**CMPSCI 577 Operating Systems Design and Implementation** **Spring 2020**

Lecture 10: 2/25/2020 Memory Management Refresher

*Lecturer:* ***Prashant Shenoy*** *Scribe:* ***Qifan Yang (Master)***

## 10.1 Background:Computer Architecture

When a process starts, an executable and data will be fetched from the disk by the OS and loaded to the Main Memory (RAM). Where to put these data and files and how to put them into RAM is the context of Memory Management.

## 10.2 Contiguous Memory Allocation

**Definition:** The process data are placed in the RAM contiguously as a whole.

**Definition:Segment** A chunk of memory assigned to a process.

**Definition:Physical Address** A real address in RAM

**Definition:Virtual Address** An address relative to the start of a process's address space.

**Uniprogramming OS** In this kind of OS there is only one process in the RAM any time and the process address always start from 0 so the virtual address is equal to the physical address.

**Why do we have to have virtual address in Multiprogramming OS** When the program compiles it cannot know where it will be put in the memory when the OS loads it. So virtual address is needed and can be generated when compiling and when the program is loaded or in the runtime, virtual memory can be transferred to physical memory via a process called relocation.

**Static Relocation** When loading, the OS adjusts the virtual address to physical address once for all. Everytime a process is moved the OS has to do the transformation again.

**Dynamic Relocation** In runtime, the process starting address and ending address is loaded into registers. Sum of The virtual address and the process starting address gives us the physical address. The sum is compared to the ending address to prevent memory corruption. This transformation is done by hardware so it is as quick as it can.

**Memory Allocation** OS keeps a table of holes in the RAM which are available and when a process starts it has to scan the table and pick a hole for the process via different policies. But this will bring about External Fragmentation.

**External Fragmentation** As process goes out of and enters into RAM, the RAM will be cut into small pieces and not one of them big enough for a new process. This can be solved by Compaction.

**Compaction** OS actively gather all the process together space-wise and create a bigger and contiguous space for new processes. This comes with a cost and can be done using different policies (on demand/periodically)

## 10.3 Paging

**Definition:** Divide RAM into equally sized frames and divide programs into equally sized pages. Then each page can be loaded into any frames. The scheme is aimed at eliminate external fragmentation.

**Page table** Every process will keep a page table recording the mapping relationship of the pages within the process and the frames in the RAM.

**Relocation Process** When a process is going to be executed, its page table address is loaded to a register. Whenever the relocation is needed, the virtual address is transferred into a page num and a offset using mod(or binary operations if the page size is power of 2). Then the OS find the page table address and access it to transfer the page num to the starting address and ending address of the actual frame. The sum of the frame starting address and offset is the physical address which will be compared with the frame endting address to ensure safety.

**Translation Look-aside Buffer** Cache for the page table entries.Store the recently used entry in the page table and when OS is trying to access the page table it would first check the TLB for a hit.

## 10.4 Segmentation

**Definition:** Programs are cut into different segments according to user interpretation. And different segments are allocated memory independently. It is good for Memory sharing.

**Segment table** Each entry contains a base address in memory, length of segment, and protection information (can this segment be shared, read, modified, etc.). To speed up the transformation of the addresses, this can be loaded into registers.

**Relocation process** Virtual addresses is a tuple of segment num and offset. OS use the segment num and the segment table to find out a base address and add it up with the offset to compute the physical address.

**Segment and paging** Each segment is divided to fixed size pages and the memory is divided to equally sized frames. Then each page can be loaded into any frames. For each segment there is a page table and the segment table keeps the address of the page table. When relocating, the page table address is first accessed (in segment table stored in registers), then the OS accesses the page table to find the frame starting address and compute the physical address.

**Memory sharing** Sometimes data can be shared between different process like the standard c library. The way this is done is that we change to segment table of the process to make the segment point to the shared segment.