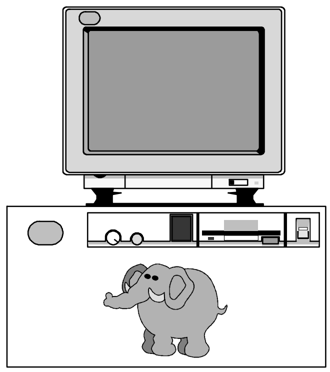
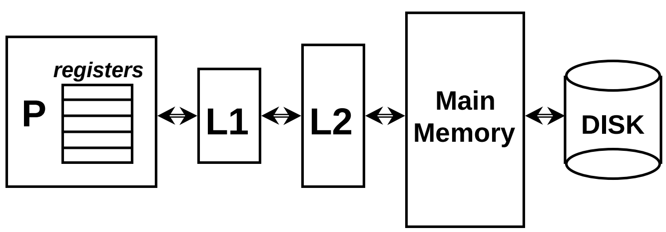
6004 Spring 1998: L38

Virtual Memory and Paging
(Pretending that your PC has a GigaByte of memory)



1

Memory Hierarchy



	P	L1	L2	Main Memory	DISK
Access time	0.5 clk	1 clk	5 clks	10-50 clks	10^5 clks
Capacity	1KB	16KB	1MB	1GB	10GB
Block Size	8B	64B	128B	4-16 KB	
Associativity	Explicitly managed (compiler)	2-way set assoc.	Direct mapped	Fully Associative	

2

Main Idea

Gee Whiz! Main memory is just a cache for the disk!

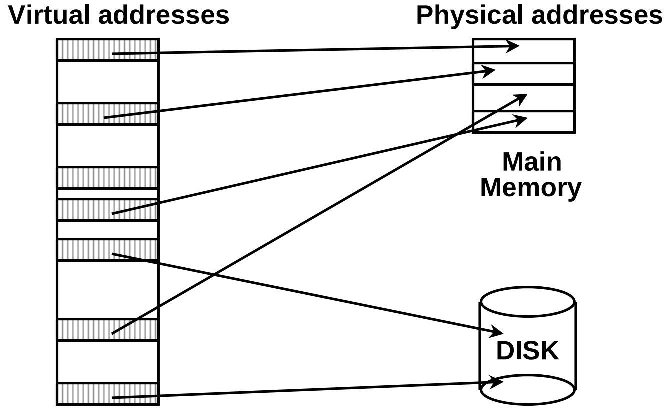
Have a large virtual address space of 2^{32} bytes.
Advantages are:

- **Removes programming burden of having a small, limited amount of main memory**
- **Efficient sharing of main memory among multiple programs which can be isolated (next lecture).**

Virtual address space is usually quite sparse, i.e., some virtual addresses are never used.

3

Virtual Memory



Virtual page may not be in physical memory, but on disk.

4

Access Time Ratios

Access time ratios for various memories:

$$\frac{L2}{L1} = 5 \qquad \frac{\text{Main Memory}}{L2} = 5$$

$$\frac{\text{Disk}}{\text{Main Memory}} = 10^4$$

Huge penalty for miss in Main Memory

5

Example

P, a 33 MIP processor gets 30% *LD* instructions. Then, P makes 10^7 load requests/second.

Assume L1 and L2 together serve 95% of the requests. Then, Main Memory serves 5×10^5 requests/second.

Assume Disk can serve 50 requests/second.

Main Memory Hit Rate	Disk Requests per second	Program Execution Time
100%	0	1.0 (normalized)
99.999%	5	________
99.99%	50	________
99.9%	50	________
90.0%	50	________

6

Anything to Increase Hit Rate…

Main Memory characteristics

Called a page

- **Block size: 4-16KB**
- **Associativity: Fully associative, implying a virtual page can be mapped to <u>any</u> physical page**
- **Replacement: LRU**
- **Write policy: Write-Back**

7

Virtual Memory Lingo

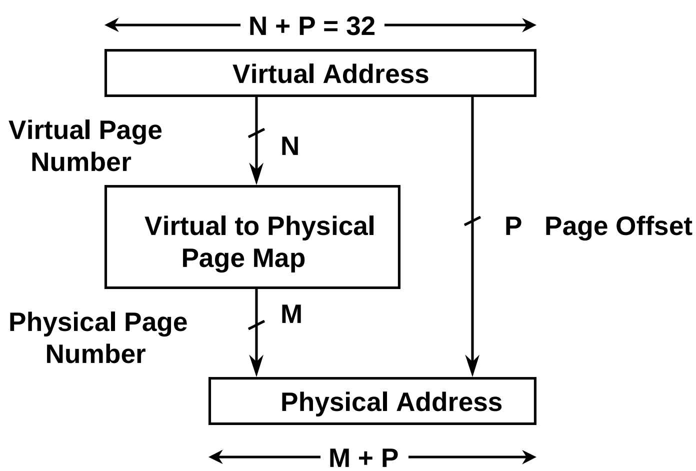
<u>Working set</u> : Number of pages required in Main Memory for a program to get an acceptable hit rate.

<u>Thrashing</u> : Loss of performance when working set size exceeds Main Memory size.

<u>Page Swap</u> : Act of bringing in new page into Main Memory and writing out victim.

8

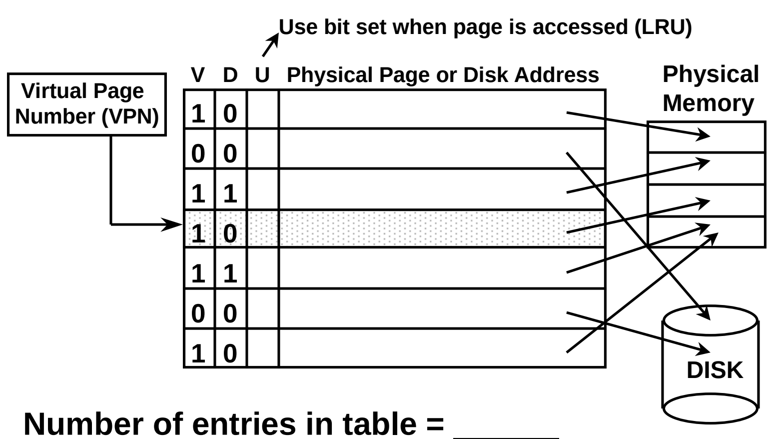
Virtual Memory Implementation



9

Virtual-to-Physical Page Map or Table

Page Table resides in Main Memory



Number of entries in table = ______

Size of table = ________

10

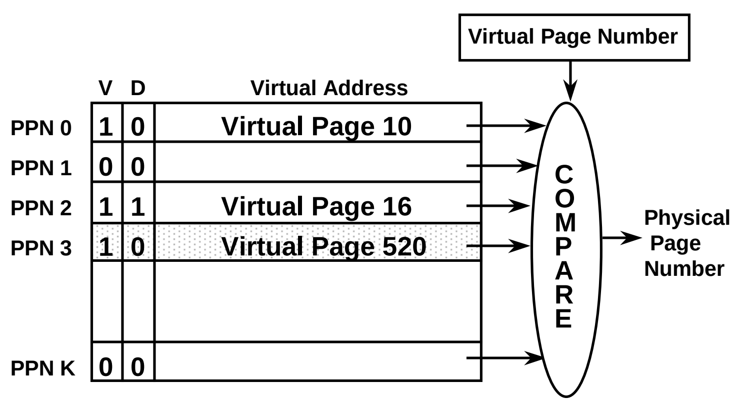
Dealing with the size of the Page Table

- **Page the Page Table: Allow parts of the page table to reside in the virtual address space. Bring parts in from disk to Main Memory when required.**

- **Inverted Page Table: Apply hashing function to the virtual address so that the page table size need only be the number of physical pages in Main Memory.**

11

Inverted Page Table



Number of entries in table K = _______

12

Performance Penalty of Virtual Memory

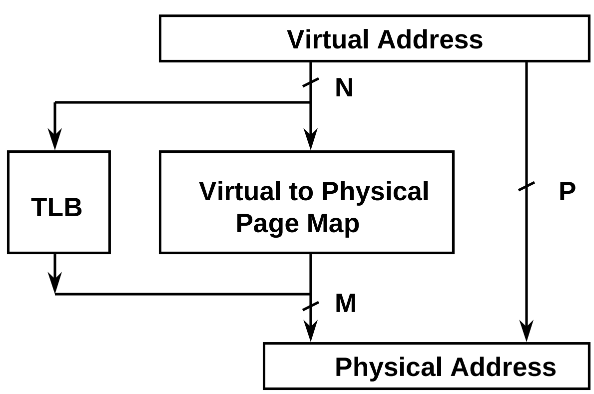
Wish to read a location in Main Memory. Steps are:

- Present a virtual address to Main Memory.
- Determine if corresponding page is in Main Memory by accessing Page Table.
- If page is in Main Memory, access Main Memory again to obtain contents of location in page.
- If page is not in Main Memory, interrupt program and call a Page Fault Handler to bring in the page from disk.
- Repeat read, this time page is in Main Memory

Even assuming 100% HIT rate in Main Memory, the penalty of using virtual memory is 2X.

13

Making the Page Table Faster

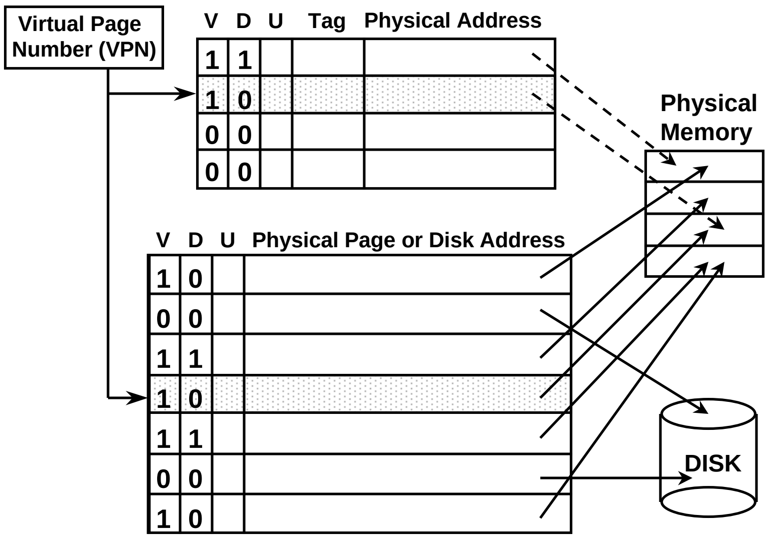


Cache a small number of Page Table entries in a fully associative Translation Lookaside Buffer (TLB)

14

6004 Spring 1998: L38

Translation Lookaside Buffer (TLB)



15

TLB Numbers

Block size: 1-2 page table entries (4 bytes each)

Hit time: 0.5 to 1 clocks

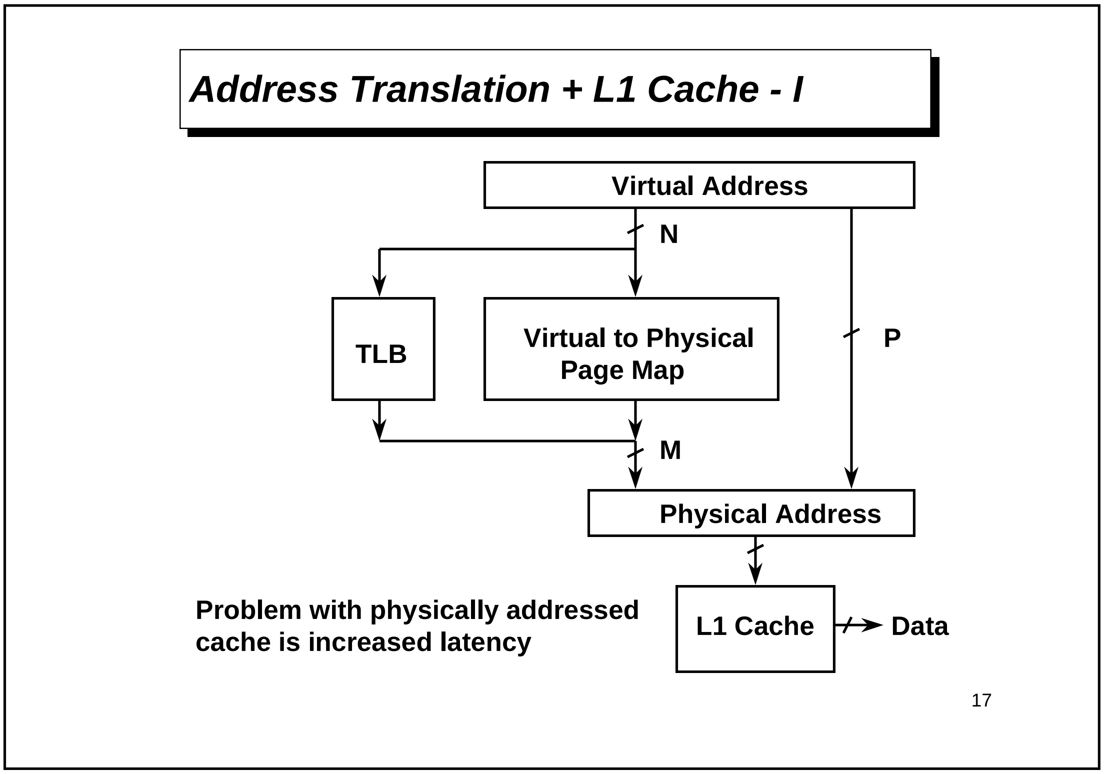
Miss penalty: 10-50 clocks

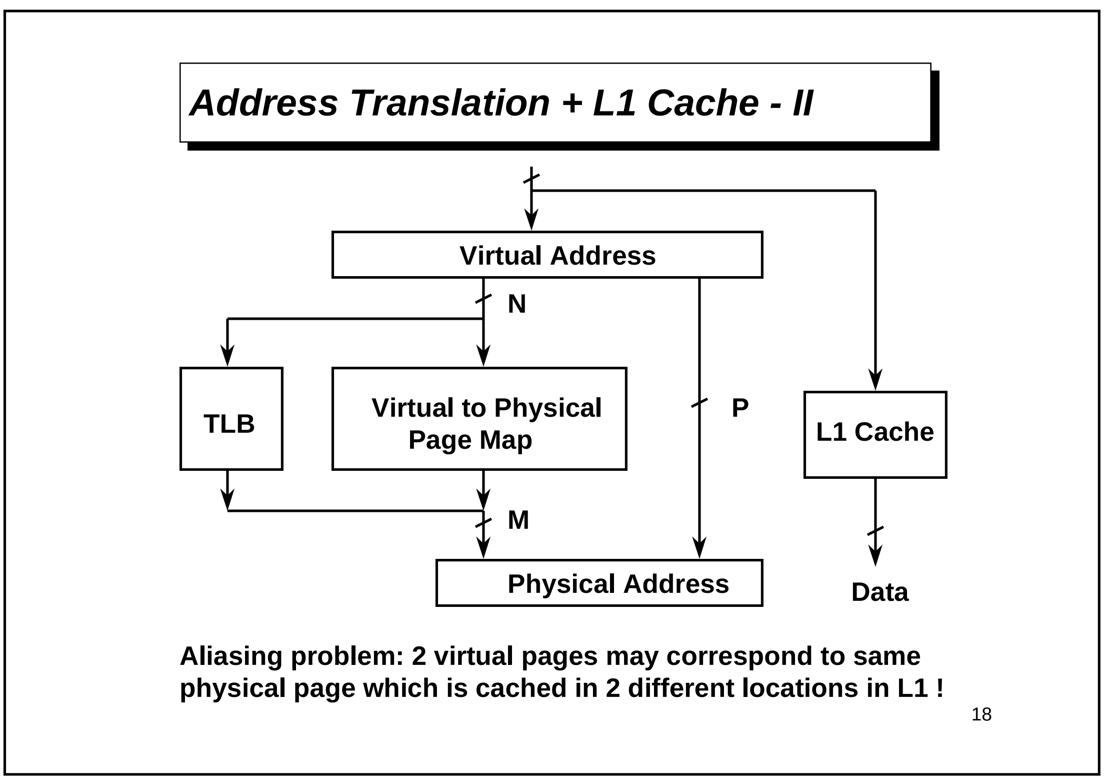
Hit rate: 99% to 99.99%

TLB size: 32-1024 entries

16

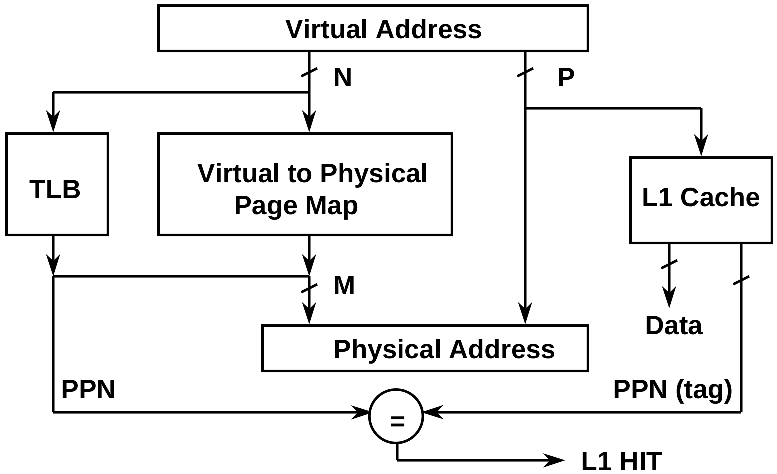
6004 Spring 1998: L38





6004 Spring 1998: L38

Address Translation + L1 Cache - III



Problem is the limitation on the size of the L1 Cache

19