Virtual Memory Overview

Virtual address (VA): What your program uses

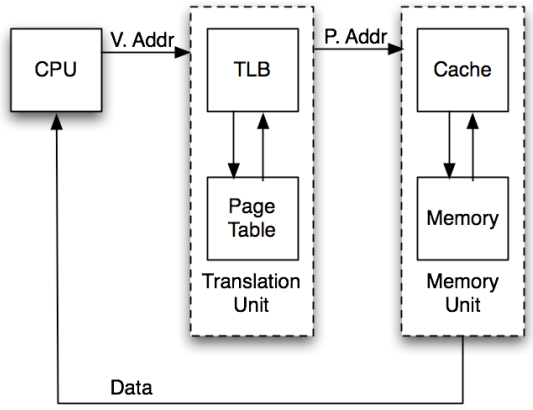
Virtual Page Number	Page Offset

Physical address (PA): What actually determines where in memory to go

Physical Page Number	Page Offset

With 4 KiB pages and byte addresses, 2^(page offset bits) = 4096, so page offset bits = 12.

The Big Picture: Logical Flow
Translate VA to PA using the TLB and Page Table. Then use PA to access memory as the program intended.



Pages
A chunk of memory or disk with a set size. Addresses in the same virtual page get mapped to addresses in the same physical page. The page table determines the mapping.

The Page Table

Index = Virtual Page Number (VPN) (not stored)	**Page Valid**	**Page Dirty**	**Permission Bits (read, write, ...)**	**Physical Page Number (PPN)**
0				
1				
2				
…				
(Max virtual page number)				

Each stored row of the page table is called a **page table entry** (the grayed section is the first page table entry). The page table is stored *in memory*; the OS sets a register telling the hardware the address of the first entry of the page table. The processor updates the "page dirty" in the page table: "page dirty" bits are used by the OS to know whether updating a page on disk is necessary. Each process gets its own page table.

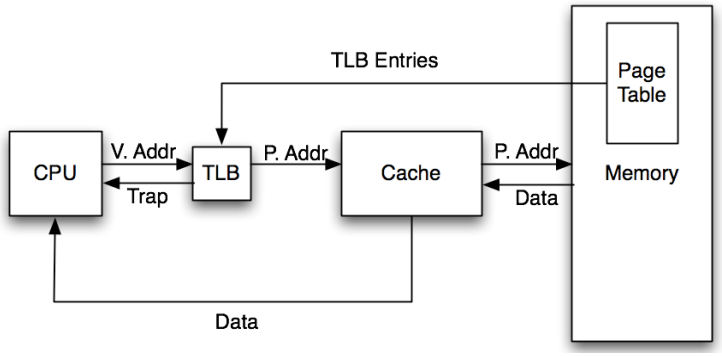
- **Protection Fault**--The page table entry for a virtual page has permission bits that prohibit the requested operation
- **Page Fault**--The page table entry for a virtual page has its valid bit set to false. The entry is not in memory.

The Translation Lookaside Buffer (TLB)

A cache for the page table. Each block is a single page table entry. If an entry is not in the TLB, it's a TLB miss. Assuming *fully associative*:

TLB Entry Valid	Tag = Virtual Page Number	Page Table Entry		
		Page Dirty	Permission Bits	Physical Page Number
…	…	…	…	…

The Big Picture Revisited



Exercises

1) What are three specific benefits of using virtual memory?

Program isolation; avoiding fragging; not dependent on physical memory location?
Bridges memory and disk in memory...?

2) What should happen to the TLB when a new value is loaded into the page table address register? A new process has been started, so the TLB should be marked as dirty. It will then fetch the correct addresses.

3) Fill in the following formulas below.

#Offset Bits = $\log_2$(Page size in bytes)

#Virtual Address Bits = #(Virtual Page number bits) + #(Page Offset bits)

#Physical Address Bits = #(Physical Page Number Bits) + #(Page offset bits)

#Bits per row of Page Table = #(Tag bits physical page number) + #(Extra bits [valid, dirty bits, etc.])

4) Using the formulas from part 3, fill this table out!

Virtual Address Bits	Physical Address Bits	Page Size	VPN Bits	PPN Bits	Bits per row of PT (4 extra bits)
32	32	16KiB	18	18	14 22
32	26	8KiB	19	13	17
36	32	32KiB	21	17	21
40	36	32KiB	25	21	25
64	40	64KiB	48	24	28

5) 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 the initial state given in the table below. Assume that all current page table entries are in the initial TLB. Assume also that all pages can be read from and written to. Fill in the final state of the TLB according to the access pattern below.

Free physical pages: `0x17, 0x18, 0x19`

Access pattern:

Read	0x11f0
Write	0x1301
Write	0x20ae
Write	0x2332
Read	0x20ff
Write	0x3415

Initial TLB

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

Final TLB

VPN	PPN	Valid	Dirty	LRU
01	11	1	1	5
13	17	1	1	3
10	13	1	1	6
20	12	1	1	1
23	18	1	1	2
11	14	1	0	4
ac	15	1	1	7
34	19	1	1	0