Translation Lookaside Buffer (TLB) in Paging

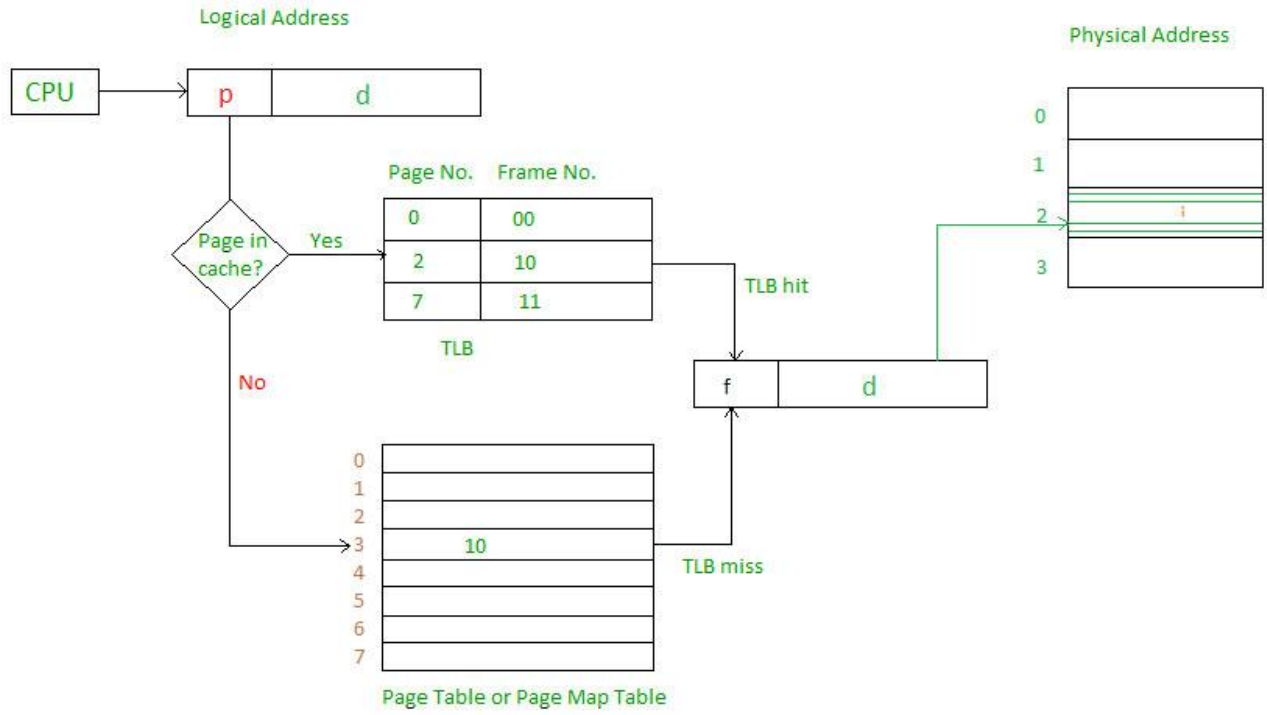
In Operating System (Memory Management Technique : Paging), for each process page table will be created, which will contain Page Table Entry (PTE). This PTE will contain information like frame number (The address of main memory where we want to refer), and some other useful bits (e.g., valid/invalid bit, dirty bit, protection bit etc). This page table entry (PTE) will tell where in the main memory the actual page is residing.
Now the question is where to place the page table, such that overall access time (or reference time) will be less.

The problem initially was to fast access the main memory content based on address generated by CPU (i.e logical/virtual address). Initially, some people thought of using registers to store page table, as they are high-speed memory so access time will be less.
The idea used here is, place the page table entries in registers, for each request generated from CPU (virtual address), it will be matched to the appropriate page number of the page table, which will now tell where in the main memory that corresponding page resides. Everything seems right here, but the problem is register size is small (in practical, it can accommodate maximum of 0.5k to 1k page table entries) and process size may be big hence the required page table will also be big (lets say this page table contains 1M entries), so registers may not hold all the PTE's of Page table. So this is not a practical approach.

To overcome this size issue, the entire page table was kept in main memory. but the problem here is two main memory references are required:

1. To find the frame number
2. To go to the address specified by frame number

To overcome this problem a high-speed cache is set up for page table entries called a Translation Lookaside Buffer (TLB). Translation Lookaside Buffer (TLB) is nothing but a special cache used to keep track of recently used transactions. TLB contains page table entries that have been most recently used. Given a virtual address, the processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If a page table entry is not found in the TLB (TLB miss), the page number is used as index while processing page table. TLB first checks if the page is already in main memory, if not in main memory a page fault is issued then the TLB is updated to include the new page entry.



Steps in TLB hit:

1. CPU generates virtual (logical) address.
2. It is checked in TLB (present).
3. Corresponding frame number is retrieved, which now tells where the main memory page lies.

Steps in TLB miss:

CPU generates virtual (logical) address.

1. It is checked in TLB (not present).
2. Now the page number is matched to page table residing in main memory (assuming page table contains all PTE).
3. Corresponding frame number is retrieved, which now tells where the main memory page lies.
4. The TLB is updated with new PTE (if space is not there, one of the replacement technique comes into picture i.e either FIFO, LRU or MFU etc).

Effective memory access time(EMAT) : TLB is used to reduce effective memory access time as it is a high speed associative cache.

EAT = hit ratio * (TLB access time + Main memory access time) + (1 – hit ratio) * (TLB access time + 2 * main memory time)

1. Find the Average Effective Memory Access Time when 80 percent is the hit ratio and it take 20 nanoseconds to search TLB and 100 nanoseconds to access memory (Hint : then take 120

nanoseconds to access mapped memory). In Case of Miss, It also takes 100 nanosecond to access frame number. The Total time taken in case of Miss is (20 + 100 + 100 = 220 nanosecond)

Solution:

$$\text{Average Access Time} = 0.80 * (20+ 100) + 0.20 * (20 + 100 + 100)$$

$$= 0.80 * 120 + 0.20 * 220$$

$$= 96 + 44 = 140 \text{ nanoseconds}$$

2. Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is _____.

   Solution:

   Effective Access Time = hit ratio * (TLB access time + Main memory access time) +
   (1 – hit ratio) * (TLB access time + 2 * main memory time)

   = 0.6*(10+80) + (1-0.6)*(10+2*80)

   = 0.6 * (90) + 0.4 * (170)

   = 122