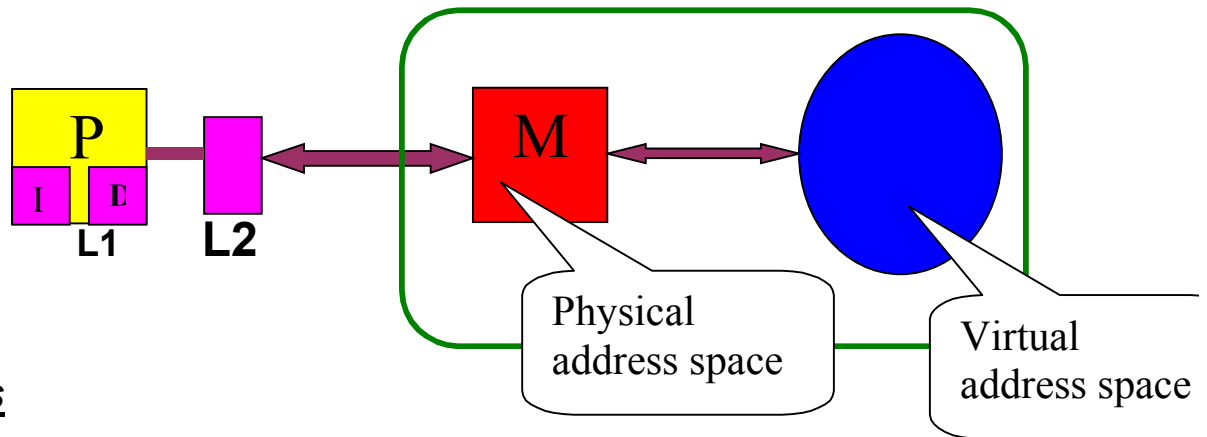


Virtual memory



Goals

1. Creates the **illusion** of an **address space much larger than the physical memory**
2. Make provisions for **protection**

The main idea is that if a virtual address is not mapped into the physical memory, then it has to be fetched from the disk. The unit of transfer is **a page** (or **a segment**). Observe the similarities (as well as differences) between virtual memory and cache memory. Also, recall how slow is the disk (~ ms) to the main memory (50 ns). So each miss (called a **page fault** or a segment fault) has a large penalty.

What is a **page**? What is a **segment**?

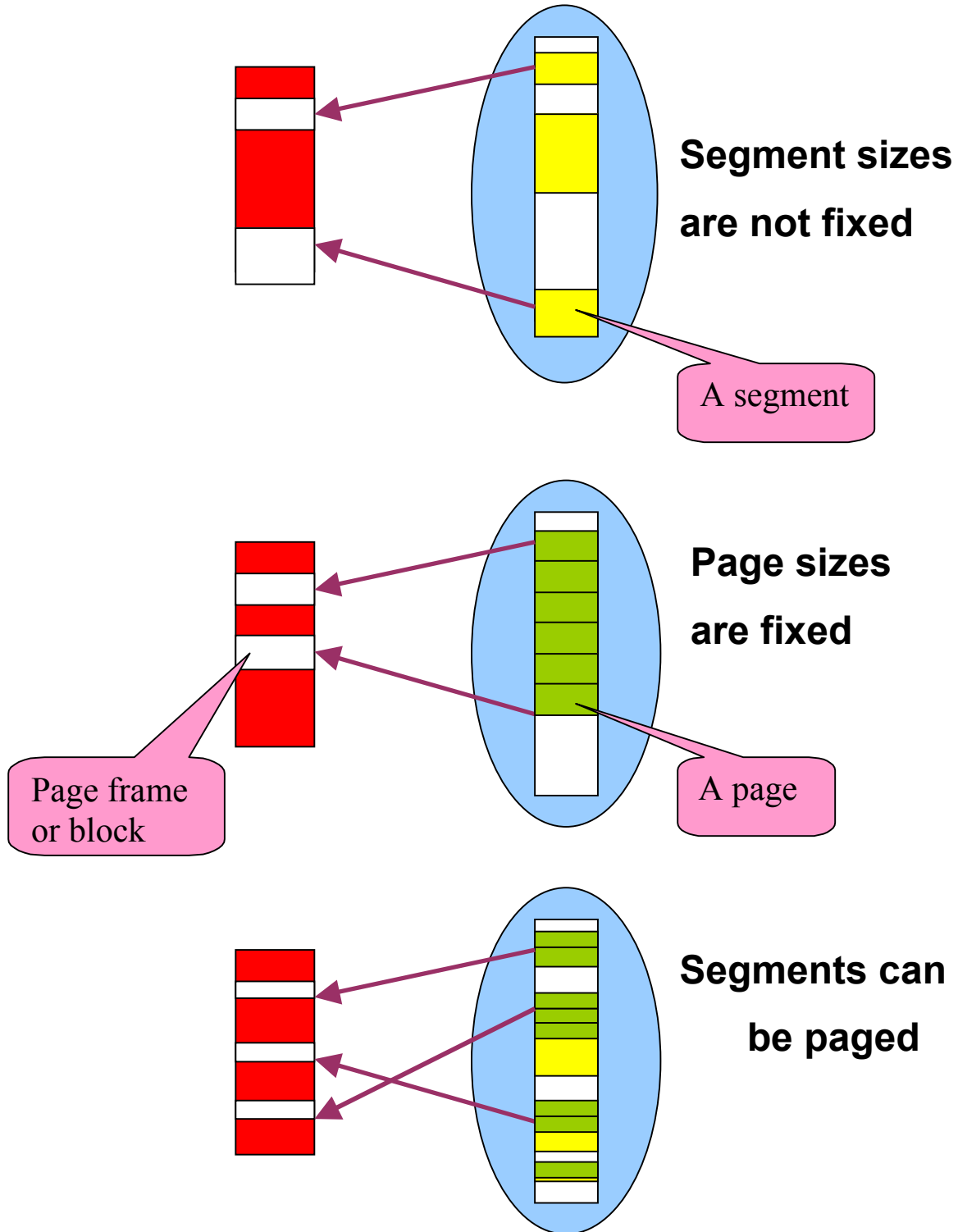
A page is a **fixed size** fragment (say 4KB or 8 KB) of code or data. A segment is a logical component of the program (like a subroutine) or data (like a table). The size is variable.

VM Types

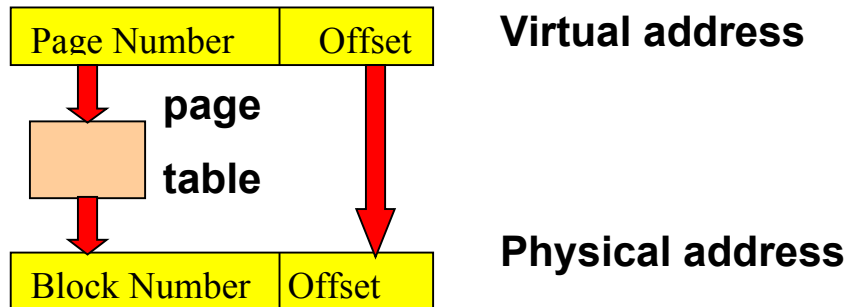
Segmented, paged, segmented and paged.

Page size	4KB –64 KB
Hit time	50-100 CPU clock cycles
Miss penalty	$10^6 - 10^7$ clock cycles
Access time	$0.8 \times 10^6 - 0.8 \times 10^7$ clock cycles
Transfer time	$0.2 \times 10^6 - 0.2 \times 10^7$ clock cycles
Miss rate	0.00001% - 0.001%
Virtual address Space size	4 GB - 16×10^{18} byte

A quick look at different types of VM



Address Translation



Page Table (Direct Map Format)

Page No.	Presence bit	Block no./ Disk addr	Other attributes like protection
0	1	7	Read only
1	0	Sector 6, Track 18	
2	1	45	Not cacheable
3	1	4	
4	0	Sector 24, Track 32	

Page Table (Associative Map Format)

Pg, Blk, P	Block no./ Disk addr	Other attributes
0, 7, 1	7	Read only
1, ?, 0	Sector 6, Track 18	
2, 45, 1	45	Not cacheable
3, 4, 1	4	
4, ?, 0	Sector 24, Track 32	

Address translation overhead

Average Memory Access Time =

Hit time (no page fault) +

Miss rate (page fault rate) x Miss penalty

Examples of VM performance

Hit time = 50 ns.

Page fault rate = 0.001%

Miss penalty = 2 ms

$$T_{av} = 50 + 10^{-5} \times 2 \times 10^6 \text{ ns} = 70 \text{ ns.}$$

Improving the Performance of Virtual Memory

1. Hit time involves one extra table lookup. *Hit time* can be reduced using a TLB

(TLB = Translation Lookaside Buffer).

2. *Miss rate* can be reduced by allocating enough memory to hold the working set. Otherwise, thrashing is a possibility.

3. *Miss penalty* can be reduced by using disk cache

Page Replacement policy

Determines which page needs to be discarded to accommodate an incoming page. Common policies are

Least Recently Used (LRU)

Least Frequently Used (LFU)

Random

Writing into VM

Write-through is possible if a **write buffer** is used.

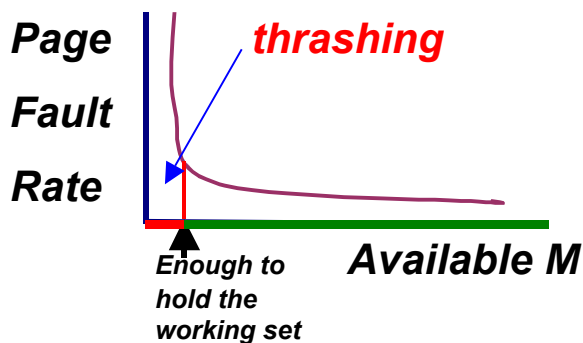
But write-back makes more sense. The page table must keep track of **dirty** pages. There is **no overhead** to discard a **clean page**, but to discard **dirty pages**, they must be written back to the disk.

Working Set

Consider a page reference string

0, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, ... 100,000 references

The size of the *working set* is 2 pages.

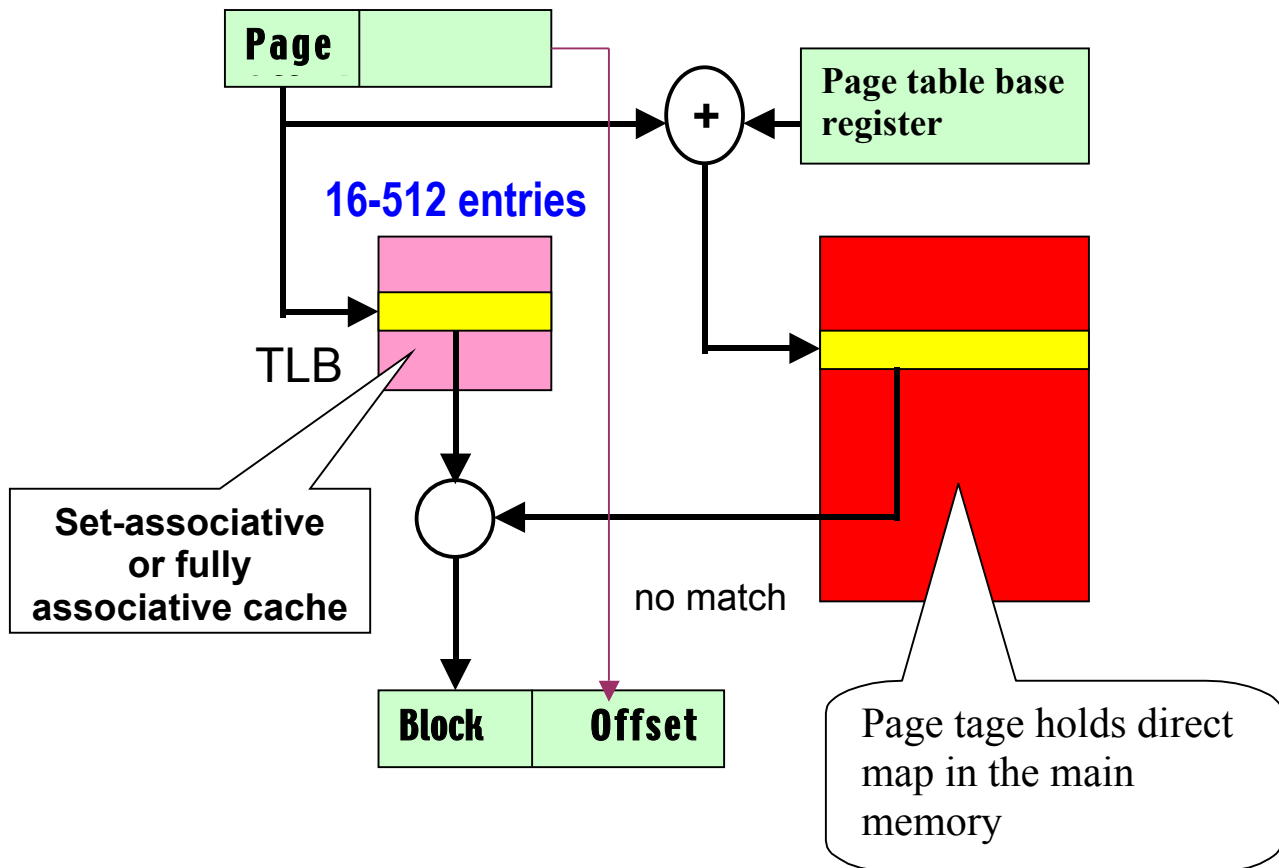


Always allocate enough memory to hold the working set of a program (**Working Set Principle**)

Disk cache

Modern computers allocate up a large fraction of the main memory as file cache. Similar principles apply to disk cache that drastically reduces the miss penalty.

Address Translation Using TLB



TLB is a **set-associative cache** that holds a partial page table. In case of a TLB hit, the block number is obtained from the TLB (fast mode). Otherwise (i.e. for TLB miss), the block number is obtained from the direct map of the page table in the main memory, and the TLB is updated.