# Embedded Operating Systems

## Segmentation

- Refer yesterday's notes.

## 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.
- If whole frame is not utilized by the page, then remaining space (in that frame) is wasted. This is internal fragmentation.
- Bigger the page size, more is the internal fragmentation. But it reduces number of pages and hence number of page table entries.
- Smaller the page size, less is the internal fragmentation. But it increases number of pages and hence number of page table entries.

#### 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".

## Segmented Paging

- In OS, processes are first divided into segments i.e. text, data, heap, stack, etc.
- In paging, these segments are further divided into pages.