Ch 9.4

9.4 Structure of the Page Table

9.4.1 Hierarchical Paging

- page table이 계층 구조를 가지는 것으로, **page table 자체를 paging해서 저장해놓는 방법**이다. 기존에는 logical address가 page number(p)와 page offset(d)으로만 구성되어있었다면, 여기서는 **page number를 다시 page number(p1)와 page offset(p2) 두 개의 파트**로 나뉜다.

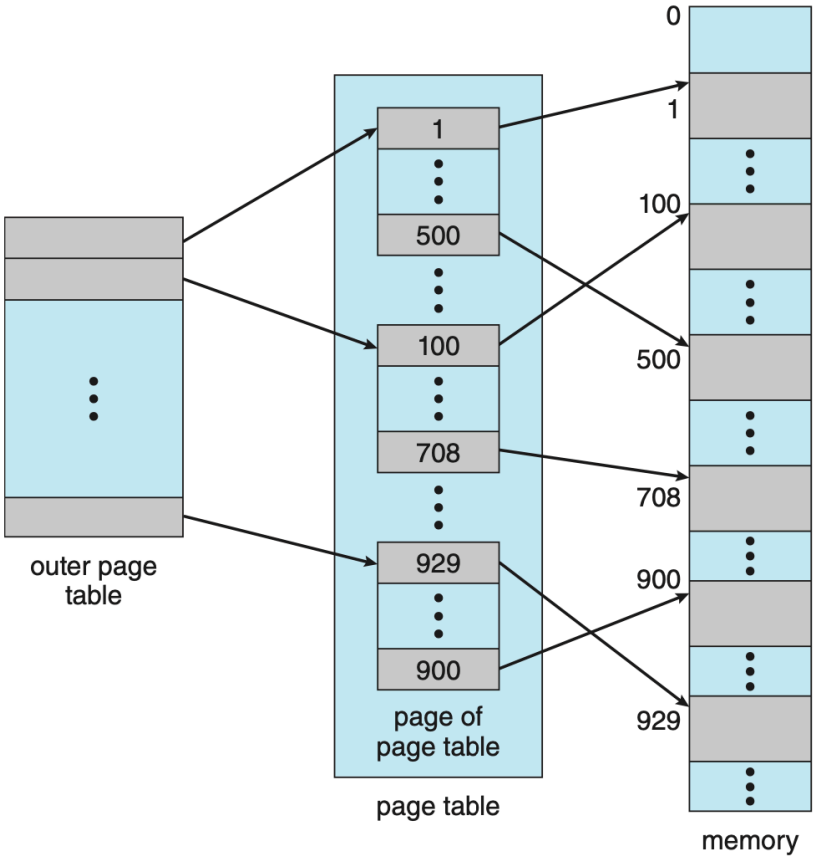
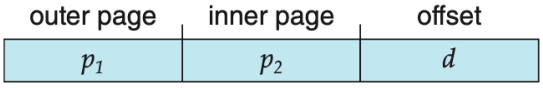


Figure 9.15 A two-level page-table scheme.



- 위의 사진은 two-level page table 구조를 나타낸 것으로 page table이 outer page table과 inner page table 2개가 존재하지만, **필요에 따라서는 outer page table를 다시 paging해서 three-level page table**를 만들 수 있으며, 필요한 만큼 paging을 해서 multi-level page table를 만들 수 있다.
- 따라서 virtual address를 physical address로 매핑하려면 우선 p1을 보고 outer page table에서 우리가 찾으려는 inner page table의 위치를 찾은 후, p2을 보고 inner page table에서 우리가 찾고자 하는 frame number를 찾은 후, d를 보고 해당 frame 안에서 offset만큼 더해서 physical address를 찾는다.

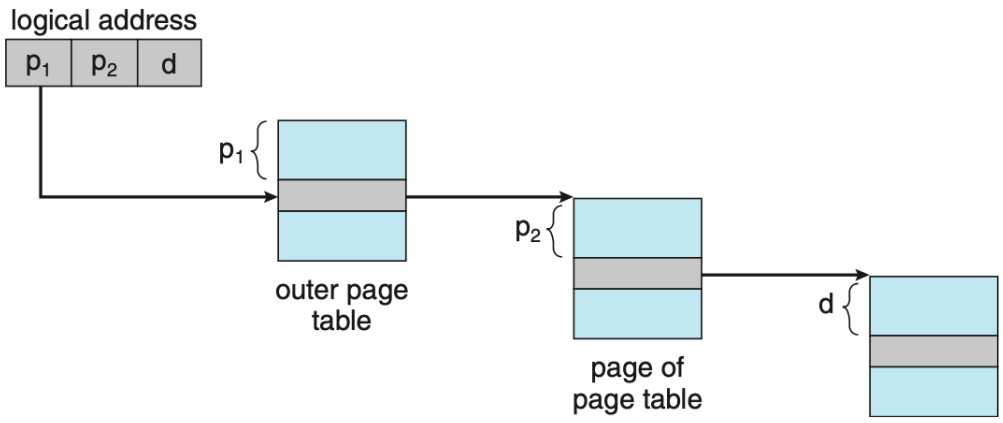


Figure 9.16 Address translation for a two-level 32-bit paging architecture.

9.4.2 Hashed Page Tables

- hashed page table를 두는 방법으로, **hash table의 각 엔트리는 linked list로 이루어져** 있으며, linked list의 각 element는 1) virtual page number, 2) mapped page frame의 number, 3) linked list에서 다음 순서의 element를 가리키는 pointer로 이루어져있다.
- 따라서 physical address를 찾으려면 우선 page number(p)를 hashing해서 hash table에서 해당하는 엔트리를 찾은 후, 해당 엔트리의 linked list를 따라가면서 page number가 일치하는 element를 찾는다. 일치하는 element를 찾으면 해당 element의 2번째 값인 mapped page frame number를 가지고 logical address에 있던 page offset(d)과 결합하여 physical address를 찾는다.

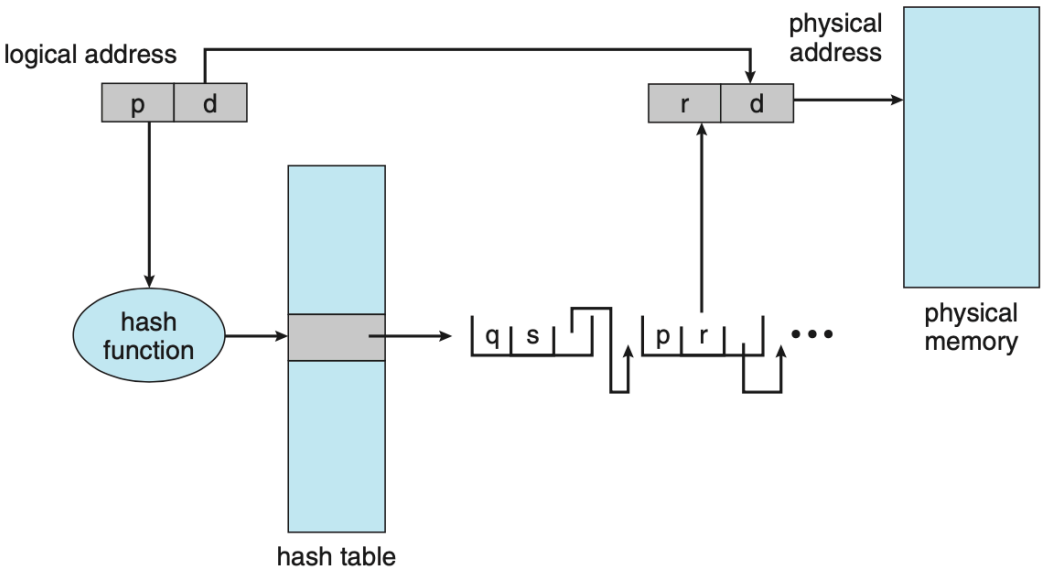


Figure 9.17 Hashed page table.

- 이 구조의 variation으로 **clustered page table**이 존재하는데, 이 경우에는 hash table의 각 엔트리가 여러 개의 페이지를 가리킨다.

9.4.3 Inverted Page Tables

- 프로세스별 page table이 하나씩 따로 있었던 것을 하나로 합쳐서 **시스템 내에 단 하나의 page table만이 존재하게 하는 방법**으로, 어떤 프로세스에 해당하는 메모리인지 알기 위해 logical address와 page table의 각 엔트리는 **process id(pid) 정보**를 가지고 있다.
- 따라서 virtual address를 physical address로 매핑하려면 우선 pid와 page number(p)를 가지고 hash table에서 해당하는 엔트리를 찾는다. 그 엔트리의 index 값인 i는 physical address를 찾는데에 있어 page number가 되며, 따라서 page offset(d) 정보와 결합해서 physical address를 찾을 수 있게 된다.

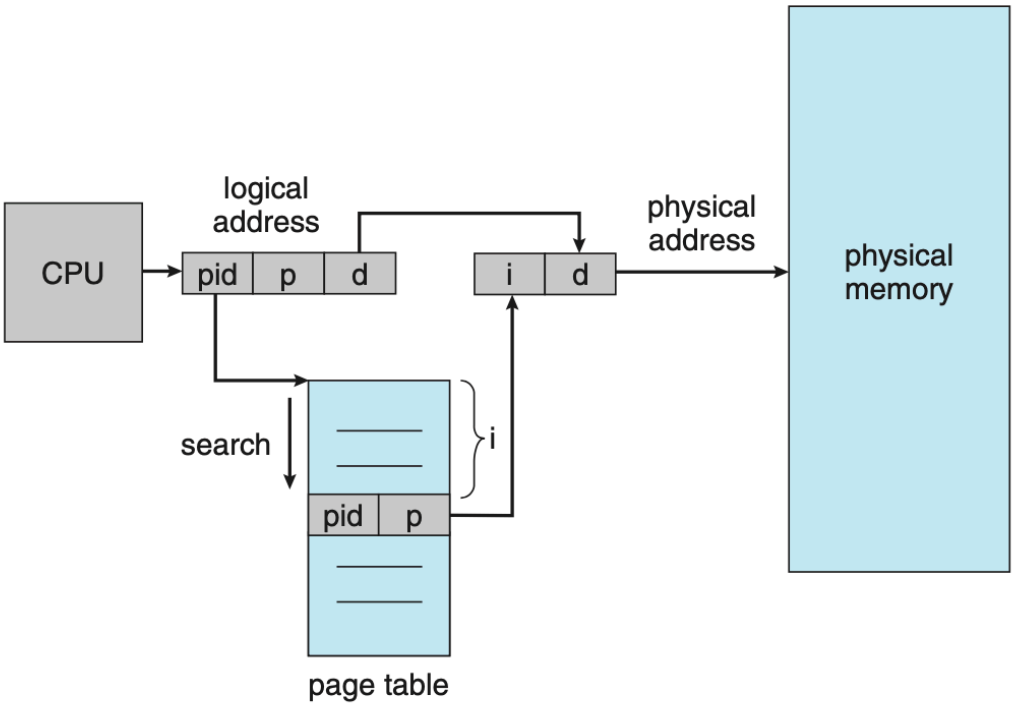


Figure 9.18 Inverted page table.

- 하나의 page table만이 존재하기에 전체적인 page table 사이즈가 작아서 좋지만, pid를 탐색해야되는 시간이 추가된다는 비효율적인 측면도 있다. 이를 해결하기 위해 hashing을 사용하기도 한다.

참고영상

- two-level page table
- multilevel paging and Inverted page table