Demand Paging

PA 3

Intel System Programming

- Outer Page Table(Page Directory) = 1024 page directory entries in a page directory

  Page Table = 1024 page table entries in a page table

  Page - 4-KB flat address space

- PDBR = Page Directory Base Register (CR3)

  points to the start address of Page Directory(Outer Page Table)

- TLB - lookup in page tables in memory are performed only when the TLBs do not contain the translation information for a requested page.

  invalidate - automatically invalidated any time the CR3 register is loaded.

(Read Intel System Programming API - Chapter 3 (Paging), Chapter 5 (Page Fault interrupt)

From Boot

1. Initialize (zero out the values)
    - backing store - (create data structures)
    - frames - (create data structures)
    - install page fault handler
2. Create new page table for null process:
    - create page directory (outer page table)
    - initialize 1:1 mapping for the first 4096 pages
        - allocate 4 page tables (4x1024 pages)
        - assign each page table entry to the address starting from page number 0 to 1023
    - this page tables should be shared between processes

From Boot -2

3. Enable paging
   - set bit 31st of the CR0 register
   - take care that PDBR is set, because subsequent memory address access will be virtual memory addresses

4. Creating new process (eg. main):
   - create page directory (same as with null process)
   - share the first 4096 pages with null process

5. Context switch:
   - every process has separate page directory
   - before ctxsw() load CR3 with the process's PDBR

Using Virtual Memory

1. Allocate pages in backing store
2. Map it to virtual page using xmmap()
   - for example if you do xmmap(A, backingstore, 10)
   - then the mapping would be made to consecutive locations in backingstore for
     virtual pages: A, A+1, A+2, ..., A+9
3. Then try accessing the virtual address
4. If the page is not present a Page Fault is generated:

Page Fault

1. Address that caused page fault
   - content of CR2 register

2. Search for the page table entry. Two cases:
   - a). second level page table does not exist
   - b). second level page table exists but the page table entry does not exist
   - How do we know? Use the P flag for page directory/table entry

Page Fault - 2

3. Case a)
   - allocate a frame -> initialize (zero out the page table frame)
   - update the page directory entry with base address of the page table frame
   - Now this case becomes Case (b)

Page Fault - 3

4. Case b)
   - Locate backing store id of the faulted page, the page number in the backing store.
   - Find a free frame to store the page from backing store
     - if found: use the free frame
       if not found: evict a page frame (Page Replacement Algorithm)
   - Update the page table entry for the page and possibly for evicted page frame
5. Finally: Flush TLB content, by reloading CR3 with page directory address

Virtual address has page table offset as well as page directory offset.
PageTableNumber(31-22) PageNumber(21-12) Offset(11-0)

Page Directory/Table Entry Format

31-12	PFA	page frame address
11-9	Avail	available to OS
8	0	must be 0
7	L	PTE -- Must be 0. Dir Entry -- 4MB page
6	D	dirty (PTE only -- documented as undefined in directory entry)
5	A	accessed
4	PCD	page cache disable (can't cache data on this page)
3	PWT	page write transparent (tell external cache to use write-through strategy for this page)
2	U	user accessible
1	W	writeable
0	P	present