Embedded Operating Systems

Agenda

- Memory Management
  - Contiguous allocation
  - Segmentation
  - Paging

Contiguous Allocation

Fixed Partition

- RAM is divided into fixed sized partitions.
- This method is easy to implement.
- Number of processes are limited to number of partitions.
- Size of process is limited to size of partition.
- If process is not utilizing entire partition allocated to it, the remaining memory is wasted. This is called as "internal fragmentation".

Dynamic/Variable Partition

- Memory is allocated to each process as per its availability in the RAM. After allocation and deallocation of few processes, RAM will have few used slots and few free slots.
- OS keep track of free slots in form of a table.
- For any new process, OS use one of the following mechanism to allocate the free slot.
  - First Fit: Allocate first free slot which can accommodate the process.
  - Best Fit: Allocate that free slot to the process in which minimum free space will remain.
  - Worst Fit: Allocate that free slot to the process in whic maximum free space will remain.
- Statistically it is proven that First fit is faster algo; while best fit provides better memory utilization.
- Memory info (physical base address and size) of each process is stored in its PCB and will be loaded into MMU registers (base & limit) during context switch.

- CPU request virtual address (address of the process) and is converted into physical address by MMU as shown in diag.
- If invalid virtual address is requested by the CPU, process will be terminated.
- If amount of memory required for a process is available but not contiguous, then it is called as "external fragmentation".
- To resolve this problem, processes in memory can be shifted/moved so that max contiguous free space will be available. This is called as "compaction".

Virtual Memory

- The portion of the hard disk which is used by OS as an extension of RAM, is called as "virtual memory".
- If sufficient RAM is not available to execute a new program or grow existing process, then some of the inactive process is shifted from main memory (RAM), so that new program can execute in RAM (or existing process can grow). It is also called as "swap area" or "swap space".
- Shifting a process from RAM to swap area is called as "swap out" and shifting a process from swap to RAM is called as "swap in".
- In few OS, swap area is created in form of a partition. E.g. UNIX, Linux, ...
- In few OS, swap area is created in form of a file E.g. Windows (pagefile.sys), ...
- Virtual memory advantages:
  - Can execute more number of programs.
  - Can execute bigger sized programs.

Segmentation

- Instead of allocating contiguous memory for the whole process, contiguous memory for each segment can be allocated. This scheme is known as "segmentation".
- Since process does not need contiguous memory for entire process, external fragmentation will be reduced.
- In this scheme, PCB is associated with a segment table which contains base and limit (size) of each segment of the process.
- During context switch these values will be loaded into MMU segment table.
- CPU request virtual address in form of segment address and offset address.
- Based on segment address appripriate base-limit pair from MMU is used to calculate physical address as shown in diag.
- MMU also contains STBR register which contains address of current process's segment table in the RAM.

Demand Segmentation

- If virtual memory concept is used along with segmentation scheme, in case low memory, OS may swap out a segment of inactive process.
- When that process again start executing and ask for same segment (swapped out), the segment will be loaded back in the RAM. This is called as "demand segmentation".

- Each entry of the segment table contains base & limit of a segment. It also contains additional bits like segment permissions, valid bit, dirty bit, etc.
- If segment is present in main memory, its entry in seg table is said to be valid (v=1). If segment is swapped out/on disk, its entry in segment table is said to be invalid (v=0).

Paging

- RAM is divided into small equal sized partitions called as "frames" / "physical pages".
- Process is divided into small equal sized parts called as "pages" or "logical/virtual pages".
- page size = frame size.
- One page is allocated to one empty frame.
- OS keep track of free frames in form of a linked list.
- Each PCB is associated with a table storing mapping of page address to frame address. This table is called as "page table".
- During context switch this table is loaded into MMU *.
- CPU requests a virtual address in form of page address and offset address. It will be converted into physical address as shown in diag.
- MMU also contains a PTBR, which keeps address of page table in RAM.
- If a page is not utilizing entire frame allocated to it (i.e. page contents are less than frame size), then it is called as "internal fragmentation".
- Frame size can be configured in the hardware. It can be 1KB, 4KB, 64KB, ...
- Typical Linux and Windows OS use page size = 4KB.

Demand Paging

- If virtual memory concept is used along with paging scheme, in case low memory, OS may swap out a page (or few pages) of inactive process.
- When that process again start executing and ask for swapped out page, the page will be loaded back in the RAM. This is called as "demand paging".
- Each entry of the page table contains frame address. It also contains additional bits like page permissions, valid bit, dirty bit, etc.
- If page is present in main memory, its entry in page table is said to be valid (v=1). If page is not present in main memory (may be on disk), its entry in page table is said to be invalid (v=0).

Page table entry

- Each PTE is of 32-bit (on x86 arch) and it contains
  - Frame address
  - Permissions (read or write)

- Validity bit
- Dirty bit
- ...

Two Level Paging

- Primary page table has number of entries and each entry point to the secondary page table page.
- Secondary page table has number of entries and each entry point to the frame allocated for the process.
- Virtual address requested by a process is 32 bits including
  - p1 (10 bits) -> Primary page table index/addr
  - p2 (10 bits) -> Secondary page table index/addr
  - d (12 bits) -> Frame offset
- If frame size is 4KB, 12 bits are sufficient to speficy any offset in the frame. This will also ensure that "d" will not contain any invalid frame offset.
- If virtual address of a process is of 32 bits [p1|p2|d], then maximum address of the process can be 1024 * 1024 * 4 * 1024 = 4 GB.

TLB (Translation Look-Aside Buffer) Cache

- TLB is high-speed associative cache memory used for address translation in paging MMU.
- TLB has limited entries (e.g. in P6 arch TLB has 32 entries) storing recently translated page address and frame address mappings.
- The page address given by CPU, will be compared at once with all the entries in TLB and corresponding frame address is found.
- If frame address is found (TLB hit), then it is used to calculate actual physical address in RAM (as shown in diag).
- If frame address is not found (TLB miss), then PTBR is used to access actual page table of the process in the RAM (associated with PCB). Then page-frame address mapping is copied into TLB and thus physical address is calculated.
- If CPU requests for the same page again, its address will be found in the TLB and translation will be faster.
- If TLB is full, older entry will be overwritten (LRU). It ensures that TLB always contains recent entries only.
- Context switch can be handled in one of the following ways:
  - During context switch TLB is flushed i.e. all entries used by previous process will be cleared. Thus new process's page addresses will not conflict with prev process's page addresses.
  - TLB can store page addresses along with PID of the process. In this case there is no need to flush TLB during context switch. The comparison of the page address will not conflict with another process, because it is associated with PID of the process. This combination PID and Page Address is called as "ASID".