Translation Lookaside Buffers (TLBs)

Glenn Bruns

CSUMB

Motivation

Base-and-bounds addressing was fast.

Paging looks slow:

- extract the VPN from the virtual address
- index into a page table in memory to translate VPN to PFN
- glue the PFN and offset together

Address translation needs to be super-fast!
About 1 clock cycle fast.

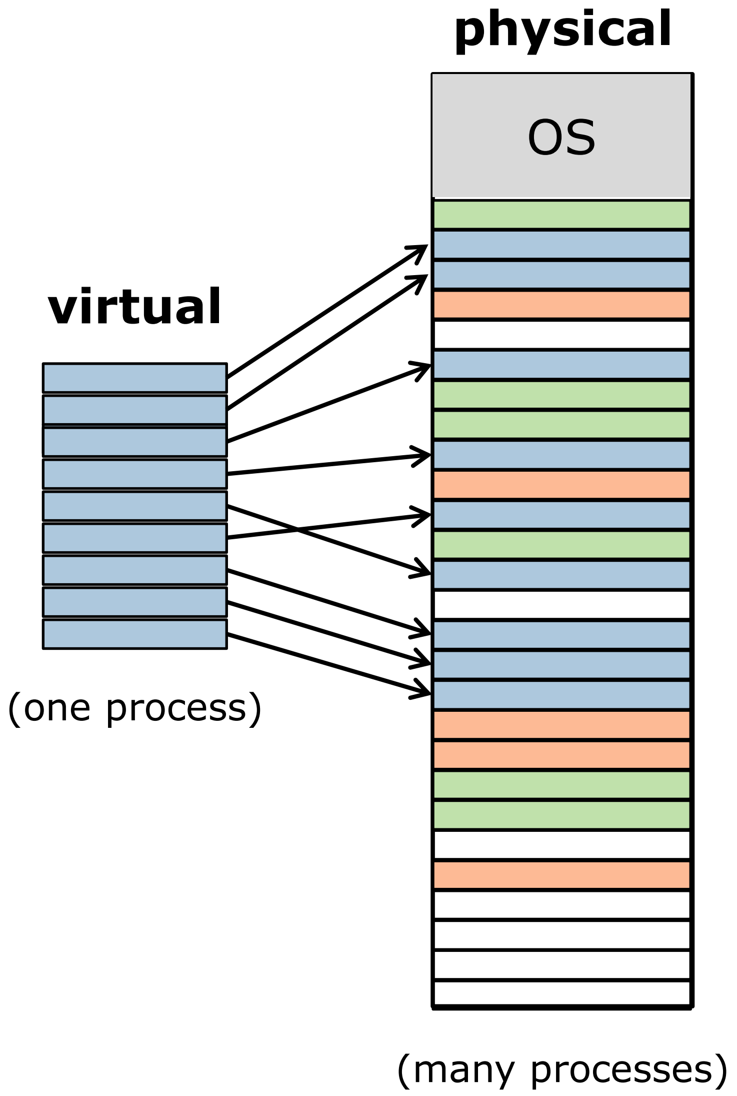
Learning outcomes

After this lecture, you should be able to:

- Explain the design idea used to get good paging speed
- Explain the ideas of “temporal locality” and “spatial locality”
- Calculate the average time it take to access memory with paging

Recap: Paging

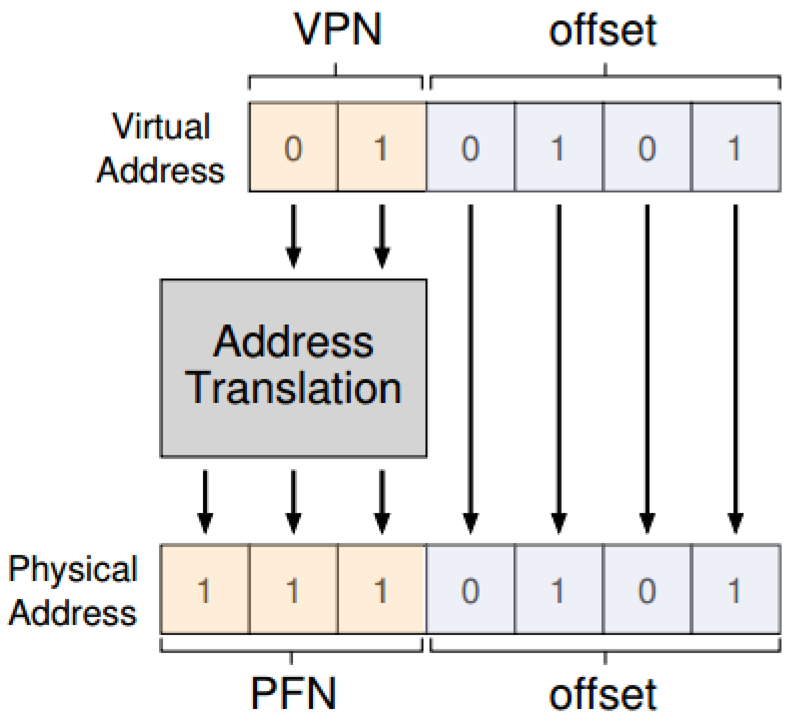
- Good news:
  - avoids fragmentation
  - simple, flexible
- Bad news:
  - page table can be large
  - speed; CPU registers can't be used for address translation



Address translation with a page table

page table

virtual page number	**page frame number**
00	011
01	111
10	101
11	010



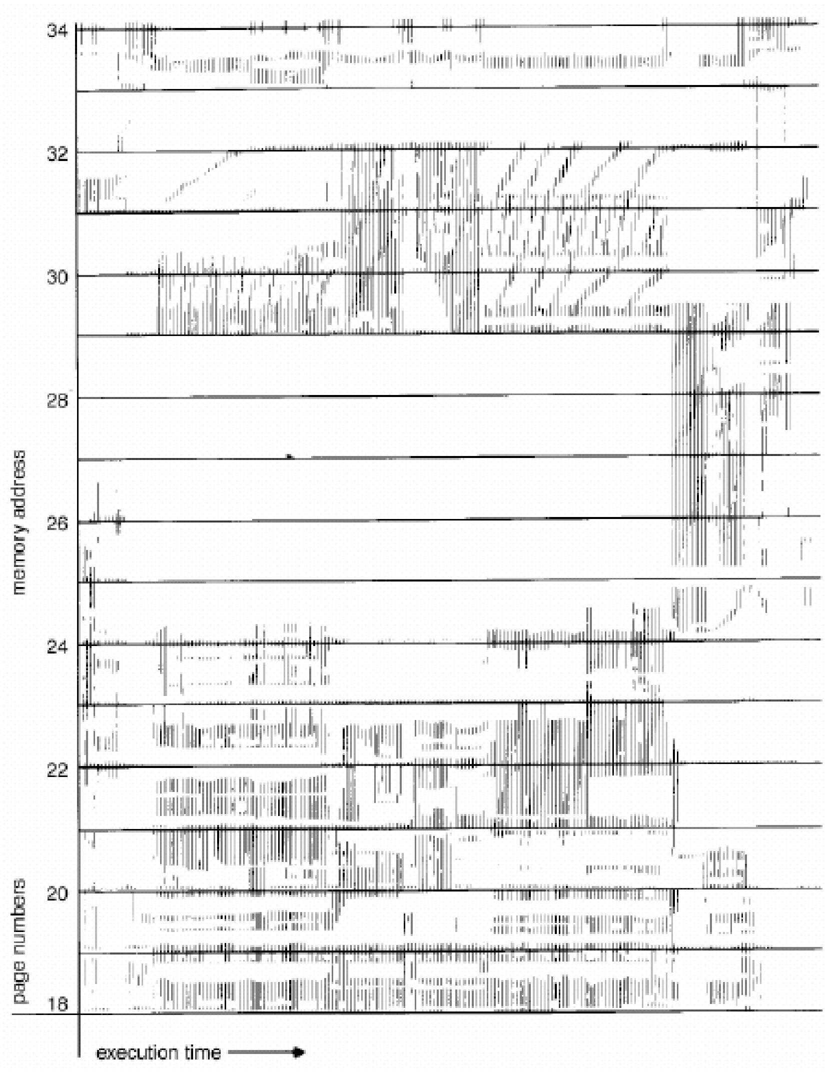
With a 32 bit address space and 4 KB pages, a page table will have about one million entries.

How to make paging fast?

Idea: cache the page table

- If a virtual page number was recently looked up, remember its page frame number
- If that page number needs to be translated again, no need to use the page table

Pattern of program page usage



In typical programs, an address is more likely to be accessed if:

1. it was accessed in the recent past, or
2. a nearby address was accessed in the recent past.

Using a TLB in address translation

VPN part of virtual address	hit?	TLB (a cache)	
01	no	[]	get PFN from table
11	no	[(01, 111)]	get PFN from table
11	yes	[(01, 111), (11, 010)]	get PFN from cache

page table

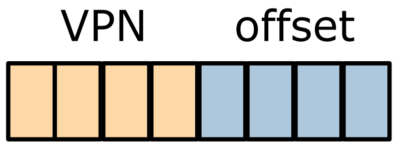
VPN	page frame number
00	011
01	111
10	101
11	010

Address translation with a TLB

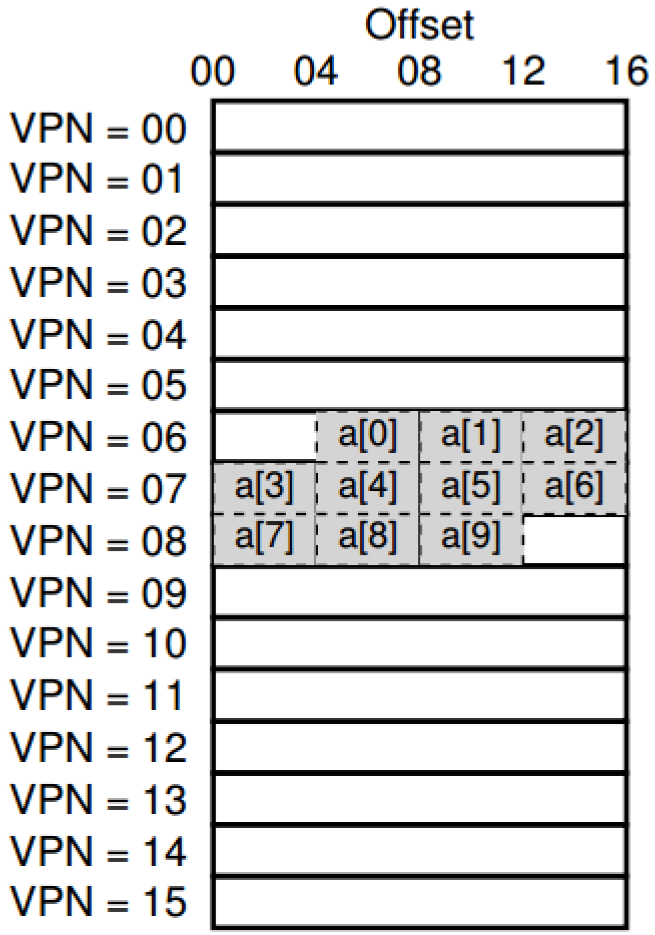
see text for gory details

```
VPN = VPN part of virtual address
if (tlb_lookup(VPN) is successful)
    tlb_entry = TLB entry for VPN
    if (tlb_entry.protect_bits don’t allow access)
        raise PROTECTION_FAULT
    else
        offset = offset part of virtual address
        combine tlb_entry.PFN and offset into physical address
        put value at the physical address into a register
else
    index into the page table using the VPN
    PTE = page table entry
    if (PTE.valid is 0)
        raise SEGMENTATION_FAULT
    if (PTE.protect_bits don’t allow access)
        raise PROTECTION_FAULT
    tlb_insert(VPN, PTE.PFN, PTE.protect_bits)
    retry instruction
```

Example: Array access



virtual address:



```
int sum = 0;
for (i = 0; i < 10; i++) {
    sum += a[i];
}
```

$\left.\begin{array}{l}\text{read a[0] – cache miss}\\ \text{read a[1] – cache hit}\\ \text{read a[2] – cache hit}\end{array}\right\}$ VPN = 6

read a[3] – cache miss

read a[4] – cache hit

read a[5] – cache hit

read a[6] – cache hit

…

(code/figure from Operating Systems: Three Easy Pieces)

TLB and context switching

- The TLB is a hardware cache
- **Exercise**: What to do with contents of TLB when a context switch occurs?

Options:
- Clear ("flush") the TLB
- Mark TLB entries with an identifier for each process

VPN	PFN	valid	prot	ASID
10	100	1	rwx	1
----	----	0	----	----
10	170	1	rwx	2
----	----	0	----	----

MMU – Memory Management Unit

The TLB is hardware. Where does it live?

In the MMU, which is usually incorporated into the CPU.

TLB lookups need to be very fast -- the TLB is often implemented in hardware as an associative cache.

If the case of a TLB miss, a page table in memory (not the MMU) is used.

Typical TLB performance

- size: 12 - 4096 entries
- hit time: about 1 clock cycle
- miss penalty: 10-100 clock cycles
- miss rate: 0.01 – 1%

What is the average memory access time if:
- hit takes 1 clock cycle
- miss takes 30 clock cycles
- miss rate is 1%

?

(data from Patterson & Hennessy: Computer Organization and Design, 2009, via Wikipedia)

Summary

- Performance of paging can be increased by using a "Translation Lookaside Buffer" (TLB)
- A TLB is simply a hardware cache for the page table
- Huge performance improvements with the TLB, but only with programs that show "locality" in addressing memory