



Memory Management - Part II

Amir H. Payberah
payberah@kth.se
2022





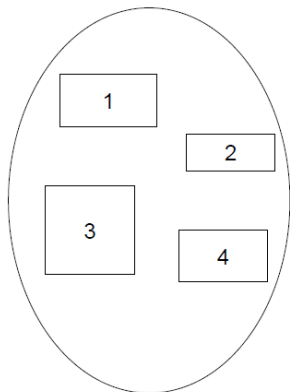
Reminder



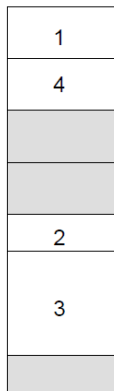
Reminder (1/3)

- ▶ External fragmentation vs. internal fragmentation
- ▶ **Compaction**: shuffle memory contents to place all free memory together in **one large block**.
- ▶ Other solutions:
 - Segmentation
 - Paging

Reminder (2/3)

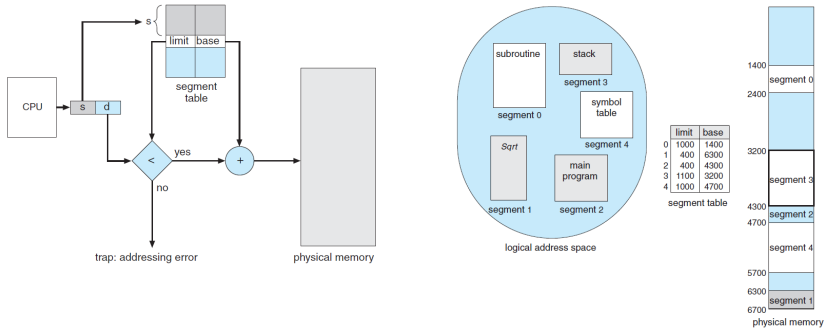


user space



physical memory space

Reminder (3/3)



- ▶ A reference to byte 53 of segment 2: $4300 + 53 = 4353$
- ▶ A reference to byte 852 of segment 3: $3200 + 852 = 4052$
- ▶ A reference to byte 1222 of segment 0: trap to OS

Paging



Paging vs. Segmentation

- ▶ **Segmentation** and **paging**, both, permit the physical address space of a process to be **noncontiguous**.
- ▶ Paging **avoids external fragmentation** and the need for **compaction**, whereas segmentation does not.



Paging (1/2)

- ▶ Physical address space of a process can be **noncontiguous**; process is allocated physical memory whenever the latter is available.
- ▶ Divide **physical memory** into **fixed-sized blocks** called **frames**.
 - Size is **power of 2**, between 512 bytes and 16 Mbytes.
- ▶ Divide **logical memory** into **blocks of same size** called **pages**.

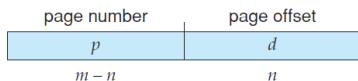


Paging (2/2)

- ▶ Keep track of all free frames.
- ▶ To run a program of size N pages, need to find N free frames and load program.
- ▶ Set up a page table to translate logical to physical addresses.
- ▶ Still have internal fragmentation.

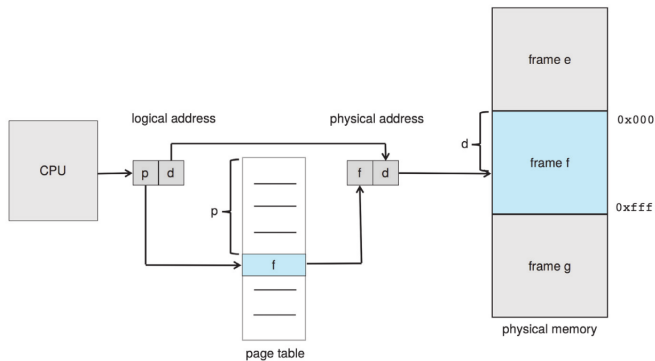
Address Translation Scheme

- ▶ **Logical address** generated by CPU is divided into **two parts**:
- ▶ **Page number** (p): used as an **index into a page table** that contains **base address of each page** in **physical memory**.
- ▶ **Page offset** (d): combined with **base address** to define the **physical memory address**.

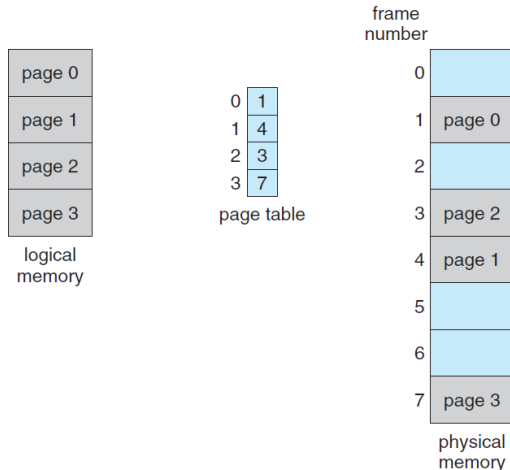


- ▶ For given logical address space 2^m and page size 2^n .

Paging Hardware



Paging Model of Logical and Physical Memory



Paging Example

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

logical memory

0	5
1	6
2	1
3	2

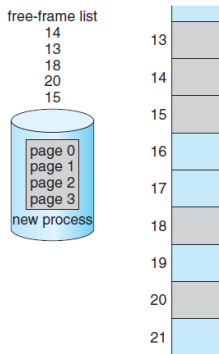
page table

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

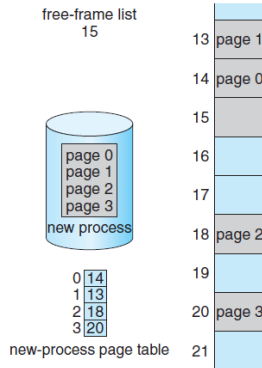
physical memory

- ▶ The logical address: $m = 4$ and $n = 2$
- ▶ Logical address 3: $5 \times 4 + 3 = 23$
- ▶ Logical address 10: $1 \times 4 + 2 = 6$

Free Frames



before allocation



after allocation



Paging Example - Internal Fragmentation

- ▶ Page size = 2048 bytes
- ▶ Process size = 72766 bytes
- ▶ $\frac{72766}{2048} = 35 \text{ pages} + 1086 \text{ bytes}$
- ▶ Internal fragmentation: $2048 - 1086 = 962 \text{ bytes}$
- ▶ Worst case fragmentation = 1 frame - 1 byte
- ▶ On average fragmentation = $\frac{1}{2}$ frame size



Small Page Size vs. Big Page Size

- ▶ On average fragmentation = $\frac{1}{2}$ page size, hence, **small page** sizes are desirable.
- ▶ **Small pages**, more **overhead** is in the page-table, this overhead is reduced as the size of the pages **increases**.
- ▶ **Disk I/O** is more efficient when the amount data being transferred is **larger** (e.g., big pages).
- ▶ Pages typically are between 4 KB and 8 KB in size.

```
getconf PAGESIZE
```




Page Table Implementation



Page Table

- ▶ Page table is kept in main memory.
- ▶ Page-table base register (PTBR) points to the page table.
- ▶ Page-table length register (PTLR) indicates size of the page table.
- ▶ In this scheme every data/instruction access requires two memory accesses.
 - One for the page table and one for the data/instruction.



Translation Look-aside Buffers (1/2)

- ▶ The **two memory access problem** can be solved by the use of a special **fast-lookup hardware cache** called **translation look-aside buffers (TLBs)**.



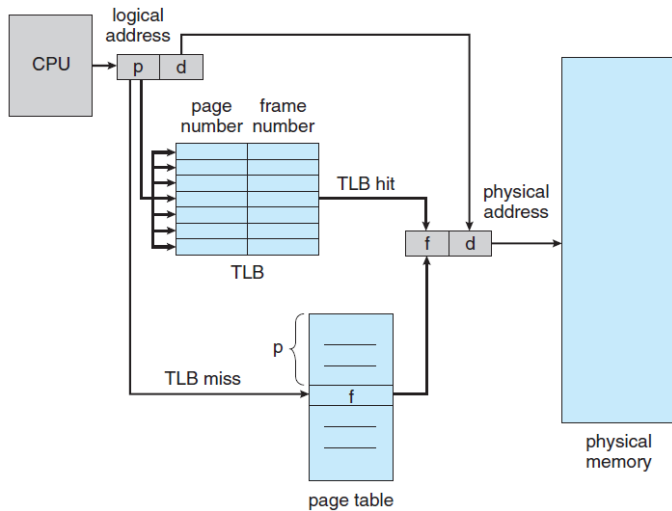
Translation Look-aside Buffers (2/2)

▶ TLB

Page #	Frame #

- ▶ Address translation (p, d)
- If p is in TLB, get $frame\#$ out.
 - Otherwise, get $frame\#$ from page table.

Paging Hardware With TLB





Shared Pages



Shared Pages

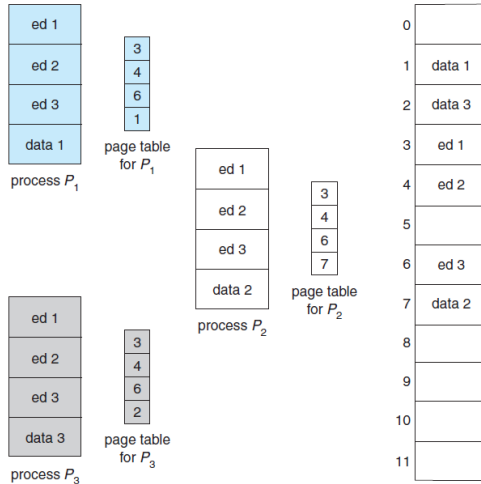
▶ Shared code

- One copy of **read-only code shared among processes** (e.g., text editors).
- Similar to multiple threads sharing the same process space.

▶ Private code and data

- Each process keeps a **separate copy of the code and data**.

Shared Pages Example



Structure of the Page Table



Structure of the Page Table (1/2)

- ▶ Consider a 32-bit logical address space ($m = 32$):
 - Page size of $4KB = 2^{12}$ ($n = 12$).
 - Page table would have 1 million entries $m - n = 20$.
 - If each entry is $4B$: $4MB$ of physical address space memory for page table alone.
 - That amount of memory used to cost a lot.
 - Don't want to allocate that contiguously in main memory.



Structure of the Page Table (2/2)

- ▶ Hashed Page Tables
- ▶ Hierarchical Paging

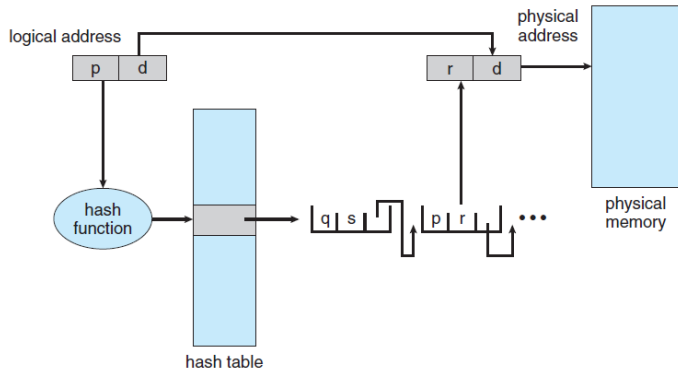
Hashed Page Tables



Hashed Page Tables

- ▶ The logical page number is hashed into a page table.
- ▶ This page table contains a chain of elements hashing to the same location.
- ▶ Each element contains
 1. The logical page number
 2. The value of the mapped page frame
 3. A pointer to the next element

Hashed Page Table Architecture



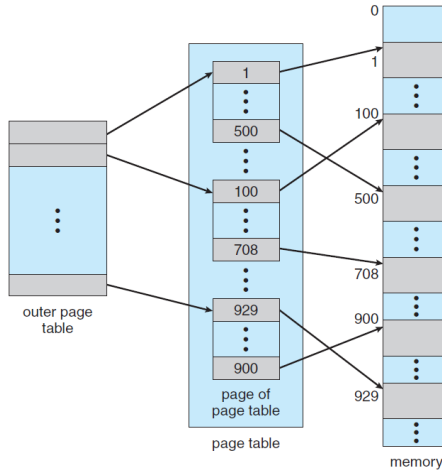
Hierarchical Paging



Hierarchical Page Tables

- ▶ Break up the **logical address** space into **multiple page tables**.
- ▶ A simple technique is a **two-level page table**.
- ▶ We then **page** the **page table**.

Two-Level Page-Table Scheme



Two-Level Paging Example

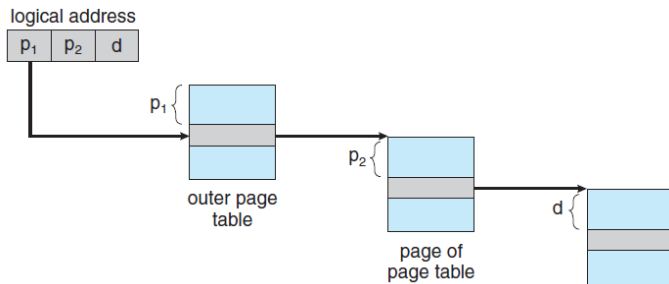
- ▶ A logical address, on 32-bit machine with 1K page size, is divided:
 - A page number consisting of 22 bits.
 - A page offset consisting of 10 bits.
- ▶ Since the page table is paged, the page number is divided into:
 - A 12-bit page number.
 - A 10-bit page offset.

- ▶ Thus, a logical address is:

page number		page offset
p_1	p_2	d
12	10	10

- ▶ p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table.
- ▶ Known as forward-mapped page table.

Address-Translation Scheme



Summary



Summary

- ▶ Paging vs. Segmentation
- ▶ Physical memory: frames, Logical memory: pages
- ▶ Page table: translates logical to physical addresses
- ▶ Translation Look-aside Buffer (TLB)
- ▶ Page table structure: hierarchical paging, hashed page tables

Questions?

Acknowledgements

Some slides were derived from Avi Silberschatz slides.