# 18 Introduction to Paging

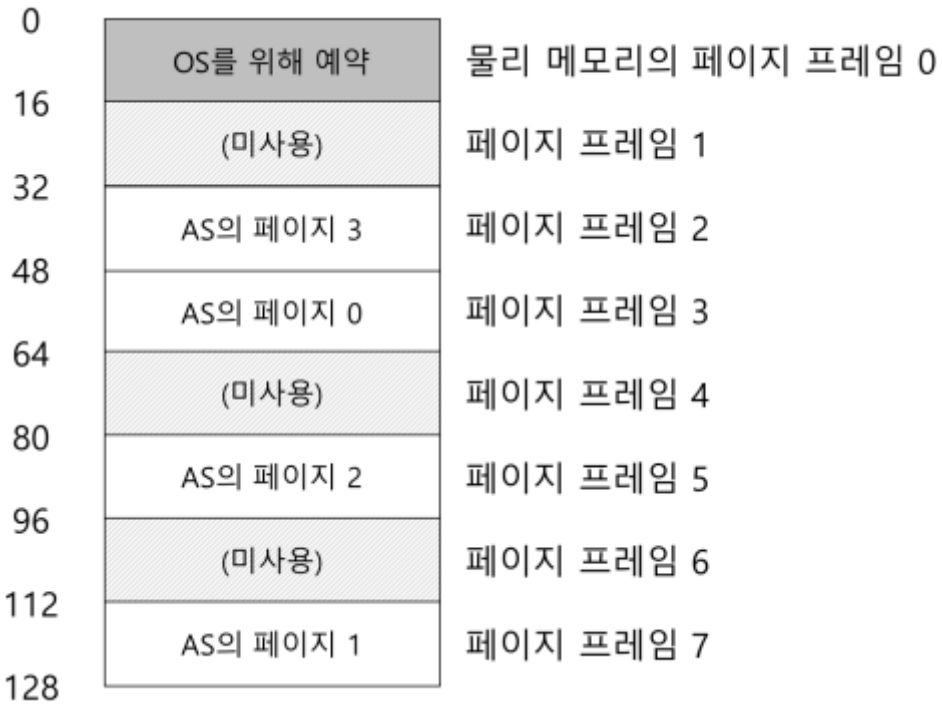
가변 크기의 조각들로 분할하는 것은 태생적으로 할당이 점점 더 어려워지고 단편화를 피하기 어렵다는 문제를 가지고 있다.

이를 해결하기 위한 두 번째 방법, 동일 크기의 조각 분할을 고려해야한다.

고정 크기 단위는 **페이지(page)**라고 부르며, 물리 메모리도 **페이지 프레임(page frame)** 이라고 불리는 고정 크기의 슬롯 배열로 간주한다.

### 예제

총 128 바이트, 8개의 16바이트 페이지



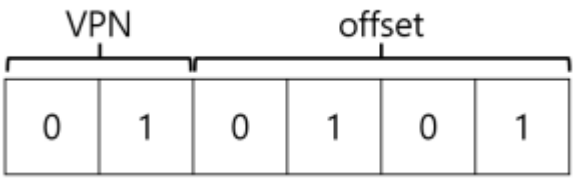

주소 공간의 각 가상 페이지에 대한 물리 메모리 위치 기록을 위해, 운영체제는 프로세스 마다 **페이지 테이블(page table)** 이라는 자료 구조를 유지한다. 위의 예제에서 페이지 테이블은, (vp0->pf3), (vp1->pf7), (vp2->pf5),(vp3->pf2) 이다.

가상 주소는 **가상 페이지 번호(virtual page number, VPN)**와 페이지 내의 **오프셋** 2개의 구성 요소로 분할 된다.\

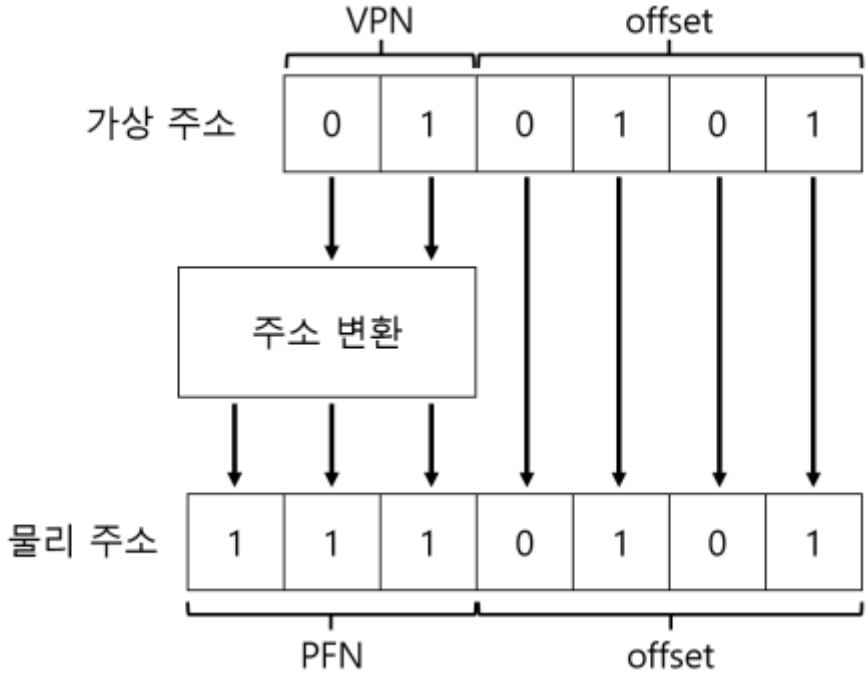
위 예시에선느 가상 페이지가 4개이고, 총 크기가 64 바이트이므로,

2비트는 vpn, 4비트는 오프셋으로 설정된다.

프로세스가 가상 주소 21을 설정하면, 이진 형식으로 **"010101"**로 변환되며 아래 2가지가 된다.




이 주소는 위의 페이지 테이블을 통해 아래와 같이 변환된다.




이런 방식으로 페이징이 작동할 때, 몇가지 의문점이 생긴다.

1. 페이지 테이블은 어디에 저장되며, 그 내용과 크기는 얼마인가?
2. 시스템 속도에 영향을 주지 않을까?

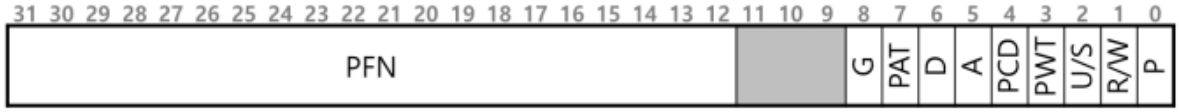
## 페이지 테이블의 저장

4kb 크기의 페이지를 가지는 32bit 주소 공간은 20비트 vpn과 12비트 오프셋으로 구성된다.

따라서 저장해야 하는 변환은 vpn의 가짓수인 2^20만큼 필요하다. 만약 한 **페이지 테이블 항목(page table entry, PTE)**마다 4바이트가 필요하다면, 한 페이지 테이블이 4MB가 필요하게 된다. 따라서 이는 MMU 안에 유지되지 않고 *메모리*에 저장하여야 한다.

## 페이지 테이블의 내용

- Valid bit
  - 프로그램이 실행할때 사용되는 힙, 스탭 이외의 공간은 invalid로 표시하고, 이에 접근하면 운영체제에 트랩을 발생시킨다.
  - 미사용 되는 페이지를 표시함으로써 물리 프레임 할당을 줄이고 메모리를 절약한다.
- protection bit
  - 페이지의 권한과 관련된 정보를 저장한다.
- Present bit
  - **스왑 아웃**되었는지 여부를 가리킨다.
- Reference bit
  - 페이지가 접근되었는지 추적하기 위해 사용된다.
  - 페이지가 인기가 있는지 결정한다.




〈그림 21.5〉 **x86 페이지 테이블 항목(PTE)**

## 페이징 속도

페이지 테이블의 크기가 커지면서 처리 속도가 저하될 수 있다.