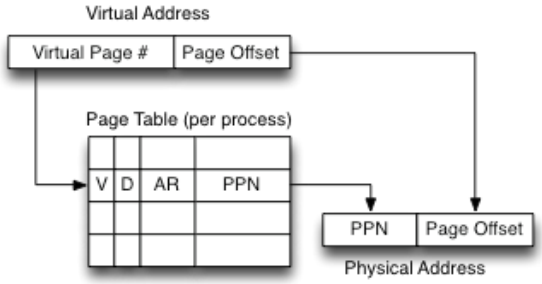
Virtual Memory

What are three specific benefits of using virtual memory? [there are *many*]

Bridges memory and disk in memory hierarchy.
Simulates full address space for each process.
Enforces protection between processes.



Virtual to Physical Address Translation

Page Table

Split memory into a bunch of equal sized chunks called pages. Better still, split a section of disk into these chunks as well. Map any virtual page into a physical page via table look up -- the page table. Each process gets its own page table.

Translation Lookaside Buffer (TLB)

A cache of page table entries. Each block is a single page table entry. If an entry (corresponding to a virtual page) is not in the TLB, it's a TLB miss. If that entry is also not in memory (it's been paged out), it's a page fault. Who gets kicked out on a page fault? What happens if there just isn't enough physical memory?
Whoever is selected according to the replacement policy. Assuming the OS knows where to find the page table on disk, we can get by with surprisingly few pages of physical memory, albeit slowly.

The TLB and the Page Table together make up a translation unit that maps from virtual addresses to physical addresses.

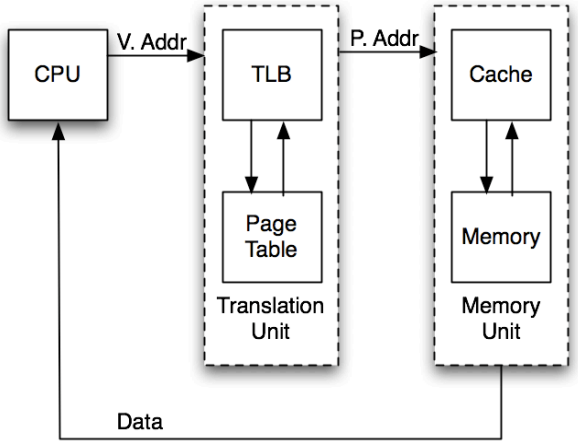


Figure 1: Logical Flow "Big Picture"

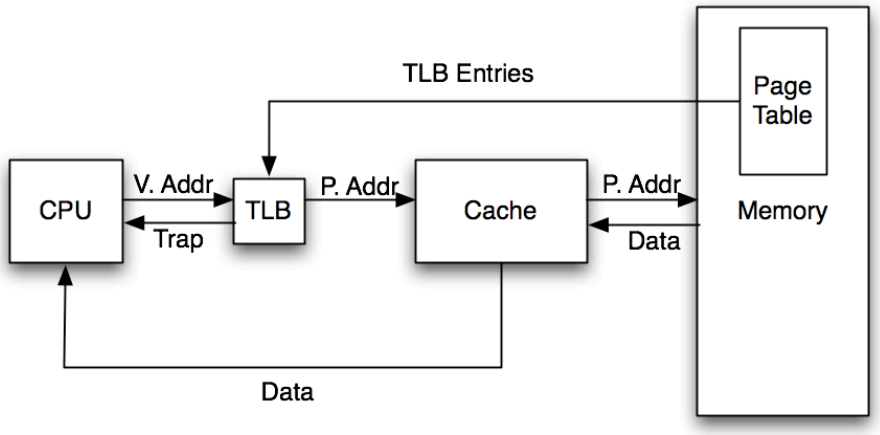


Figure 2: Physical Implementation "Big Picture"

Virtual Memory Exercises

1) Your ISA supports a 36b address space and your 2GiB RAM is broken into 2KiB pages. How many bits are in a VPN? PPN? If all the flags (valid, dirty, etc) take up 12 bits per entry, how many bits does an entire page table take up? 31 bits physical memory, 20 bit PPN, 25 bit VPN. 2^25 rows in page table, each row has 12 (flags) + 20 (PPN) bits, 2^30 bits = 2^27 bytes total. Note that this is quite a lot of space due to the large VPN – for systems like this, more clever page table schemes are used (see cs162).

2) Fill out this table!

Virtual Address Bits	Physical Address Bits	Page Size	VPN Bits	PPN Bits	Bits per row of PT (4 extra bits)
32	32	16KB	18	18	22
32	26	8KB	19	13	17
36	32	32KB	21	17	21
40	36	32KB	25	21	25
64	40	64KB	48	24	28

3) A processor has 16-bit addresses, 256 byte pages, and an 8-entry fully associative TLB with LRU replacement (the LRU field is 3 bits and encodes the order in which pages were accessed, 0 being the most recent). At some time instant, the TLB for the current process is in the initial state given in the table below. Assume that all current page table entries are in the initial TLB. Fill in the final state of the TLB according to the access pattern below.

Free physical pages: 0x17, 0x18, 0x19
Access pattern: 0x11f0 (Read), 0x1301 (Write), 0x20ae (Write), 0x2332 (Write), 0x20ff (Read), 0x3415 (Write)

Initial

VPN	PPN	Valid	Dirty	LRU	Access
0x01	0x11	1	1	0	RW
0x00	0x00	0	0	7	RW
0x10	0x13	1	1	1	RW
0x20	0x12	1	0	5	RW
0x00	0x00	0	0	7	RW
0x11	0x14	1	0	4	RW
0xae	0x15	1	1	2	RW
0xff	0x16	1	0	3	RW

For each access:
hit, LRUs: 1,7,2,5,7,0,3,4
miss w/o replacement, map VPN 0x13 to PPN 0x17, Valid/Dirty bits to 1. LRUs: 2,0,3,6,7,1,4,5
hit, LRUs: 3,1,4,0,7,2,5,6
miss w/o replacement, map VPN 0x23 to PPN 0x18, V/D bits to 1, LRUs: 4,2,5,1,0,3,6,7
hit, LRUs: 4,2,5,0,1,3,6,7
miss w/replacement, replace last element, map VPN 0x34 to PPN 0x19, D bit to 1, LRUs: 5,3,6,2,1,4,7,0

Final

VPN	PPN	Valid	Dirty	LRU	Access
0x01	0x11	1	1	5	RW
0x13	0x17	1	1	3	RW
0x10	0x13	1	1	6	RW
0x20	0x12	1	1	2	RW
0x23	0x18	1	1	1	RW
0x11	0x14	1	0	4	RW
0xae	0x15	1	1	7	RW
0x34	0x19	1	1	0	RW