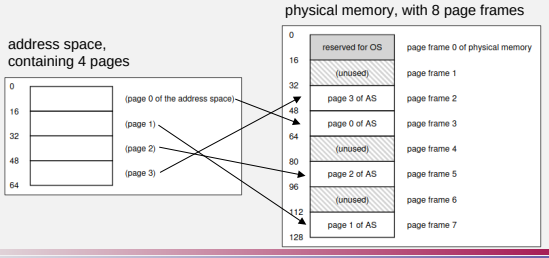
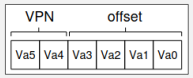
Paging

- dividing memory into variable-sized pieces leads to problems with fragmentation
- instead, divide memory into fixed-sized *pages*
  - view physical memory as an array of fixed-sized *page frames*
    - each page frame can contain one virtual page



Address Translation With Paging

- OS maintains per-process *page tables*
  - tracks where each page of the address space is in memory
- address translation
  - performed by the MMU using the page table
  - virtual address is split into a *virtual page number* (VPN) and an *offset* within the page



assume 16-byte page size and 64 byte address space
virtual address needs 6 bits ($2^6 = 64$) total to cover the address space and 4 bits ($2^4 = 16$) for the offset with the page

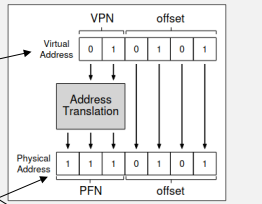
  - page table maps VPN to *physical frame number* (PFN)

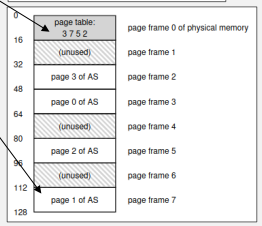
    → given virtual address, map VPN to PFN and add offset to get the physical address

Address Translation Example

```
movl 21, %eax
```

- load from virtual address 21
  - 21 is 010101 in binary
    - top two bits are the VPN (virtual page number) – 01
  - look up entry 01 in the page table
    - PFN (physical frame number) is 7 (111)
  - replace VPN with PFN to get the translated address
    - translated address is 1110101 (117)
    - no change to offset because virtual pages and page frames are the same size





physical memory