Virtual Memory Overview

In a virtual memory system, the virtual address space for a process is viewed as a series of fixed-size *pages.* To be referenced by a read or write operation during execution, the desired page must be resident in a *page frame* in RAM (one page and one frame contain the same number of bytes).

The operating system, with support from the MMU (memory management unit), translates each virtual address to a physical address. A high-level view:

If the referenced page is not currently resident in RAM, copy it from disk to RAM
Translate the virtual address to the physical address and send the memory reference to the cache

A virtual address is viewed as two fields: a page number and a byte offset within that page. In a similar way, a physical address is viewed as two fields: a frame number and a byte offset within that frame. The size of the byte offset in both a virtual address and a physical address must be the same and is determined by the size of one page.

The MMU is the hardware unit which is responsible for translating a virtual address to a physical address. Every memory reference (virtual address and R/W operation) is sent to the MMU, which converts the virtual address to a physical address and then forwards the updated memory reference (physical address and R/W operation) to the cache controller.

If the MMU is unable to map the virtual address to a physical address, it generates an interrupt and the operating system handles the exceptional situation.

The page table is a data structure which maintains the set of mappings from virtual page numbers to physical frame numbers. Each PTE (page table entry) contains the following information:

V bit – equals 1 when the page is a valid page within the process address space
P bit – equals 1 when the page is currently present (resident) in RAM
R bit – equals 1 when the page has been referenced while present in RAM
M bit – equals 1 when the page has been modified while present in RAM
Frame – the physical frame number which corresponds to this virtual page number

In some virtual memory systems, additional control bits are contained in each PTE to regulate access to pages (some pages can be marked as "no access" or "read only", for example).

The TLB (translation lookaside buffer) is a specialized cache unit which holds a subset of the PTEs (those PTEs which are currently active). When attempting to map a virtual page number to a physical frame number, the MMU checks the TLB first; if the desired PTE is not found in the TLB, the MMU accesses the page table.

If the PTE indicates that the desired page is not present in RAM, the MMU generates an interrupt ("page fault") to signal to the operating system that the desired page must be copied from disk to RAM. If there is no empty frame available to receive the incoming page, the operating system must remove some other page in order to make an empty frame available ("page replacement").

The following outline describes the steps used by the MMU to process a read or write request:

```
decompose the virtual address into fields:  page number and page offset
use the page number to access the correct PTE in the TLB or the page table
if (PTE.V == 0) then
    generate interrupt (fatal error – page number is not valid)
else
    if (PTE.V ==1 and PTE.P == 0) then
        generate interrupt (page fault processing – page must be fetched)
    endif
    if (read operation) then
        PTE.R ← 1
    else if (write operation) then
        PTE.R ← 1
        PTE.M ← 1
    endif
    compose the physical address:  PTE.Frame and page offset
    send the memory reference (physical address and operation) to the cache controller
endif
```

The following outline describes the steps used by the operating system to process a page fault:

```
if (the free frame list is empty) then
    select a page which is currently resident in RAM as the victim
    if (M == 1 in the victim's PTE) then
        copy the victim page from RAM to disk
    endif
    update the victim's PTE:  P ← 0
    move the victim's frame to the free frame list
endif
allocate a frame from the free frame list to hold the desired page
initialize the desired page's PTE:  P ← 1, R ← 0, M ← 0, Frame ← frame number
resume execution of the interrupted process
```