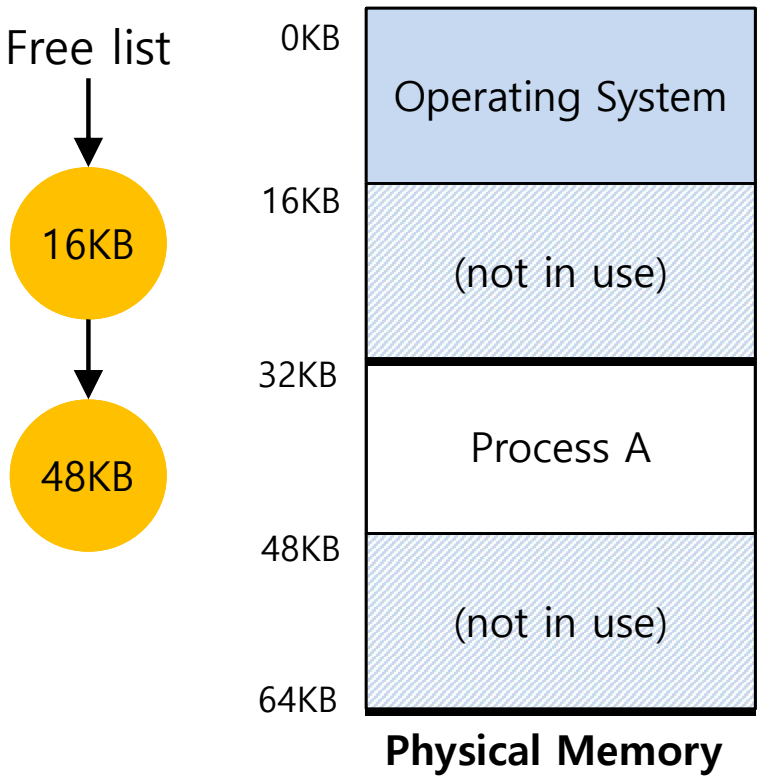


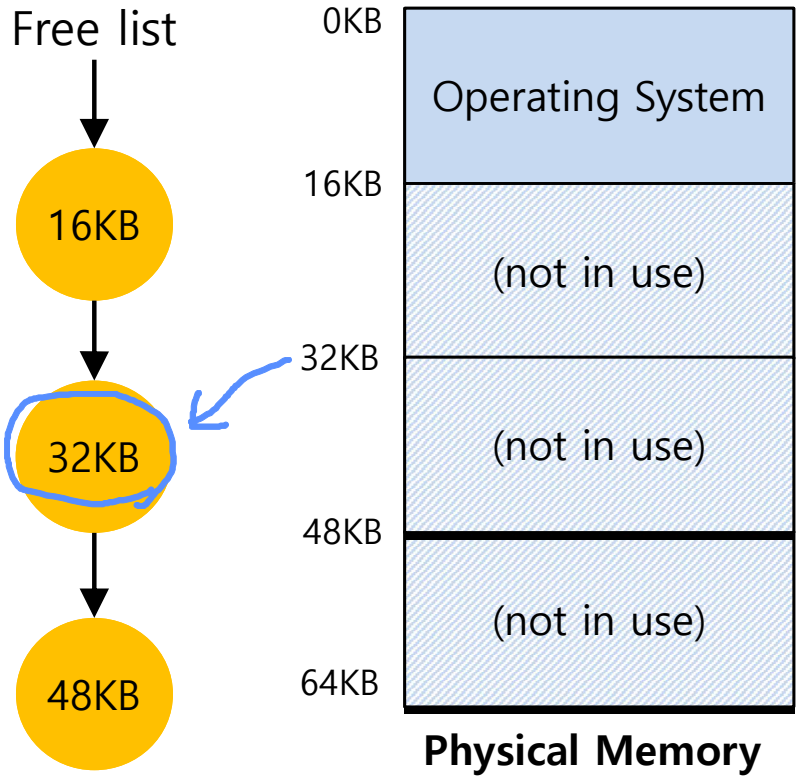


OS Issues: When a Process is Terminated

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

회수





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)