Memory Management 1

Minsoo Ryu

Operating Systems and Distributed Computing Lab.
Hanyang University

msryu@hanyang.ac.kr

Topics Covered

- Introduction
- Memory Allocation and Fragmentation
- Address Translation

Introduction

- ❒ CPU scheduling allows processes to share CPU
    - ▪ Improving both the CPU utilization and the response speed

- ❒ To realize this,
    - ▪ We must keep several processes in memory
    - ▪ This entails many complex problems for memory management

- ❒ Memory management is one of the most complex parts of the OS
    - ▪ Serves many different purposes

Introduction

❒ General goals of memory management

- **Provide a single contiguous, protected memory space to each process, make memory sharing easy for different processes, and allow for flexible memory management**
- **Provide a larger separate memory space to every process than the physically available memory space**
  - Every process can be allowed to use a 4GB memory space even though the physical memory is 1GB

❒ Tricks used by OS

물리적으로 제한된 메모리를 최대한 값이 사용할수있도록 하는 방법

- **Noncontiguous physical memory allocation via address translation**
  - Paging or segmentation
  - Differentiate addresses seen by each process from the real addresses
- **Allocate memory on demand (demand paging)**

Memory Allocation and Address Binding

배정

❒ Address binding ← 메모리를 배정함

- Assign memory addresses to all instructions and data

  메모리상의 위치를 정함

❒ Three phases of address binding

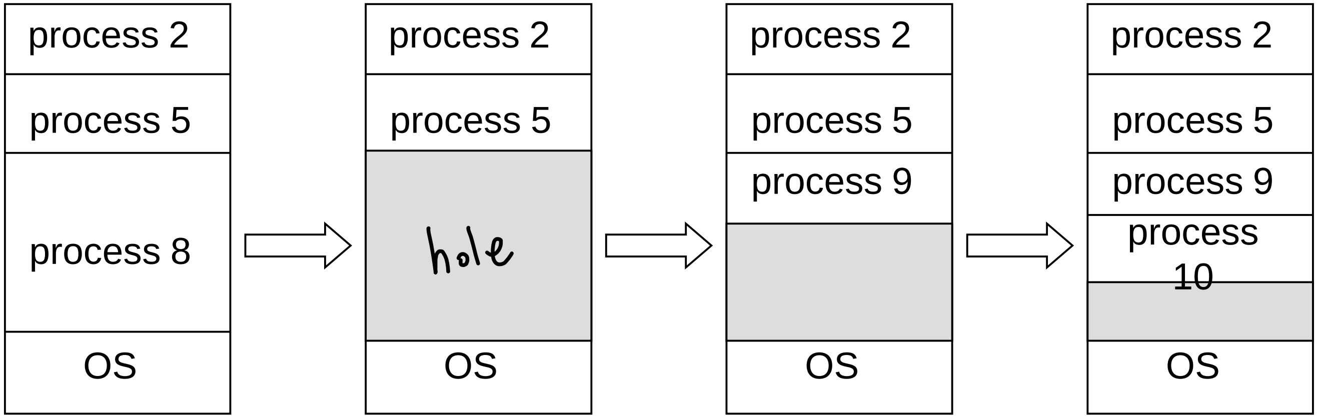
- **Compile time** (embedded system)
  - If memory location can be known a priori,
  - Absolute code can be generated; must recompile code if starting location changes
- **Load time** (general)
  - A compiler may generate relocatable code if memory location is not known at compile time
- **Execution time**
  - Binding is delayed until run time if the process can be moved during its execution from one memory segment to another

Contiguous Memory Allocation

문제 1) 효율성 보장 X
2) Fragmentation 문제

❒ CPU requires contiguous memory

- **When a process arrives, it is allocated memory from a hole large enough to accommodate it**
  - Hole – block of available memory; holes of various size are scattered throughout memory
- **Operating system maintains information about:**
  a) allocated partitions   b) free partitions (hole)



Fragmentation Problem

낭비되는 메모리공간

❐ Two types of fragmentation

→ ▪ **External Fragmentation** 배정된 메모리 공간 사이에 존재하는 hole 이 너무 작아서 프로세스가 배정되지 못하고 낭비되는 공간

- Total memory space exists to satisfy a request, but it is not contiguous

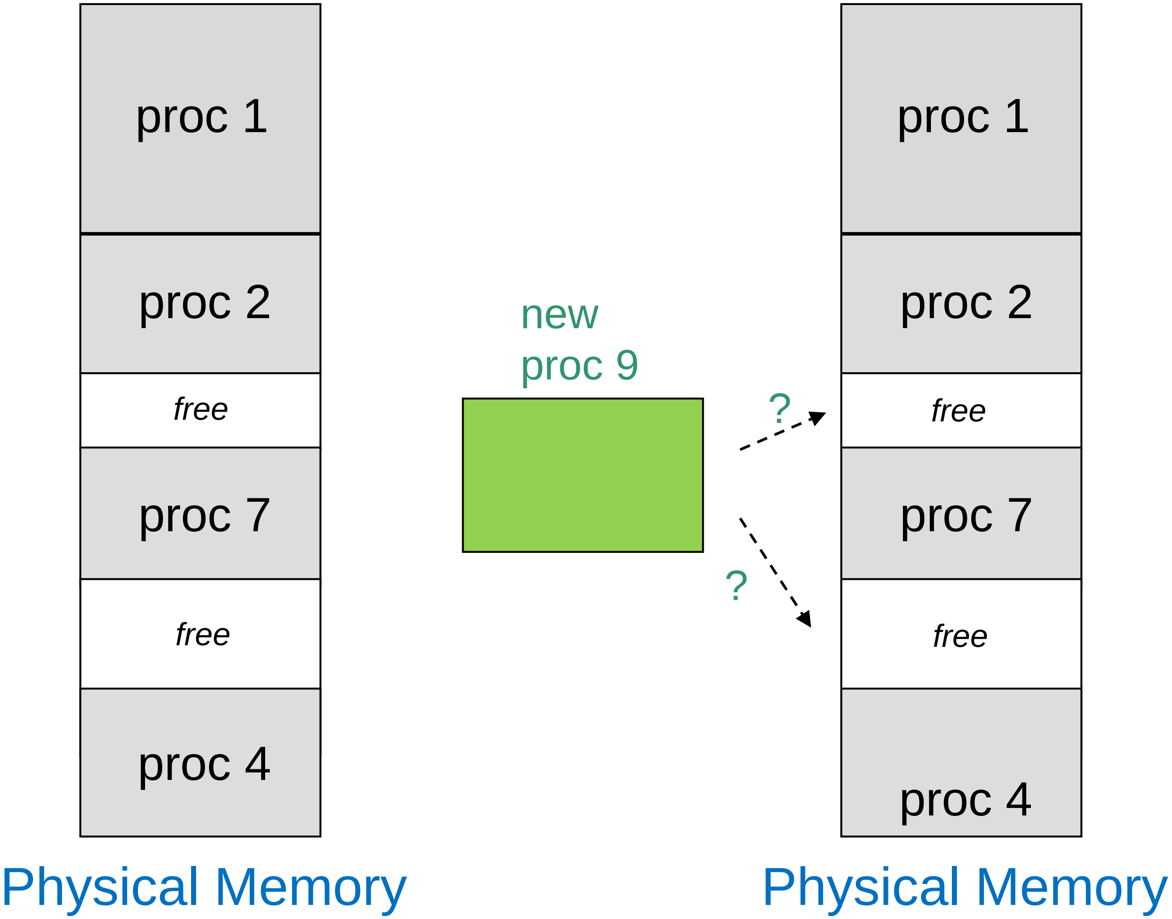
contiguous memory allocation은 External Fragmentation 문제 야기

▪ **Internal Fragmentation** 배정된 메모리공간 내부에서 사용되지 못하고 낭비되는 공간

- Allocated memory may be slightly larger than requested memory
- This size difference is memory internal to a partition, but not being used

→ 프로세스 내부의 문제이므로 운영체제의 메모리 management를 얘기할 때 Internal Fragmentation은 논외로 함

Illustration



fragmentation problem!

Contiguous Memory Allocation Algorithms

- ❒ **How to satisfy a request of size n from a list of free holes?**
- ❒ **Three algorithms**
  - **First-fit: allocate the first hole that is big enough** 속도 빠름
  - **Best-fit: allocate the smallest hole that is big enough** 효율
    - Must search entire list, unless ordered by size
    - Produces the smallest leftover hole

    but
    시간이 갈수록 external fragmentation 상태가 악화될 수 있다
  - **Worst-fit: allocate the largest hole** 속도X 효율X
    - Must also search entire list
    - Produces the largest leftover hole

    long term으로 보면 Worst-fit이 external fragment 현상을 완화시키는 것을 기대해볼수도 있다

- ❒ **First-fit and best-fit are better than worst-fit in terms of speed and storage utilization**

Solutions to Fragmentation

- ❐ Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time

- ❐ Another solution

  OS
  ↓

  - Noncontiguous memory allocation with address translation

Key Idea for Noncontiguous Allocation: Address Translation

프로그램이 사용하는 주소

- ❐ **Programs use logical (virtual) addresses and OS/HW translate them into physical (real) addresses**

실제 하드웨어가 메모리를 접근할때 사용하는 주소

- ❐ **Benefits of address translation**
  - ▪ **Enables noncontiguous memory allocation** 메모리 배정의 유연성 증대
    - Great flexibility for memory allocation
  - ▪ **Further enables efficient implementations for memory protection and sharing**

Logical vs. Physical Address Space

- ❑ The concept of a logical address space that is bound to a separate physical address space is central to proper memory management

- ❑ Logical (virtual) address
  - Generated by the CPU
  - Also referred to as virtual address

- ❑ Physical address
  - Seen by the memory unit 실제 메모리 하드웨어의 주소

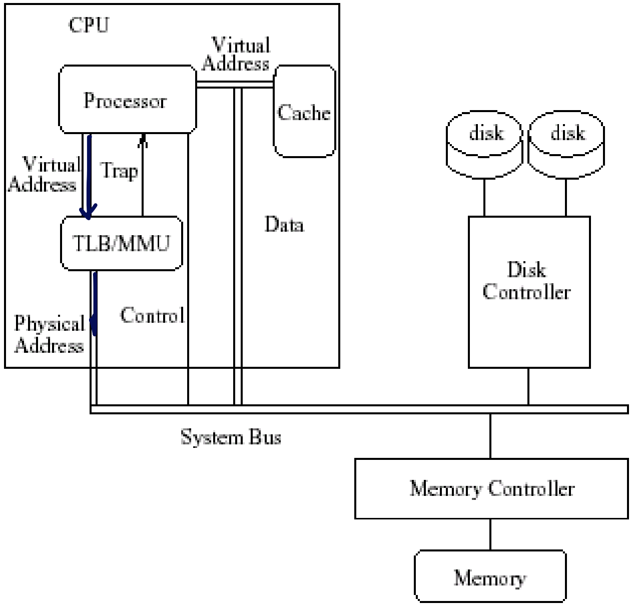
Memory-Management Unit (MMU)

❒ MMU

- Hardware device that maps virtual to physical address

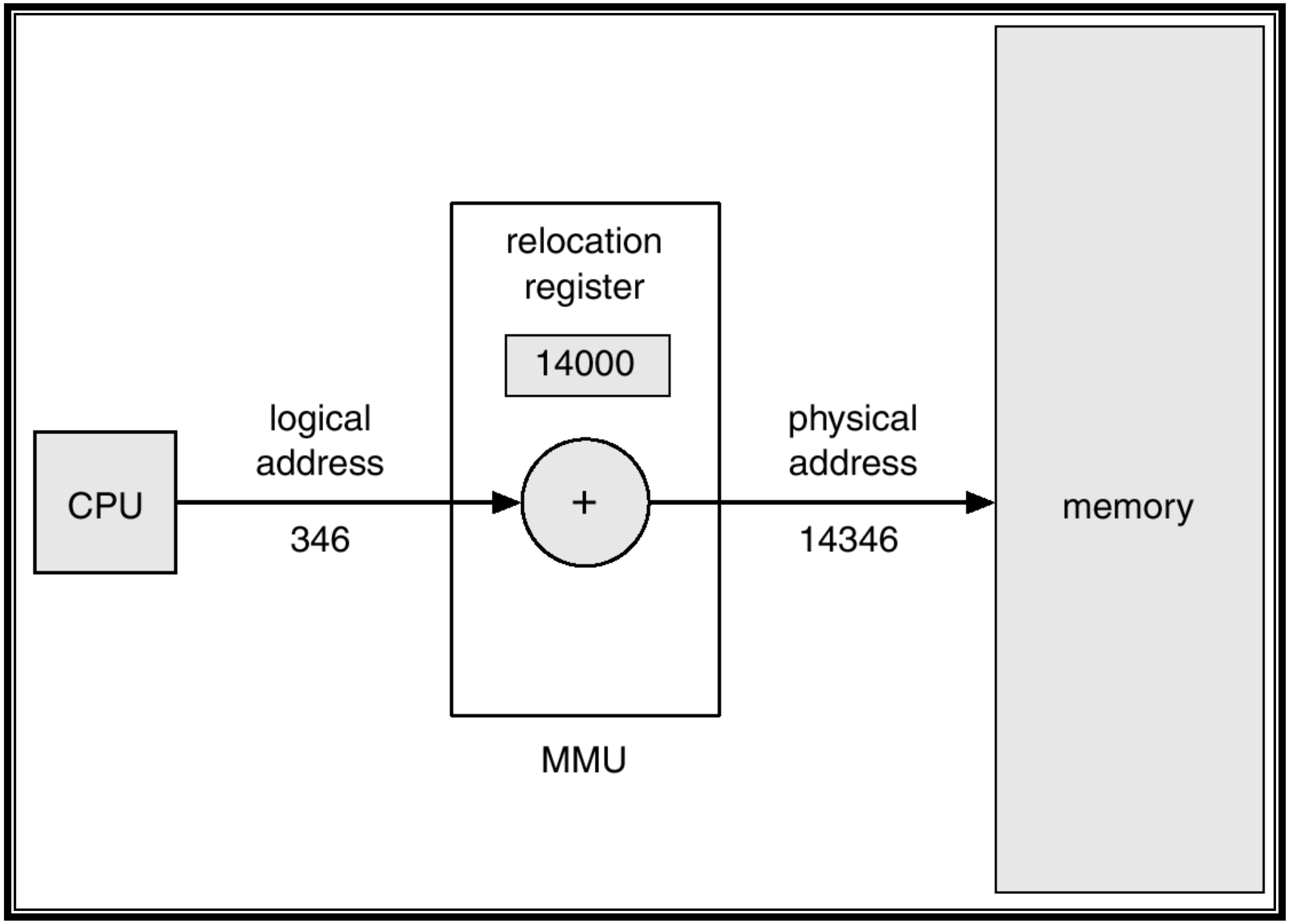
❒ The user program

- Deals with logical addresses
- It never sees the physical addresses



Dynamic Relocation Using a Relocation Register

(MMU 단순화 예제)



Hardware Support for Relocation and Limit Registers

protection

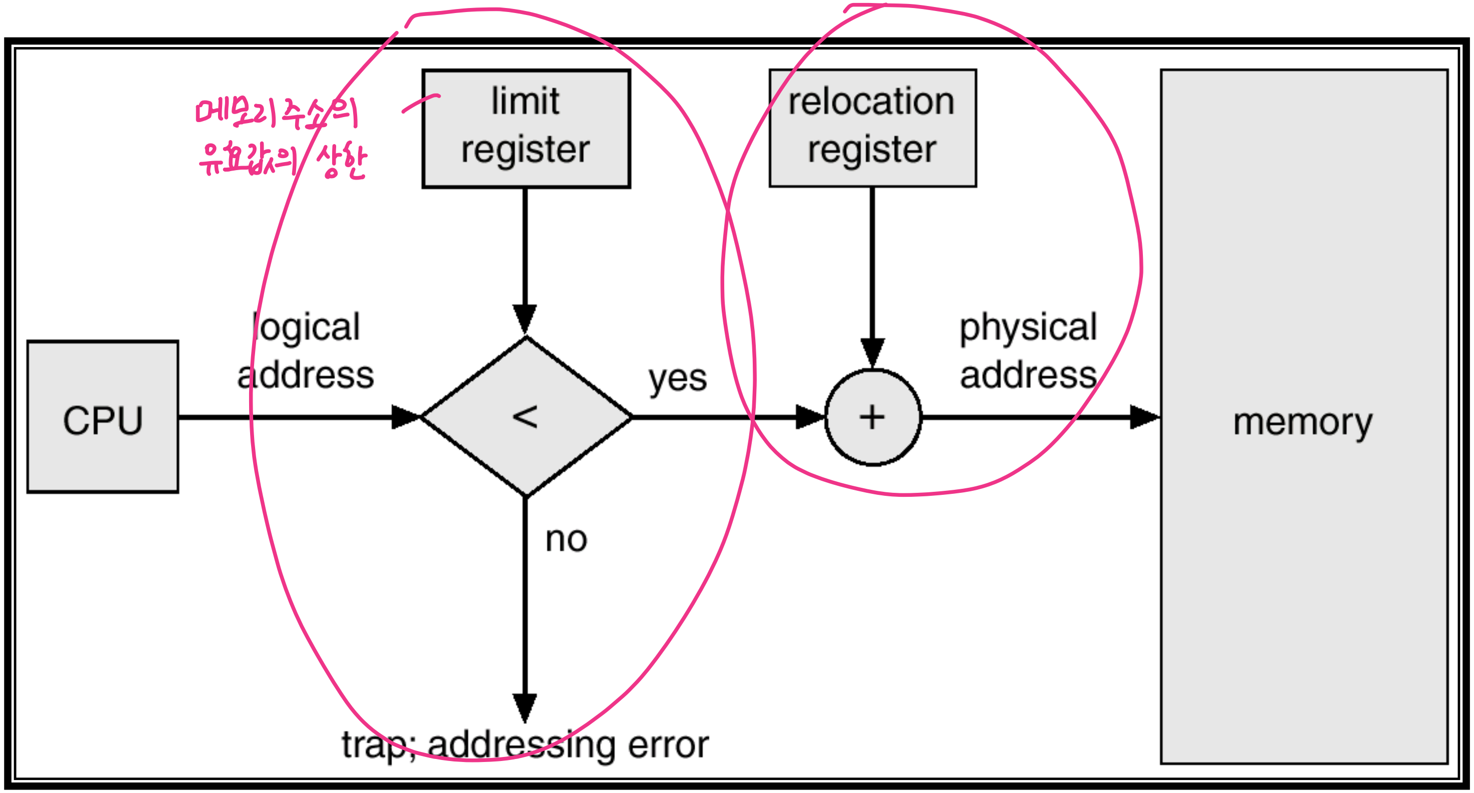
MPU

MMU

(address translation)

전체를 MMU라고 부르기도 함

메모리주소의 유효값의 상한



불법적인 메모리 참조

→ exception

Noncontiguous Memory Allocation with Address Translation

→ 메모리의 비연속적 배정이 가능해짐

❐ Segmentation

세그먼트 단위로 메모리를 배정하는 방법

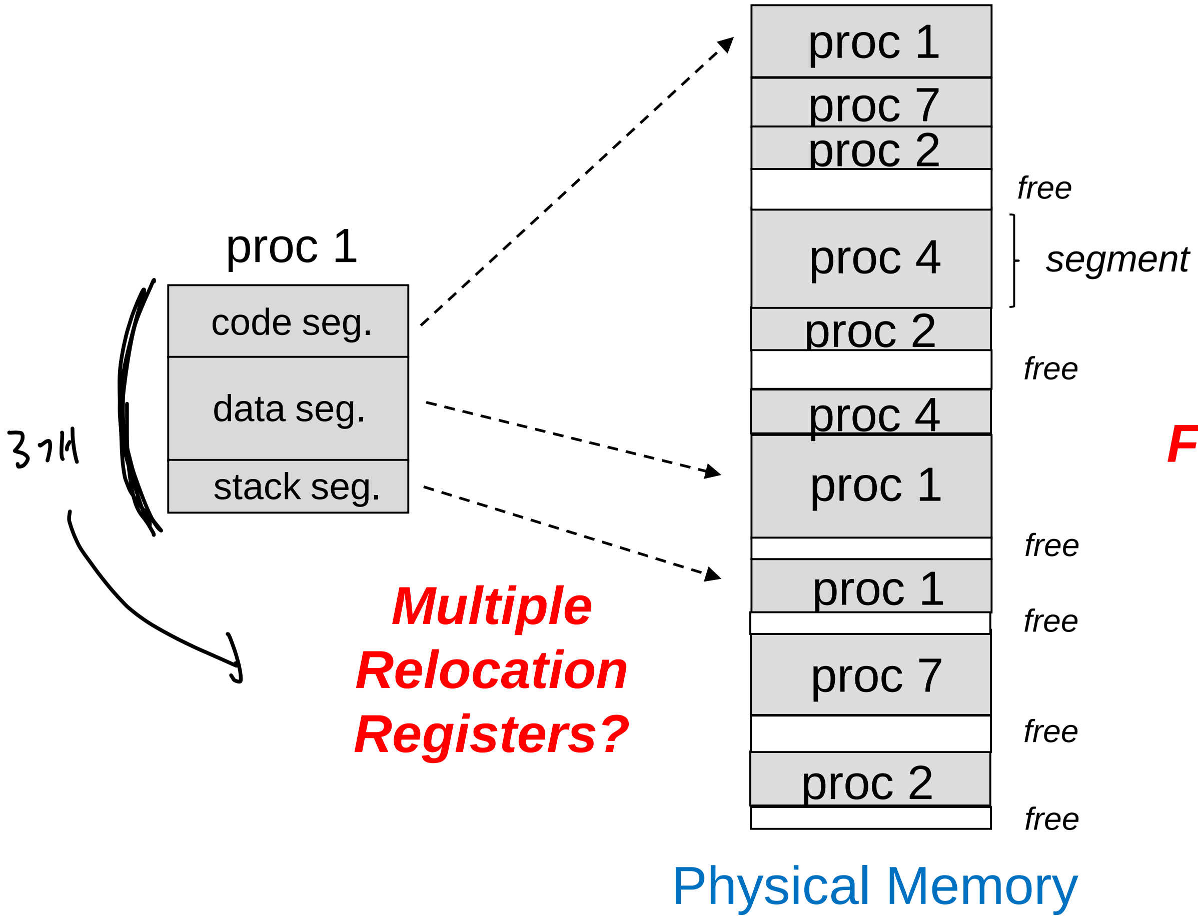
- **Allocate memory on a segment basis**
- **Process memory = code segment + data segment + stack segment + …**heap segment
    - Different segments have different sizes

❐ Paging

동일한 크기의 메모리 영역 단위로 배정함

- **Allocate memory on a page basis**
- **Process memory = page + page + page …**
    - Pages have the same fixed size

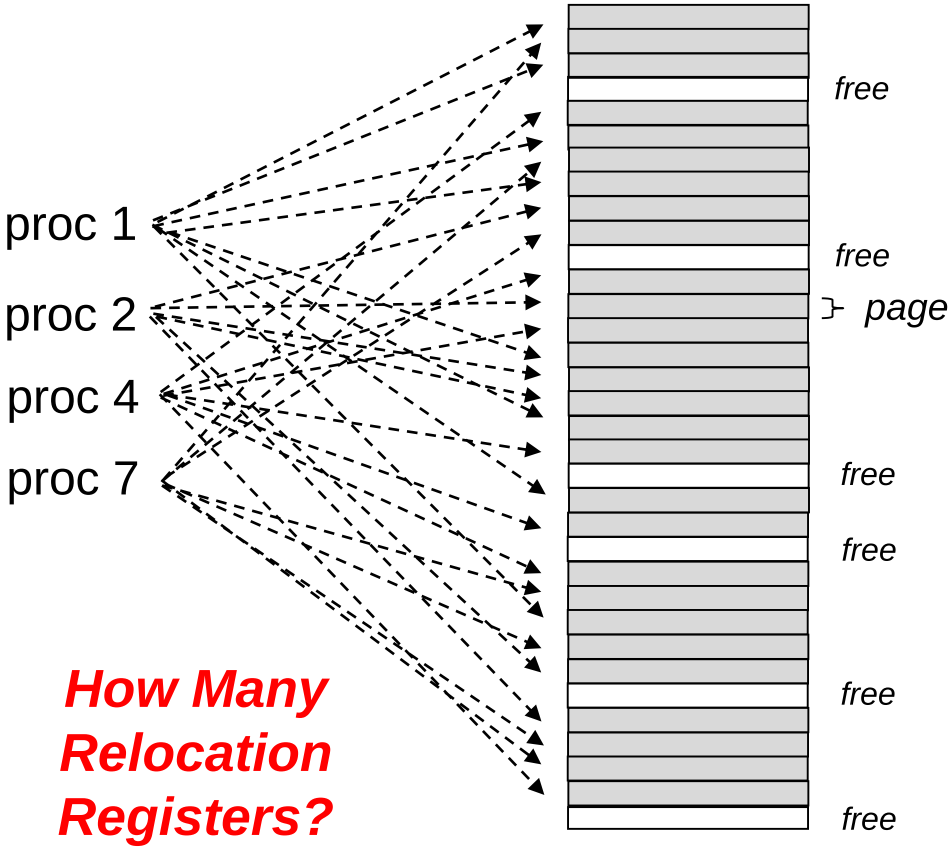
Segmentation



Multiple Relocation Registers?

Reduced Fragmentation!

Paging



Physical Memory

No (External) Fragmentation

How Many Relocation Registers?

각 프로세스마다 배정될수있는 페이지의
최대 개수만큼의 relocation regester 필요

thank you!