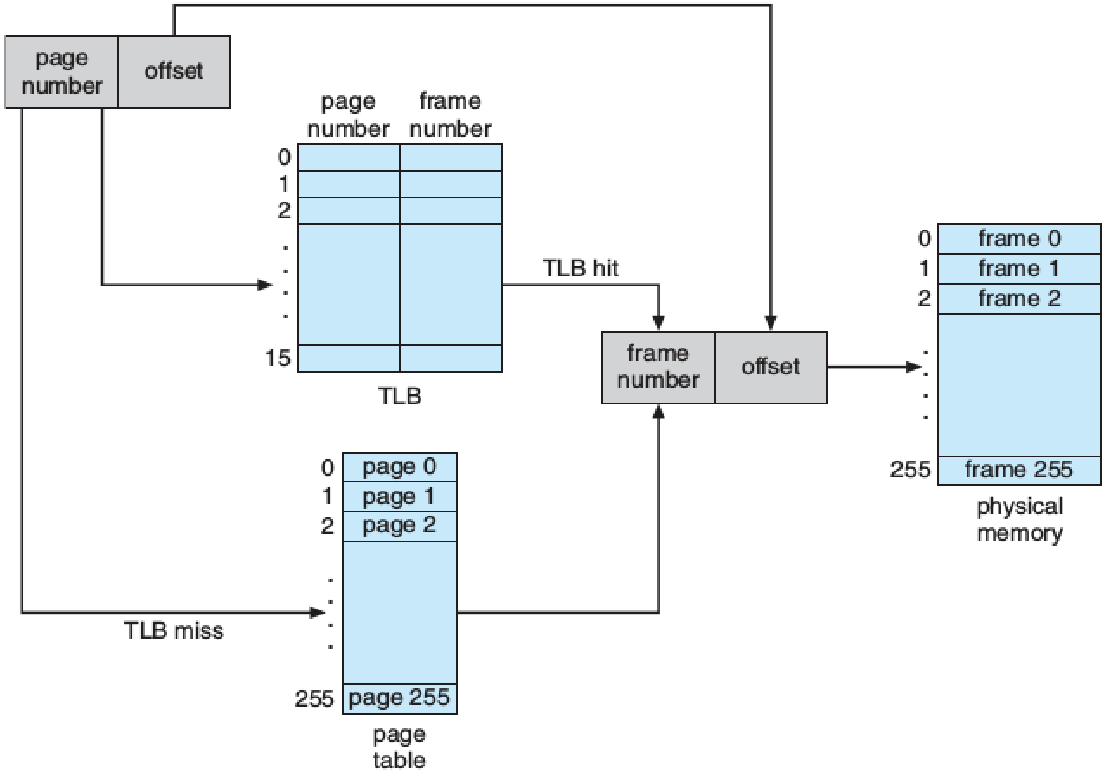
Designing a Virtual Memory Manager

Submitted to Sir Aamir Shafi
Submitted on Tuesday, 27th January'15
Presented by Atiqa Zafar

WHAT

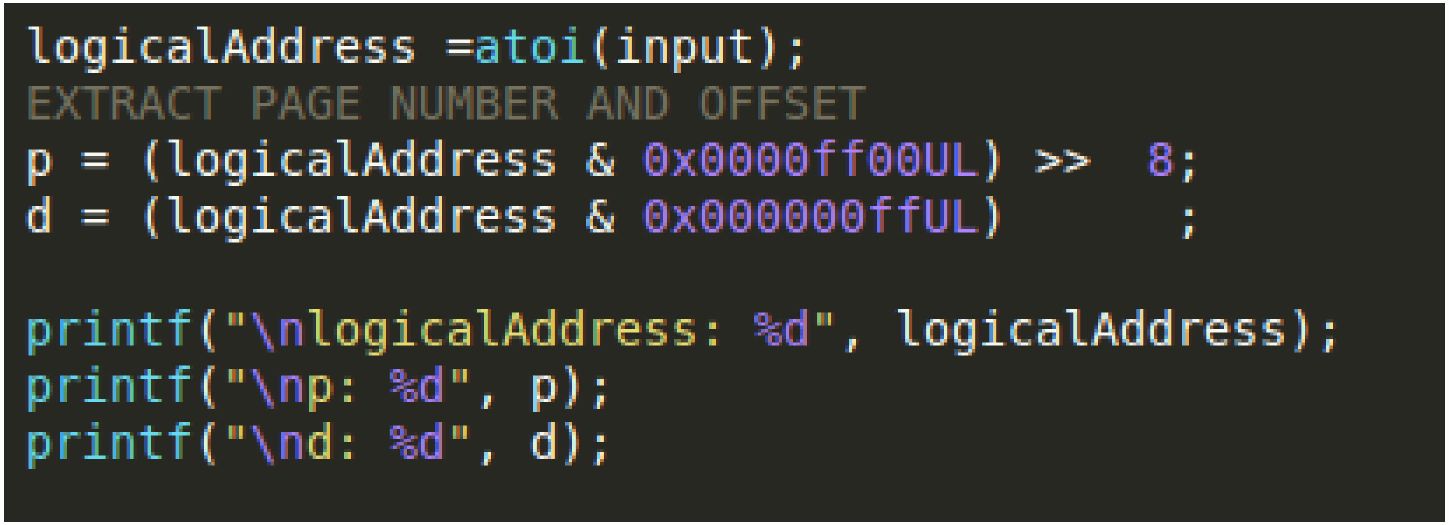
This project consists of writing a program that translates logical to physical addresses for a virtual address space of size 65,536 bytes.



Our program reads from a file containing logical addresses and, using a TLB as well as a page table, translates each logical address to its corresponding physical address and outputs the value of the byte stored at the translated physical address.

HOW

1) **Extracting Page Number and Offset from Logical Address** using bit shift operators



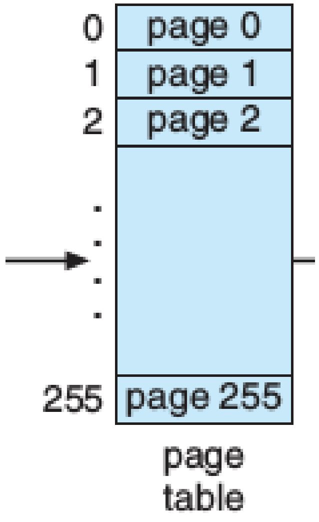
```
logicalAddress =atoi(input);
EXTRACT PAGE NUMBER AND OFFSET
p = (logicalAddress & 0x0000ff00UL) >>  8;
d = (logicalAddress & 0x000000ffUL)       ;

printf("\nlogicalAddress: %d", logicalAddress);
printf("\np: %d", p);
printf("\nd: %d", d);
```

HOW

2) Page Table

The page Table has 256 entries. Every page number is assigned a frame number as physical memory also has 256 frames.

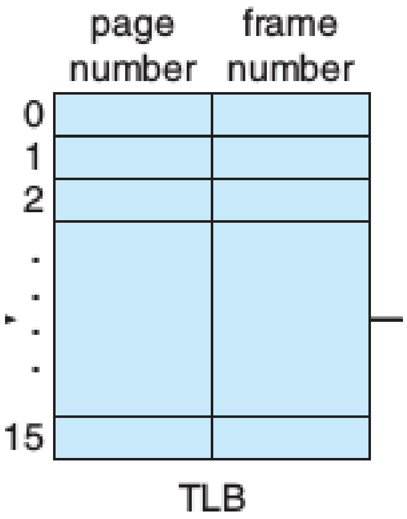
1. you get an address, fetch p and d
2. use p as an index into a page table to fetch f , table[p] = f
3. physical_address = (f * 256) + d



HOW

3) Translation Lookaside Buffer

The TLB has 16 entries. Every entry is page#, frame# pair cached.

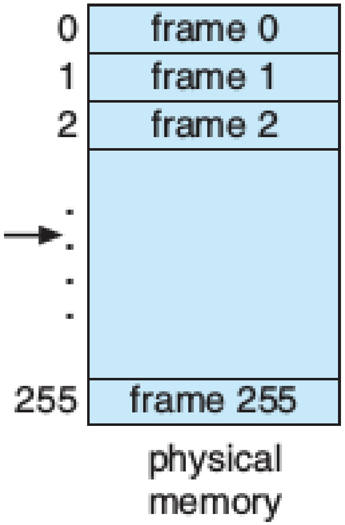


```
1. you get an address, fetch p and d
2. page-number compared with all keys (p)
3. if matched, return frame-number (f)
        4. physical_address = (f * 256) + d
  else, search pagetable
        4. use p as an index into a page table to fetch f , table[p] = f
        5. physical_address = (f * 256) + d
        6. update p,f to TLB using FIFO page replacement algo
```

HOW

PAGE FAULTS

1. locate free frame in physical memory (frames[ ])
2. read desired page from disk into allocated frame
3. update page table + TLB
4. resume program execution


0
frame 0
1
frame 1
2
frame 2
255
frame 255
physical memory