OS HW4

Designing a Virtual Memory Manager

Operating System 106 Fall

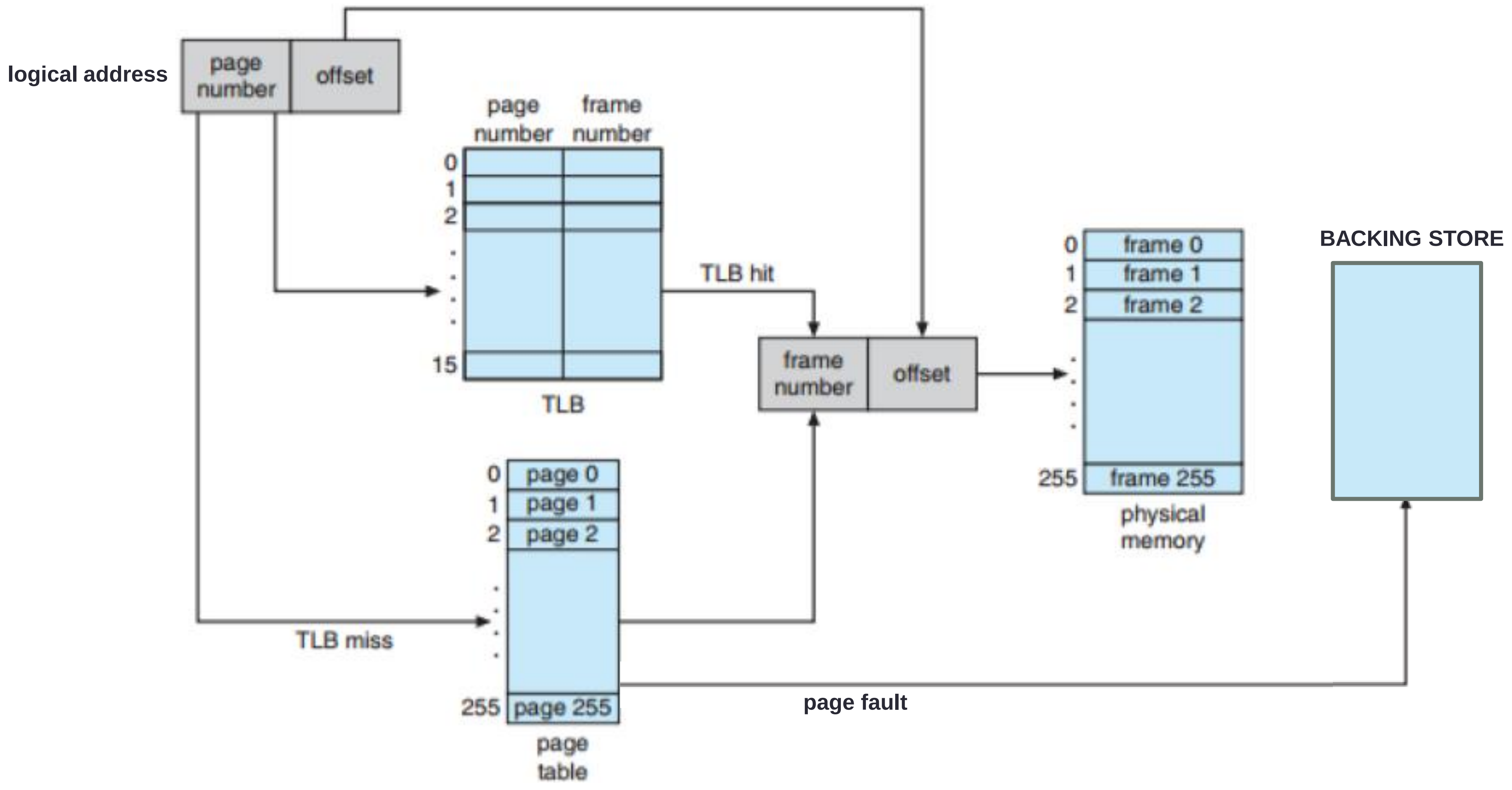
W.J.Tsai 蔡文錦 教授

TA 鍾陳恩
倪莞雅
巫怡慧
任佳靜

Goal

- Simulate the steps of translating logical addresses to physical addresses using translation look-aside buffers (TLB) and page table.
- Your program need to read logical addresses. Then, use a TLB and a page table to translate logical addresses to the corresponding physical addresses and output translated physical addresses and the byte stored at the physical memory.

Goal


logical address
page number
offset
page number
frame number
0
1
2
15
TLB
TLB hit
frame number
offset
0
1
2
frame 0
frame 1
frame 2
255
frame 255
physical memory
BACKING STORE
0
1
2
page 0
page 1
page 2
255
page 255
page table
TLB miss
page fault

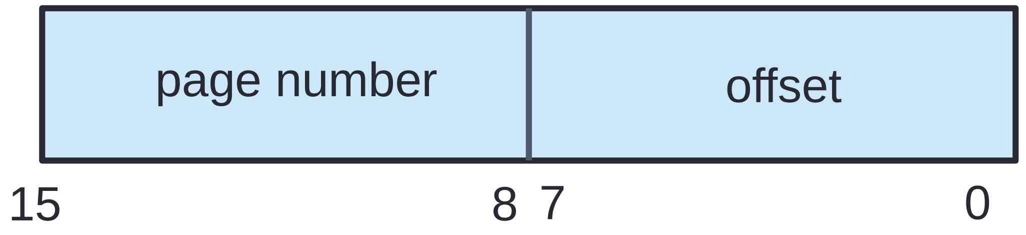
Specifics

Assume the TLB, page table, and physical memory is empty at the beginning.

- 2^4 entries in the TLB (Use LRU replacement Algorithm)
- 2^8 entries in the page table
- Page size of 2^8 bytes
- Frame size of 2^8 bytes
- 2^8 frames
- Physical memory of 65536 bytes (256 frames * 256-byte frame size)

Specifics

- Your program will read a file containing an integer numbers that represent logical addresses.
- These 16 bits logical address are divided into:
  - a page number consisting of 8 bits.
  - a page offset consisting of 8 bits.



Handling Page Fault

When a page fault occurs, you will read in a 256-byte page from the file BACKING_STORE.bin and then store it in an available page frame in physical memory and update TLB and page table.

Example:

- If a logical address with page number 15 resulted in a page fault, your program would read in the value of page 15 from BACKING_STORE.bin (remember that pages begin at 0 and are 256 bytes in size) and store the value in physical memory from 0 to 255 in order.

Input file

- BACKING_STORE.bin
  - A binary file of size 65536 (256*256) bytes.
  - Represent the backing store which store parts of pages.
  - When a page fault occurs, you need to read the correspondingly bytes and move to physical memory, updating TLB and page table.

Input file

- address.txt
  - Include n logical addresses.
  - First line implies the total number of logical addresses.
  - N = [20, 3000]

Example:

$$\begin{array}{ll} 30 & \Rightarrow \text{n} \\ \left.\begin{array}{l} 5129 \\ 58554 \\ 58584 \\ 27444 \\ \vdots \\ \vdots \end{array}\right\} & \text{n logical addresses} \end{array}$$

Output

Your hw4 program need to output the following file:

- results.txt
  - Each line consists of physical address and value according to addresses.
  - Last two line output the number of TLB hits and Page faults.
  - The output format must be same as the example TAs given and name it as “results.txt”.

Output

Your hw4 program need to output the following file:

- results.txt

Examples:

```
9      0
442    57
472    57
⋮      ⋮
⋮      ⋮
5857   0
TLB hits: 7
Page faults: 23
```

physical address & value of n logical address

TLB hits and page faults

Appendix

You can use following function to read the content in BACKING_STORE.bin:

```
int fseek(FILE *stream, long int offset, int origin);
```

Sets the position indicator associated with the file stream to a new position.

- stream: Pointer to a FILE object.
- offset: In binary files, it represents number of bytes to offset from *origin*.
- origin: Position used as reference for the *offset*

Constant	Reference position
SEEK_SET	Beginning of file
SEEK_CUR	Current position of the file pointer
SEEK_END	End of file

Appendix

You can use following function to read the content in BACKING_STORE.bin:

```
size_t fread(void *ptr, size_t size, size_t count,
                FILE *stream);
```

Read block of data from the file.

- ptr: Pointer to a block of memory with a size of at least (size*count) bytes, converted to a void*.
- size: Size, in bytes, of each element to be read. size_t is an unsigned int type.
- count: Number of elements, each one with a size of *size* bytes.
- stream: Pointer to a FILE object that specifies an input stream.

Requirements

- Use NCTU CS Workstation **linux1~linux6** as your programming environment. (No bsd1~bsd6)
- We only these commends on NCTU CS Workstation linux2:
  - g++ -std=c++11 StudentID_hw4.cpp
  - ./a.out BACKING_STORE.bin address.txt ← Use argv[1] to read BACKING_STORE.bin and argv[2] to read address.txt
- Put the file into a compressed file named "**StudentID_OS_hw4.zip**"
  - StudentID_hw4.cpp
- Wrong input/output format: -10 pts
- Wrong hand-in file name: -10 pts
- Copy or be copied: will get 0 pt directly
- Deadline: 2017/12/30 (Saturday) 23:59