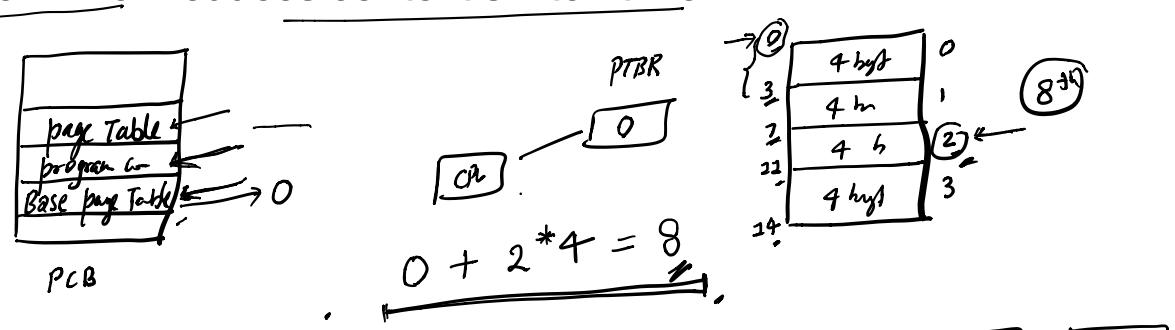
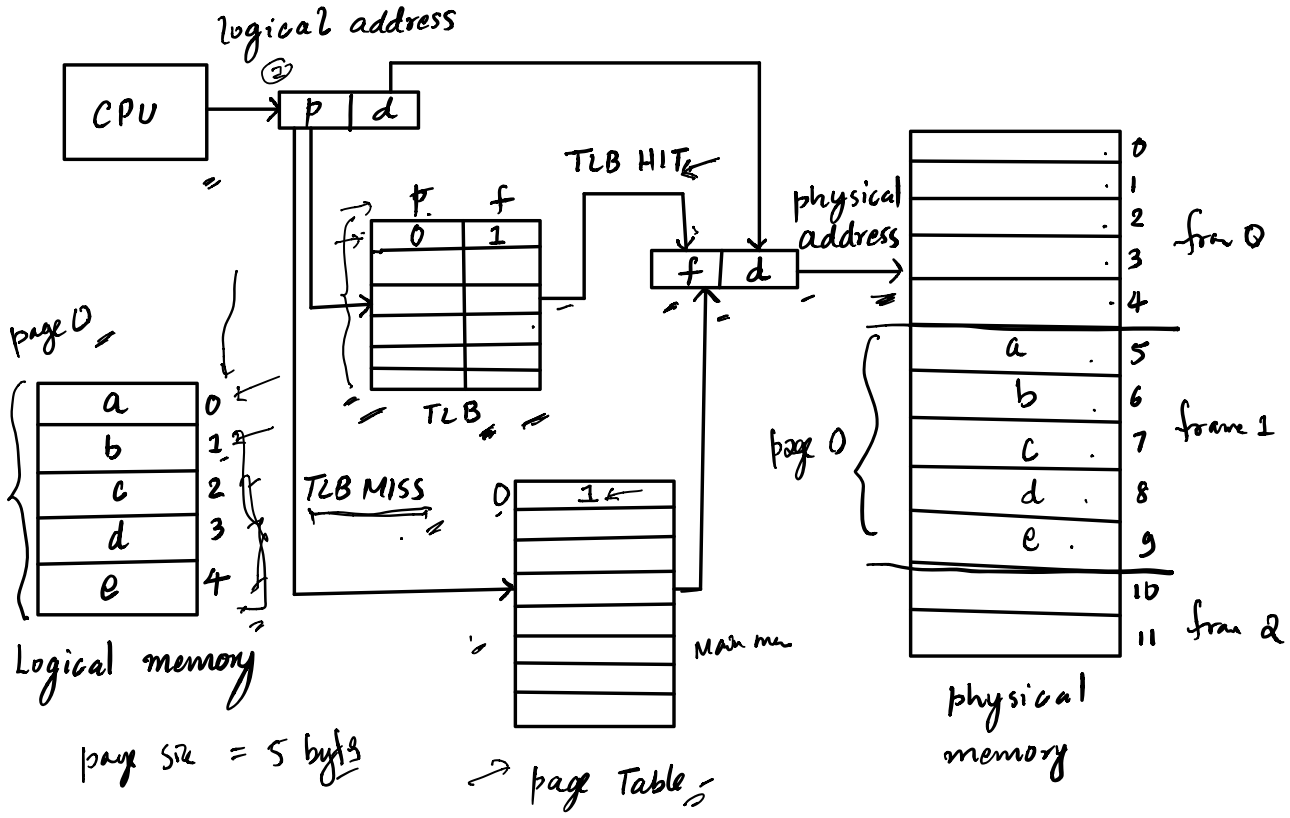
TRANSLATION LOOK-ASIDE BUFFER(TLB)

- Most operating system allocate a page table for each process
- The page table is stored in the PCB of the process and is kept in the main memory
- A pointer to the base address of the page table is also kept in the PCB like other registers values(program counter)
- Page-table base register(PTBR) stores the base address of the page table
- Changing the page table only requires changing this one register which reduces context switch time

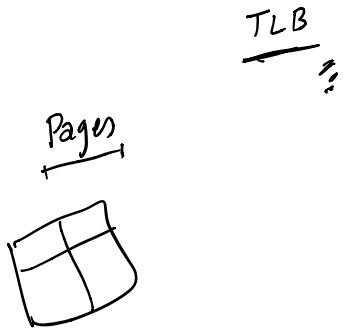


- But there is a problem with the above approach?
- Accessing a location in the main memory requires two memory accesses
  - One memory access to get the frame number from page table
  - Another memory access to access the byte or physical address in main memory
- Accessing memory twice for each instruction can be very expensive
- Can we try to minimise the number of calls we make to the page table?

- We can use a small hardware cache that sits near the CPU
- This cache is known as TLB
- Fastest access for CPU : registers > cache > RAM > disk
- Each entry in TLB consists of two parts : key(page number) and value(frame number)
- The search for a key is very fast in TLB but the hardware is expensive



- If the TLB is full of entries and the OS must select one for replacement
- Many replace algorithms exists - LRU, FIFO etc
- Some entries cannot be removed - the entries that are associated with kernel code


TLB
Pages