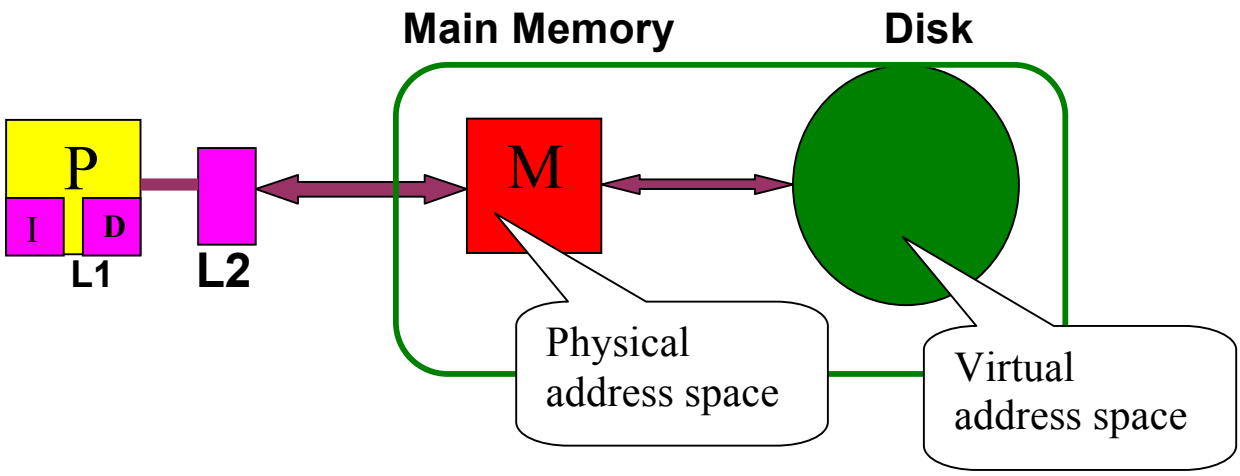
Virtual memory

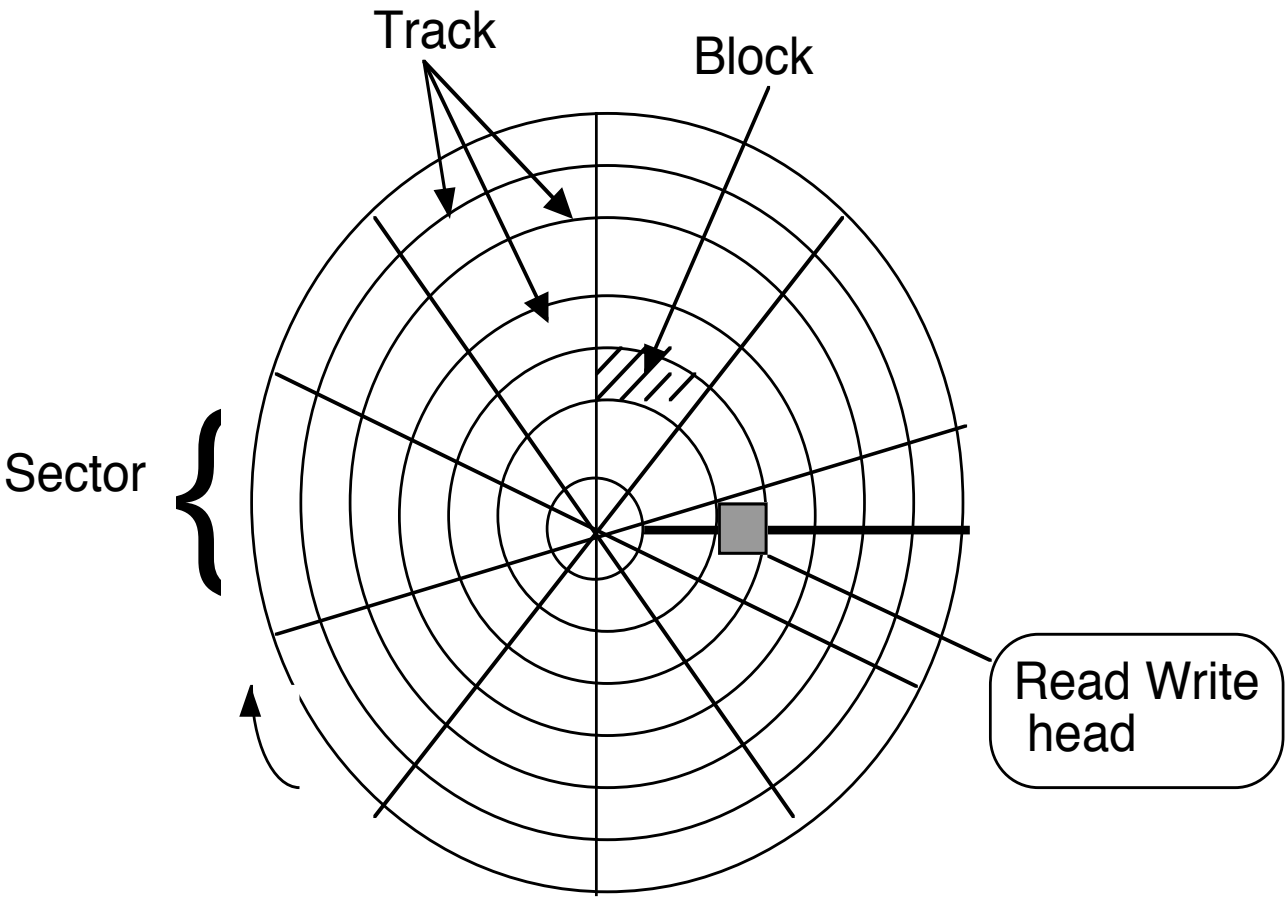


Goals

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.



DISK

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.

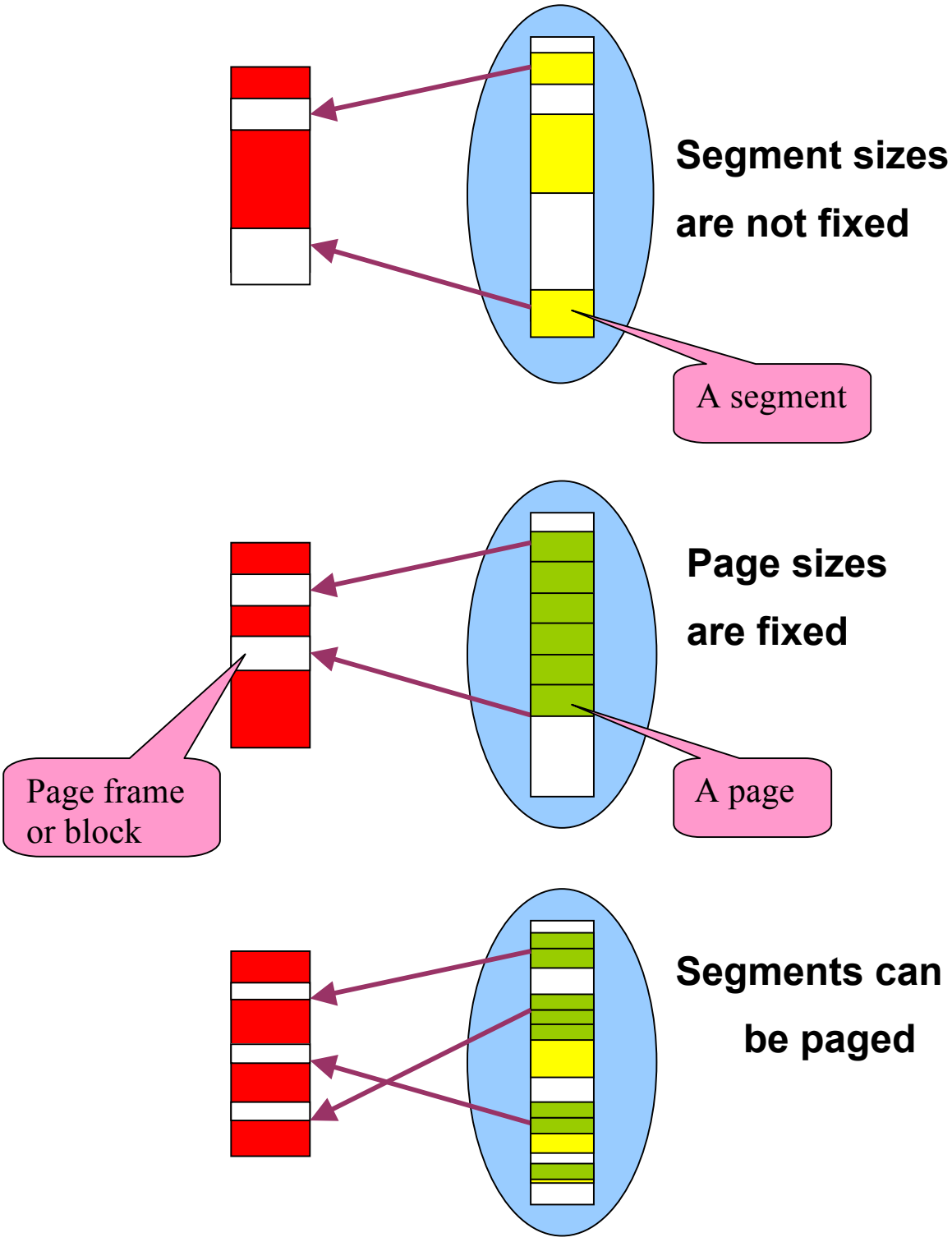
VM Types

Segmented, paged, segmented and paged**.**

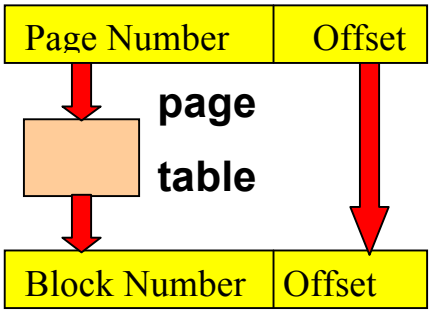
Specification of a paged VM

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

A quick look at different types of VM



Address Translation



Virtual address

Physical address

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)

Page	Block no.	Other attributes
0	7	Read only
2	45	Not cacheable
3	4	Read-write

The associate search answers the question: If page xyz available in the main memory? It is like a tag search in a cache memory. If the page is not available then its location on the disk can be obtained from the traditional version pf the page table.

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. *To reduce the miss rate,* allocate enough memory that can hold the "working set". Otherwise, thrashing is a possibility (the bulk of the CPU time is wasted moving the pages back and forth .

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

Page Replacement policy

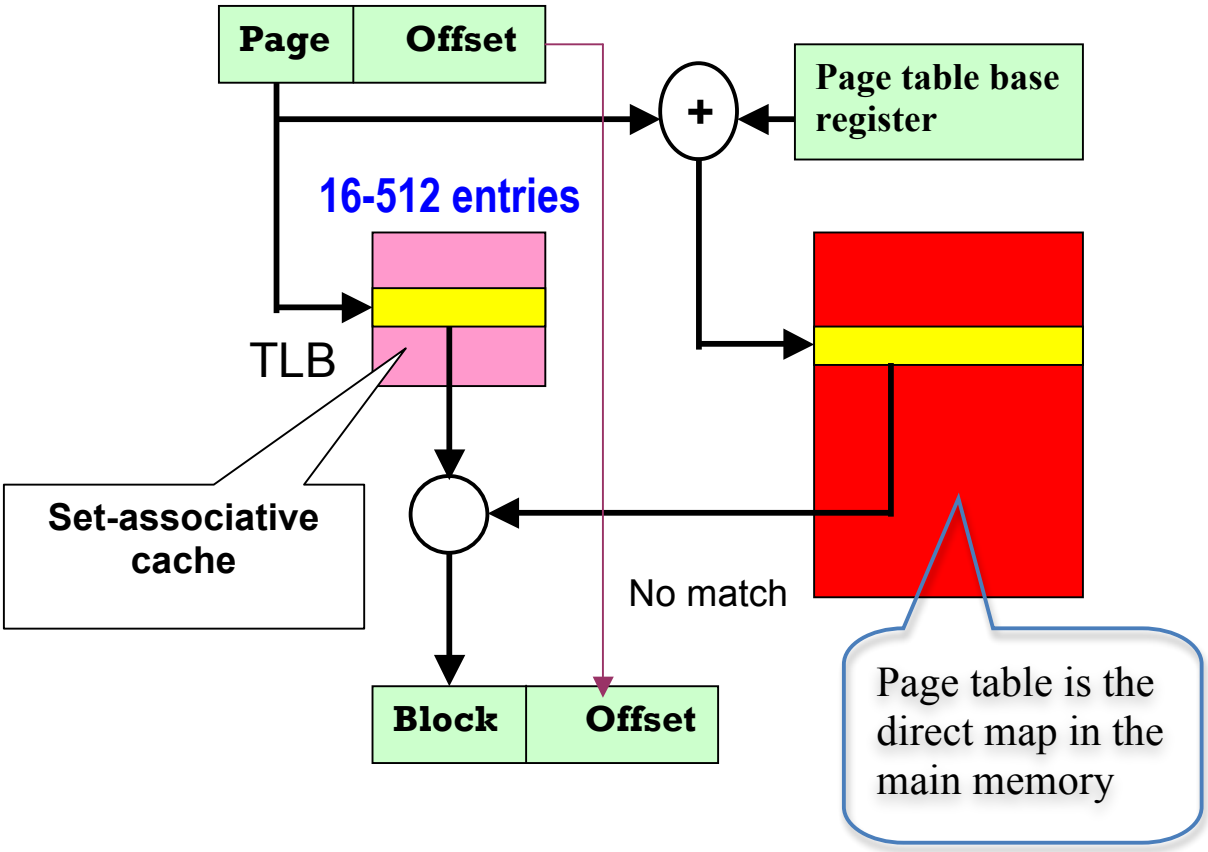
Determines which page needs to be discarded to accommodate an incoming page. Common policies are

- Least Recently Used (LRU)
- Least Frequently Used (LFU)
- Random

Writing into VM

Write-through is not practical, 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.

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.

Examples

VAX 11/780 (1978)

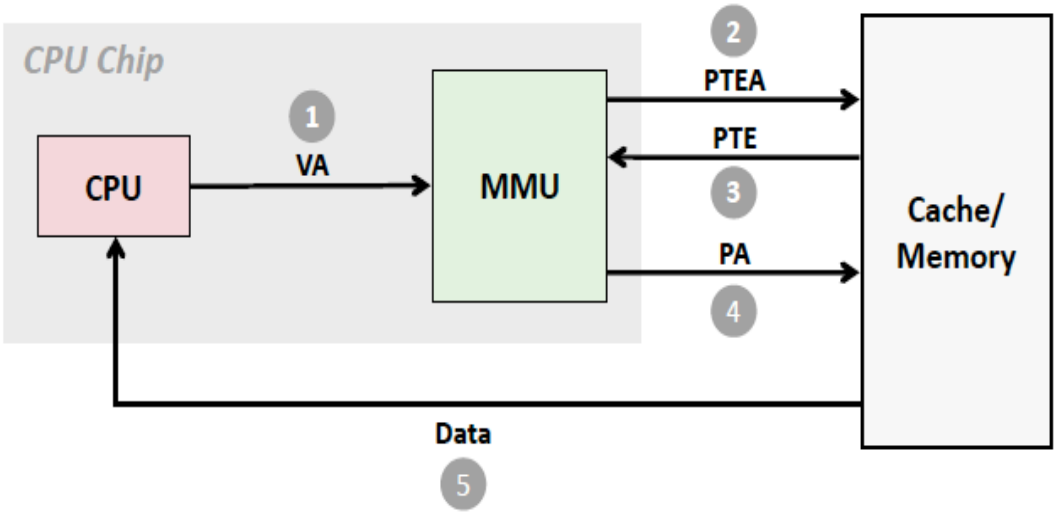
- 32-bit virtual address, 32-bit physical address, 512 bytes page
- How big are the (virtual) Page number and Offset?
- How big are the (physical) Block number & Offset?

Intel Core i7 (2008)

- Current implementation (Bloomfield, Lynnfield etc) has 48-bit virtual address, 52-bit physical address, 4MB page (Linux uses 4 KB pages)
- How many page table entries?

Memory Management Unit (MMU) facilitates the address translation. It contains the small hardware cache called TLB

Address Translation: Page Hit



1) Processor sends virtual address to MMU

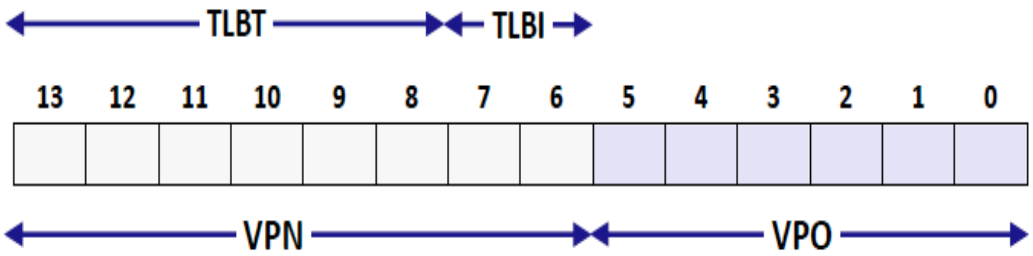
2-3) MMU fetches PTE from page table in memory

4) MMU sends physical address to cache/memory

5) Cache/memory sends data word to processor

Example: TLB

- 16 entries
- 4-way set associative (TLBT = TLB Tag, TLBI = TLB Index)



Set	Tag	Valid	PPN	Tag	Valid	PPN	Tag	Valid	PPN	Tag	Valid	PPN
0	03	0	--	09	1	0D	00	0	--	07	1	02
1	03	1	2D	02	0	--	04	0	--	0A	0	--
2	02	0	--	08	0	--	06	0	--	03	0	--
3	07	0	--	03	1	0D	0A	1	34	02	0	--

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

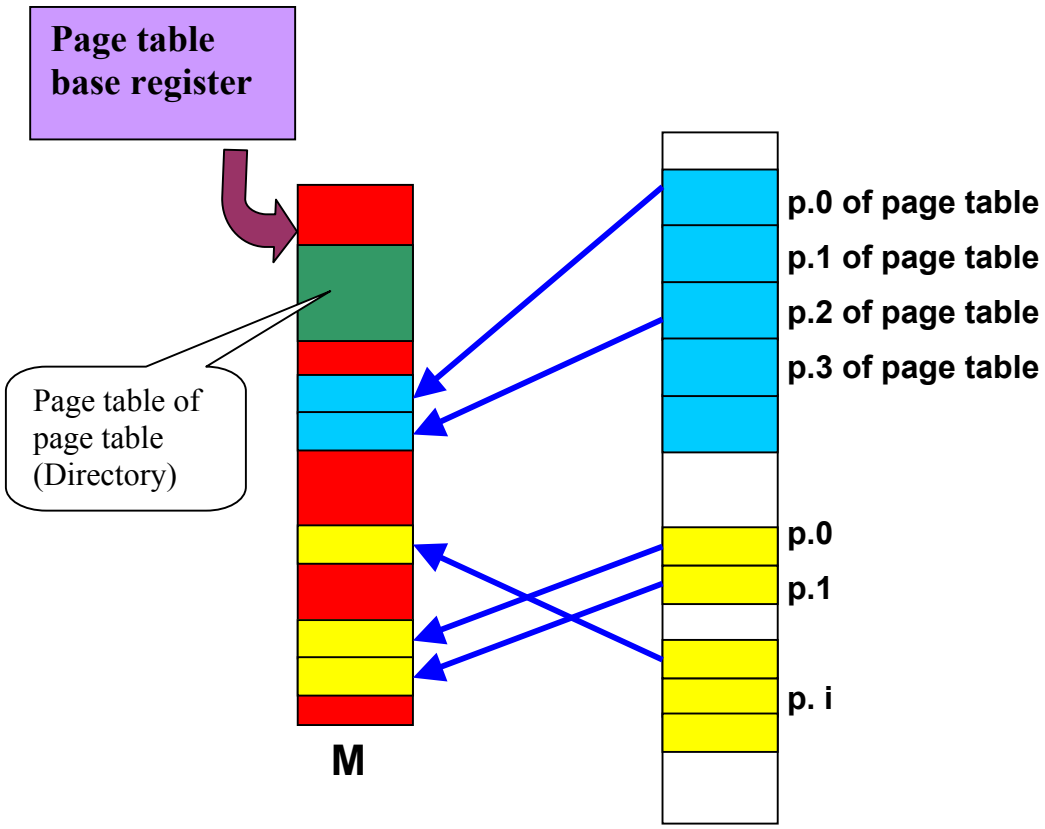
Needs 8 MB to store the page table. Too big!

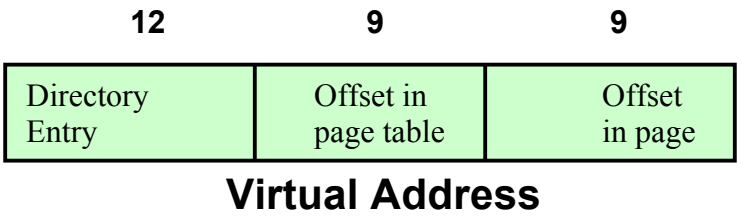
Solution?

Store the page table in Virtual Memory.

Thus, page table is also paged!

Address translation in VAX 11/780

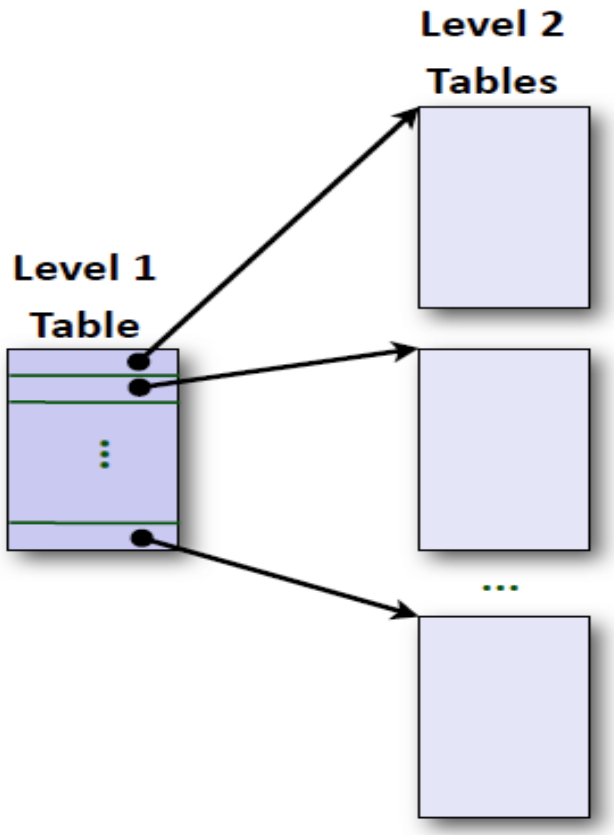




Multi-level Address Translation in Intel i7

Consider an Intel i7 based system with 4 KB pages, and each page table entry is 4-bytes long. The page table size is 256 GB long!

Will not fit in the main memory. Solution?

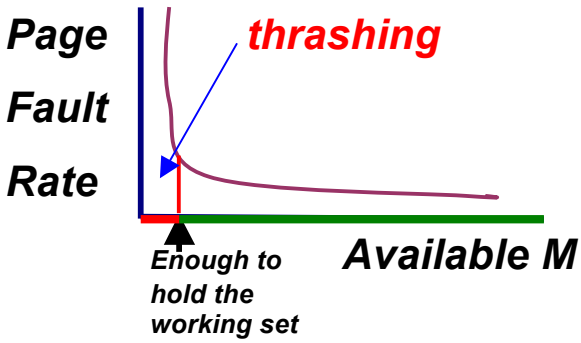


Working Set

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.