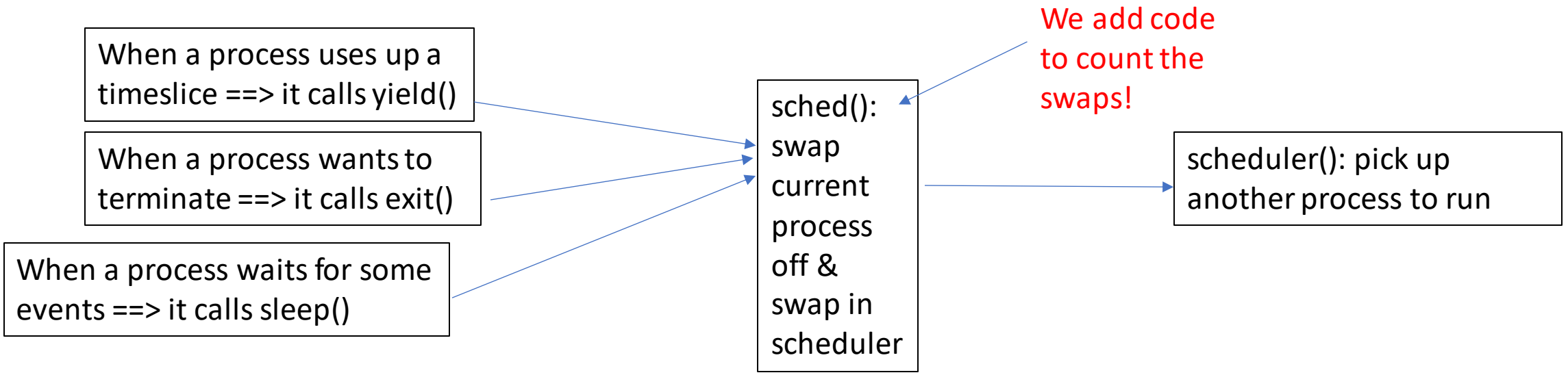
Project 1.B: add system calls to xv6-riscv

Kernel side:

- kernel/syscall.h: declare unique number for each new system call
- kernel/syscall.c: declare functions and fill the syscall table
- kernel/proc.h, kernel/proc.c: implementation of the functions of system calls



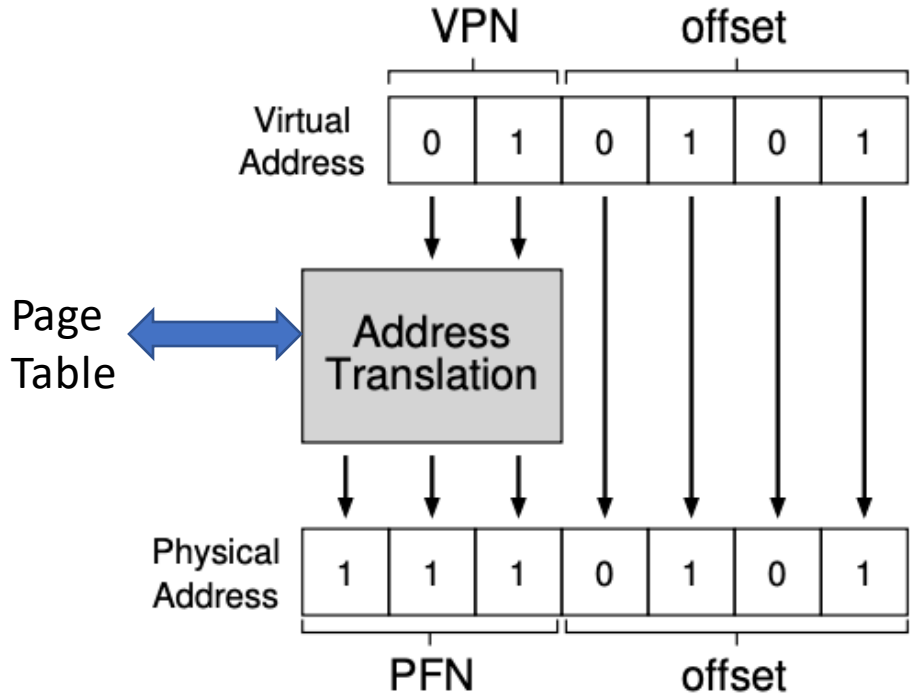
User side:

- user/usys.pl: to generate necessary assembly code
- user/user.h: declare functions (signatures) used at user side
- user/XXX.c: develop testing code

Recap

Paging

- Address space and Physical memory are divided into equal-size units: pages and frames
- A page table (mapping from VPNs to PFNs) for each process
- Translation:



L10: Translation Look-aside Buffer (TLB)

(Based on Ch 19)

Slides revised from
Matthew Tancreti
Iowa State University

Limitation of Paging

Page table is stored in main memory

Every memory access requires additional read to look up page table entry - unacceptable overhead

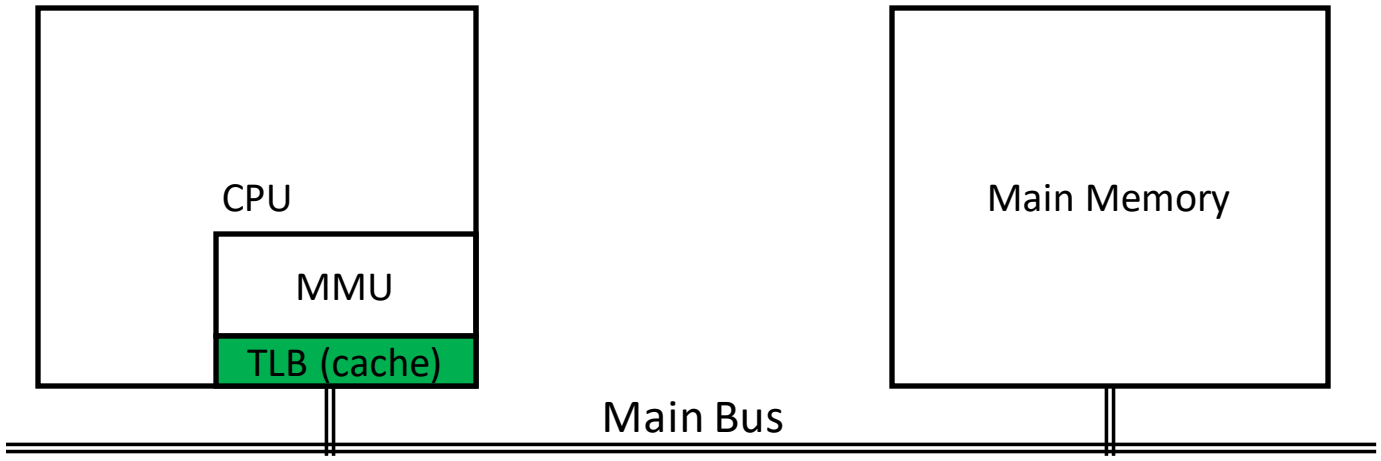
How to speed up address translation?

Basic Idea: TLB is Cache of Page Table Entries

Store a cache of “popular” page table entries near to MMU hardware

- Called **Translation-Lookaside Buffer (TLB)**

Cache can be accessed in single CPU cycle (main memory >100 cycles!)

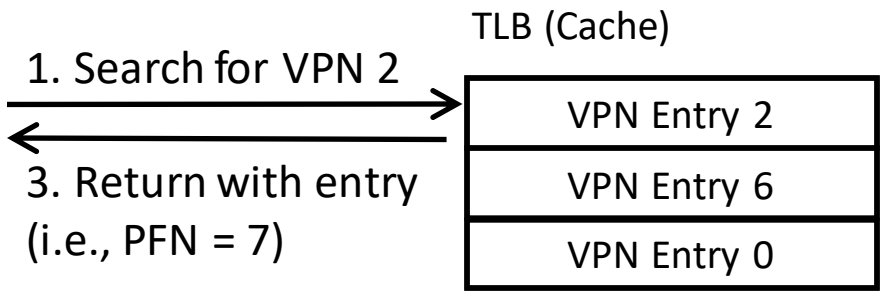


TLB Cache

On each memory access

- Look for VPN in TLB cache
- If found (**TLB hit**) return entry
- Else (**TLB miss**) get entry from page table in main memory
- Store entry to cache (possibly replacing some other entry)

Example: assume program reads from page 2



Page Table (Memory)

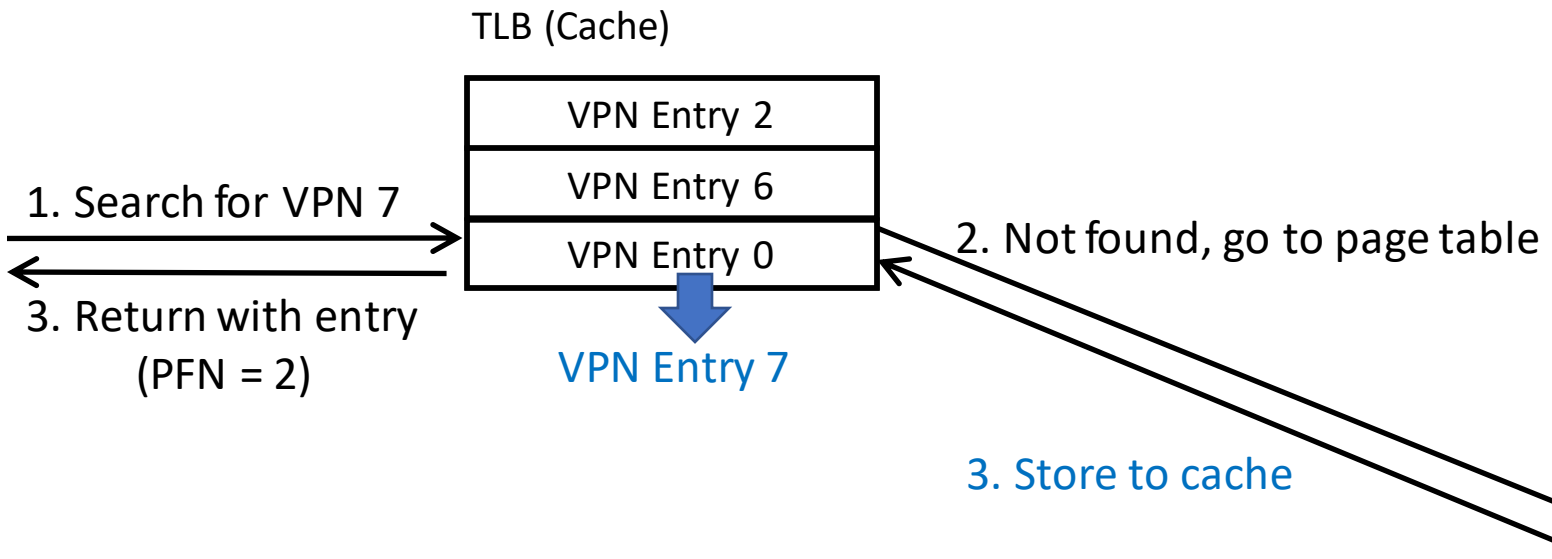
VPN	PFN	valid
0	3	1
1	5	1
2	7	1
3	-	0
4	-	0
5	-	0
6	4	1
7	2	1

TLB Cache

On each memory access

- Look for VPN in TLB cache
- If found (**TLB hit**) return entry
- Else (**TLB miss**) get entry from page table in main memory
- Store entry to cache (possibly replacing some other entry)

Example: assume program reads from page 7



Page Table (Memory)

VPN	PFN	valid
0	3	1
1	5	1
2	7	1
3	-	0
4	-	0
5	-	0
6	4	1
7	2	1

TLB Cache Entry

The basic fields of an entry are the VPN, PFN and a bit that indicates whether entry is valid

Note: valid bit in cache is not the same as valid bit in page table; it indicates if the cache entry is valid.

VPN	PFN	Valid
10	100	1
-	-	0
11	170	1
-	-	0

Why does Cache Work?

If all memory accesses were random, cache would be useless

By observation of real programs we see patterns:

- **Spatial locality** - address of memory access is likely to be close to the previous access
- **Temporal locality** – addresses likely to repeat in time

Why Does Spatial Locality Exist?

Sequential memory access is common

Example 1:

Program instructions are read sequentially from memory

Example 2:

A common task – iterating though array

```
for (int i=0; i<100; i++) {
        sum += a[i];
}
```

Why Does Temporal Locality Exist?

Programs loop… a lot

Performance

Recall that a memory access is 100s of times slower that a cache lookup

Therefore, we really want to avoid cache misses

Important performance considerations

- What is in hardware and what is in software?
- How to decide what to keep in cache?
- What happens when there is a context switch?

Hardware vs Software

The TLB cache lookup must be be in hardware

Who handles a TLB Miss?

- On some systems the answer is hardware
- On other systems TLB miss results in trap and OS takes over

How can it be acceptable to handle miss in software?

- If system is working well, misses should be very rare, its ok for rare events to be slow, priority is making common events fast

Replacement Policy

To add a new TLB cache entry when the cache is full, we need to replace an old one, how to pick which to replace?

- **Least-recently-used (LRU)** - replace the entry that was used the longest ago
- **Random** – replace an entry at random

We will cover replacement policies in detail when we investigate swap

Context Switches

Each process has its own page table

What happens to TLB cache on context switch?

- Simple solution is flush cache
- Alternative is to add **address space identifier (ASID)** to track which page table entry is from

ASID has the advantage that we might return to a previous process before all entries replaced

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

Context switches greatly increase chance of misses, another reason to avoid frequent context switches