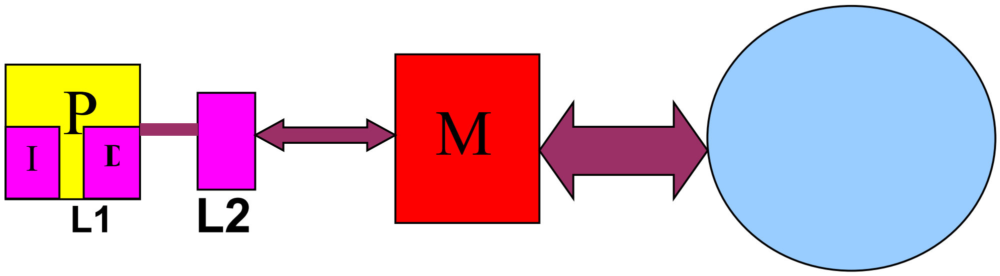
Virtual memory



Goals

1. Support large virtual address space
2. Make provisions for protection

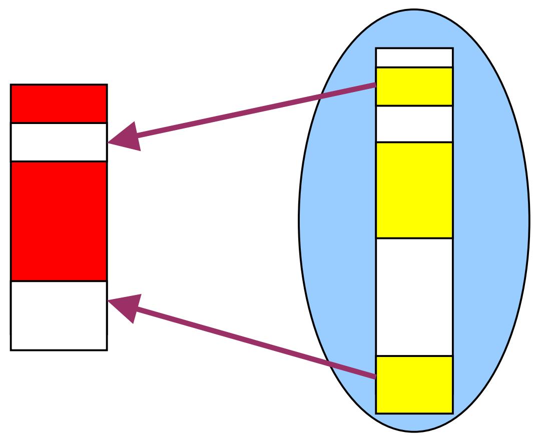
Types

Segmented, paged, segmented and paged.

Page size	4KB –64 KB
Hit time	50-100 CPU clock cycles
Miss penalty Access time Transfer time	10^6 - 10^7 clock cycles 0.8 x 10^6 –0.8 x 10^7 clock cycles 0.2 x 10^6 –0.2 x 10^7 clock cycles
Miss rate	0.0001% - 0.001%
Virtual address Space size	4 GB -16 x 10^{18} byte

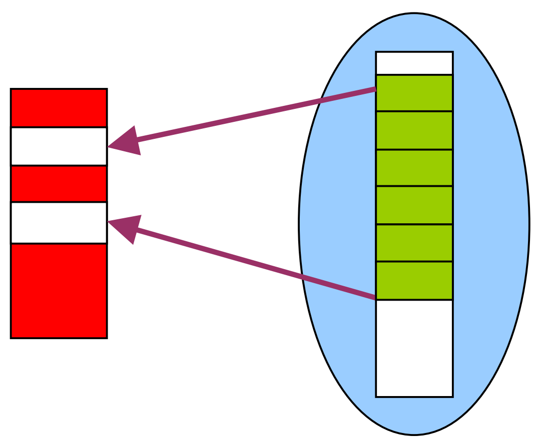
A quick look at different types of VM

sizes



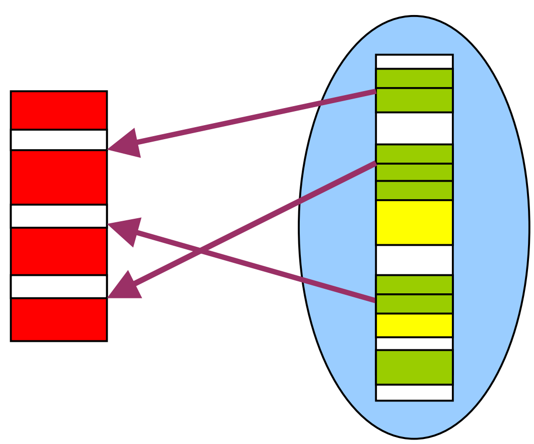
Segments

Sizes not fixed



Page sizes

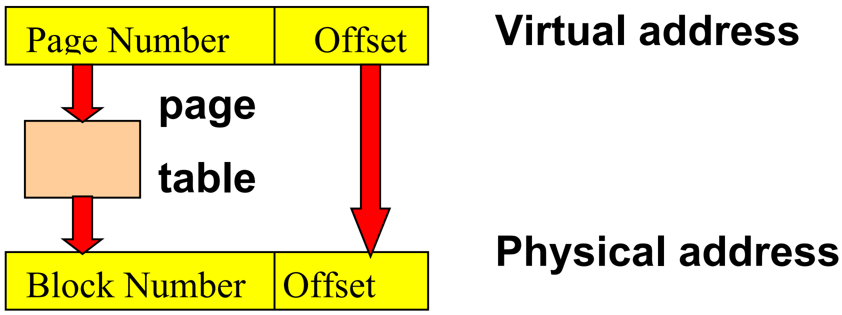
are fixed



Segments can

be paged

Address Translation



Page Table format for Direct Map

Page No.	Presence Bit P	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 format for Associative Map

Pg, Blk, P	Block no./ Disk address	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

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

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

Writing into VM

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

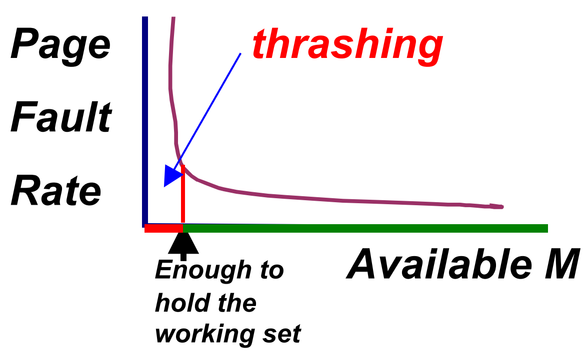
But write-back makes more sense. The page table must keep track of dirty pages. There is no overhead to discard a clean page, but to discard dirty pages, they must be written back to the disk.

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**.**

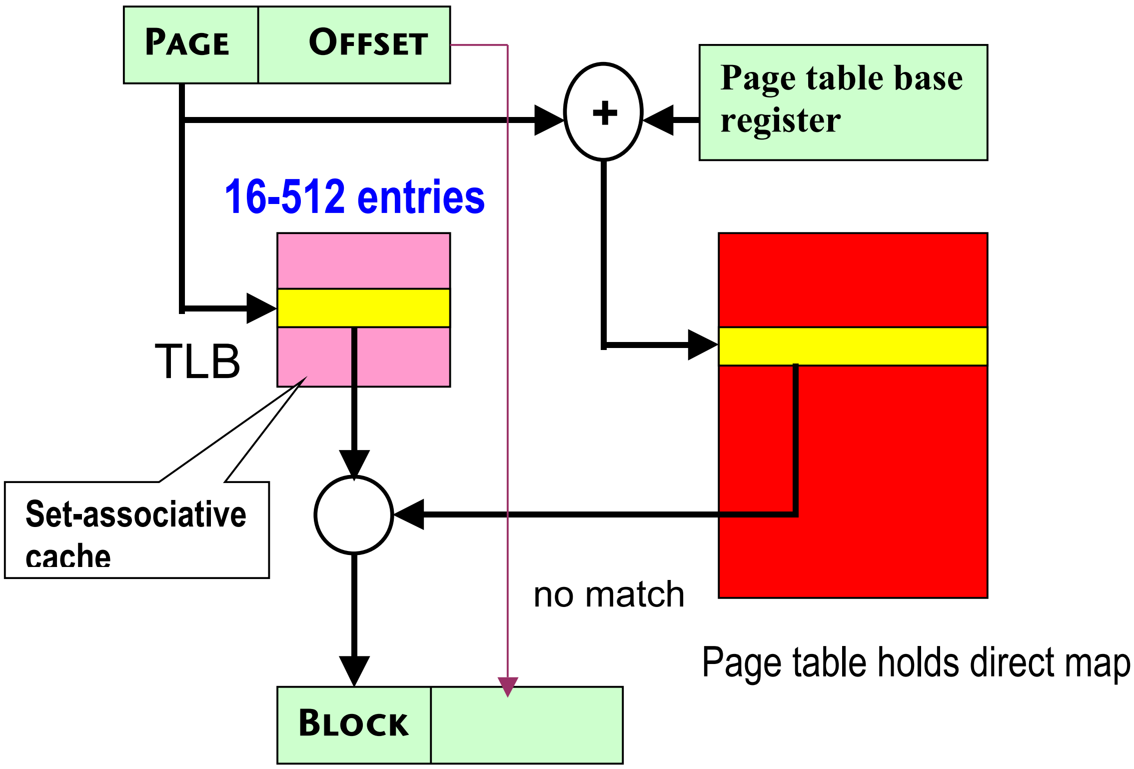


Always allocate enough memory to hold the working set of a program (Working Set Principle)

Disk cache

Modern computers allocate up to 80% of the main memory as file cache. Similar principles apply to disk cache, which can drastically reduce 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, and the TLB is updated.

Multi-level Address Translation

Example 1: The old story of VAX 11/780

30-bit virtual address (1 GB) per user

Page size = 512 bytes = 2^9

Maximum number of pages = 2^{21} i.e. 2 million

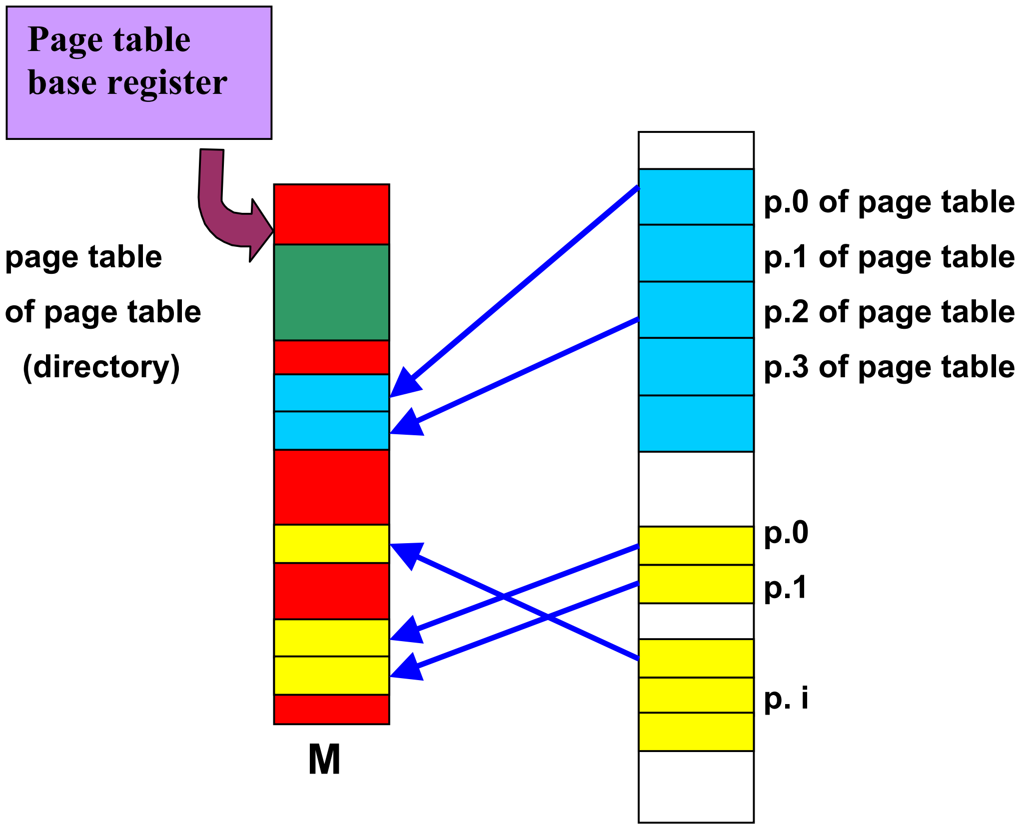
Needs 8 MB to store the page table. Too big!

Solution?

Store the page table in Virtual Memory.

Thus, page table is also paged!

Two-level address translation



Virtual address space

12	9	9
Directory Entry	Offset in page table	Offset in page

Virtual Address