4/20

Last Class

1. Pages
2. Design a VM
3. Page Table
4. Locality

I VM as a caching tool

II VM for Memory Protection

A proces should not → don't access to other proceses' memory

- modify any OS pages.
- modify its read only content (.text)

Page table helps us with this

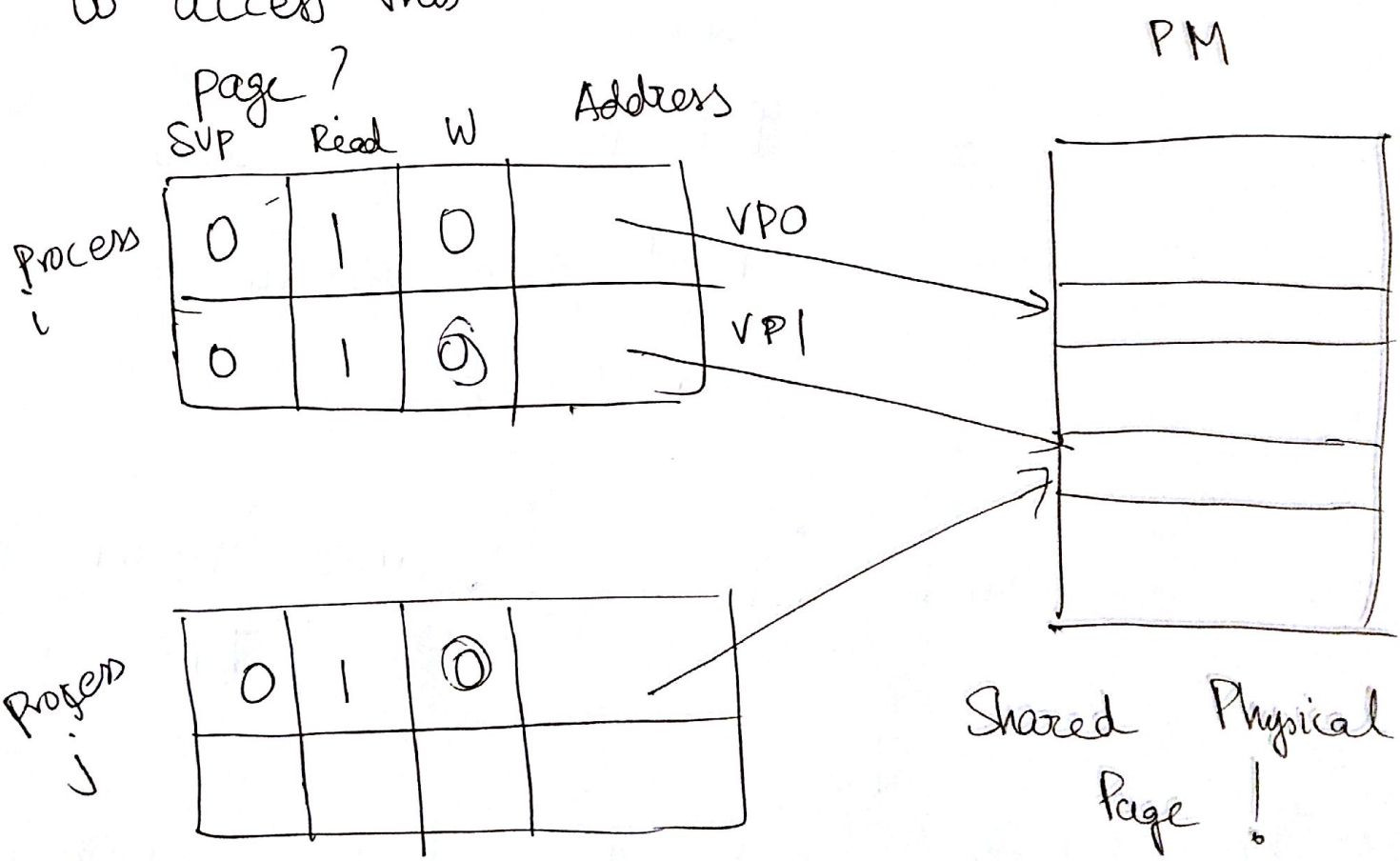
3 additional bits.

SUP bit / Read bit / Write bit

Should the process be running in supervisor mode to access this page?

Read bit: 0 → cannot read; 1 → can read

Write bit: 0 → cannot write; 1 → can write



Eg. Shared Libraries. (Printf)

III VM for Memory Management

Simplifies → Linking
→ Sharing
→ Allocation.

Address Translation.

Speeding up Address Translation

MMU —read→ PTE from the Physical Memory.

~~Worst Case~~

Worst Case → PT not in the cache. Read PM. Slowwww..

Best Case → PT is in the cache. Read from cache. Slowww. //.

Have a small cache → for small number of PTEs

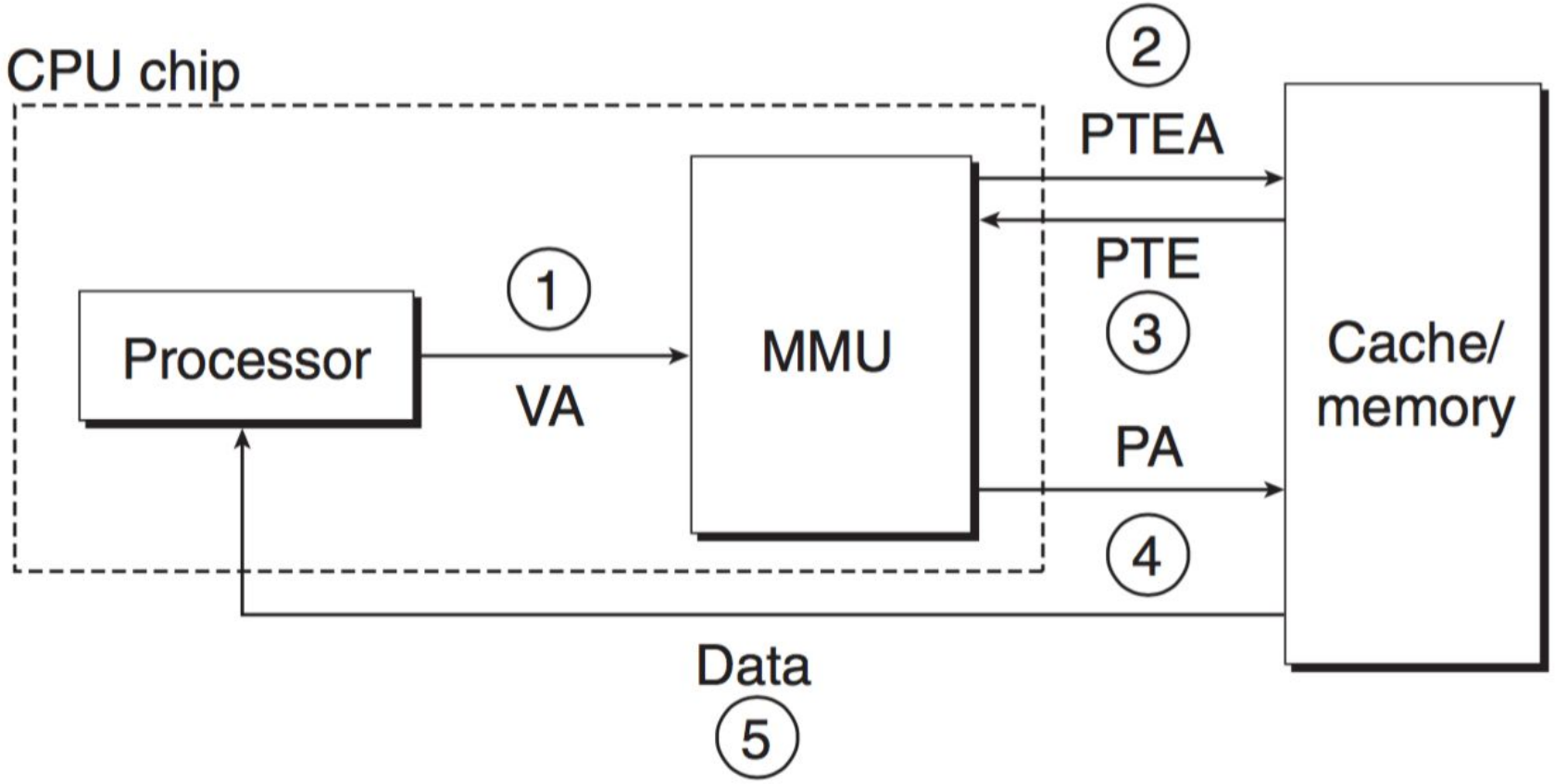
↓

Translation Lookaside Buffer (TLB)

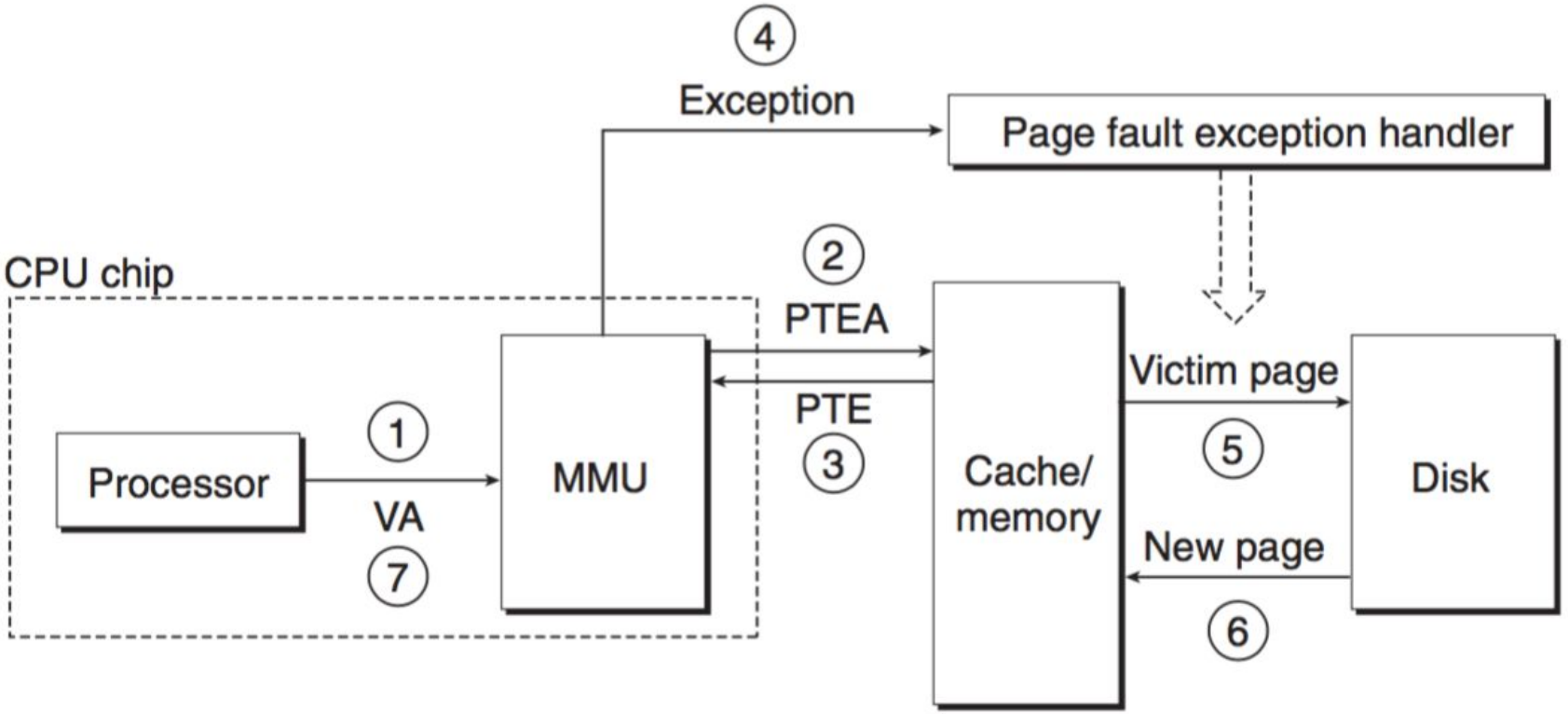
Address Translation

Ganesh Kumar . April 20, 2016

Page Hit (Operational View)



Page Fault (Operational View)



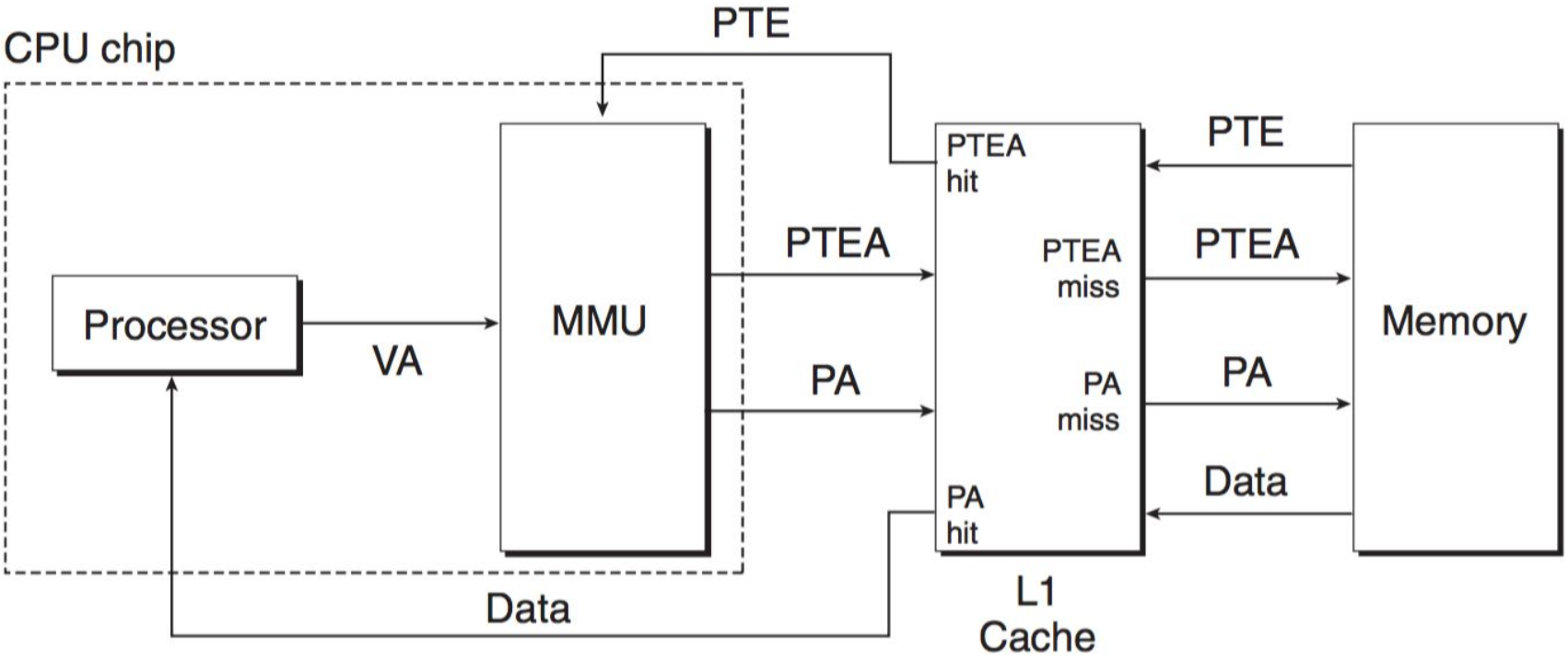
Cache and Main Memory

- They are distinct entities (duh)
- Caches also use only physical addresses (in most implementations)
- Cache Hits can still happen
- Cache Misses can also happen

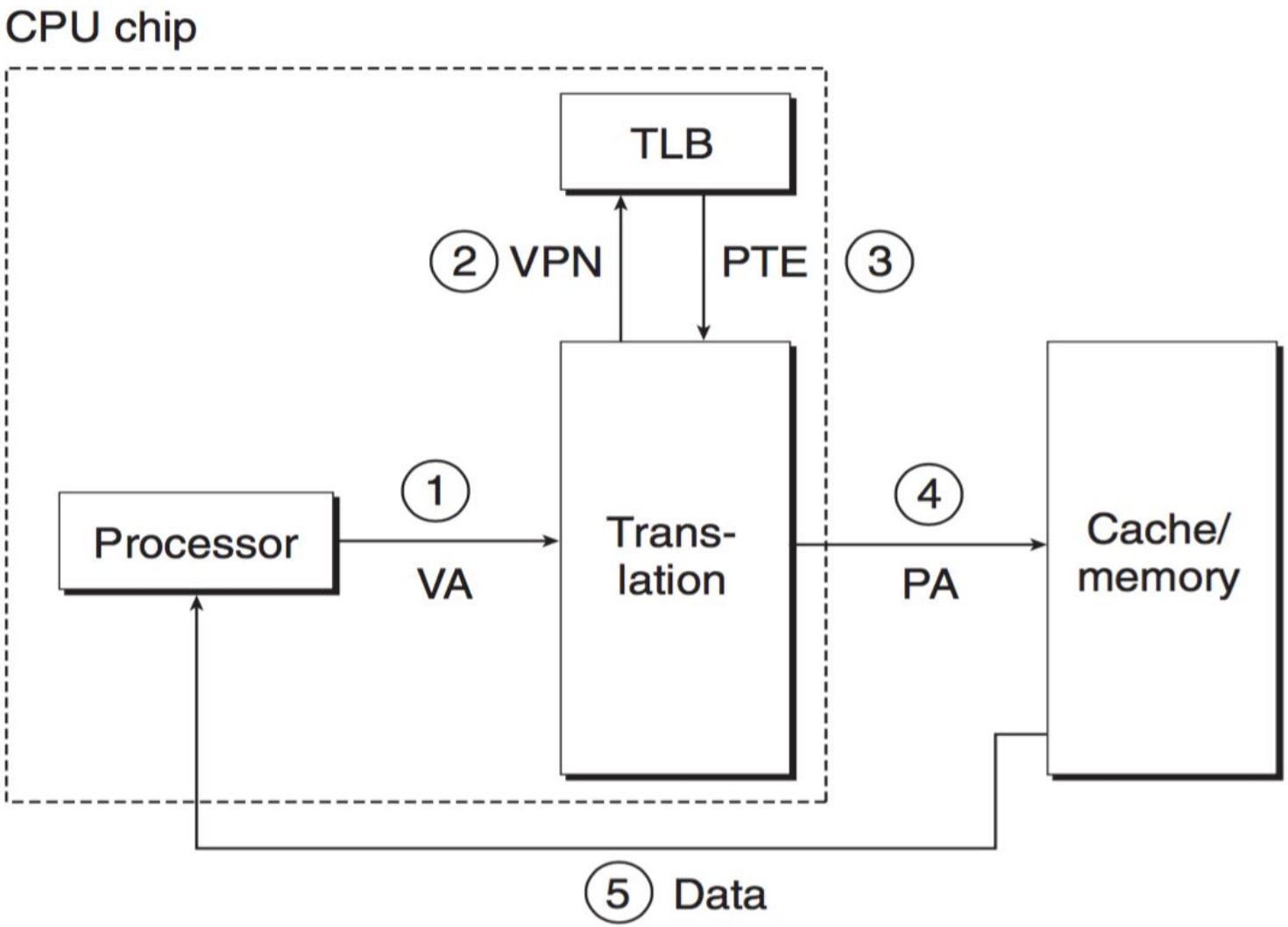
Example: Requesting a page that is in the Main Memory but not in the Cache?

- Cache miss!
- Load the block(s) onto the cache.
- Now pass along the value requested by the MMU or the CPU.

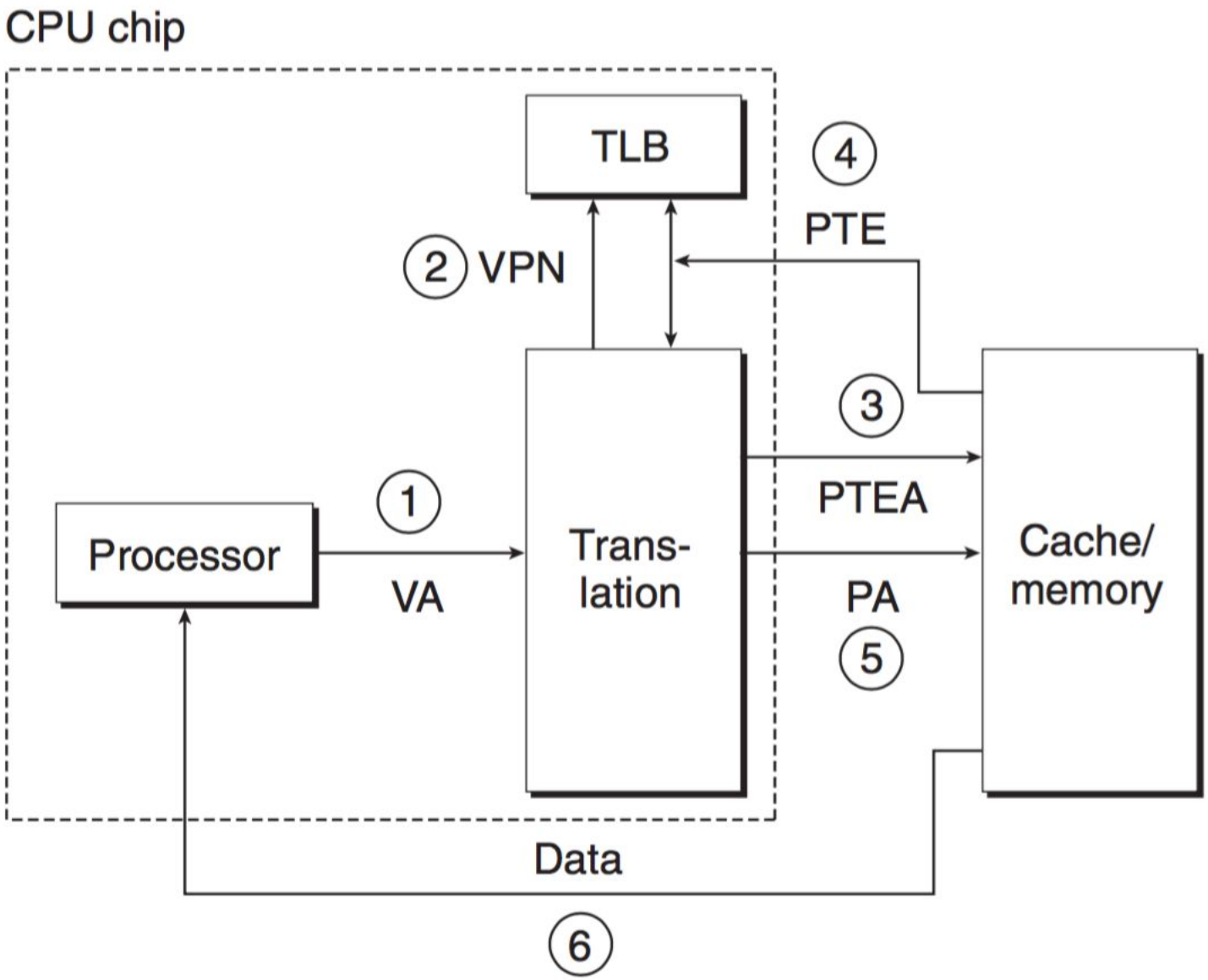
VM with Cache



TLB Hit


CPU chip
TLB
2
VPN
PTE
3
Processor
1
VA
Trans-
lation
4
PA
Cache/
memory
5
Data

TLB Miss


CPU chip
TLB
4
PTE
2 VPN
3
PTEA
1
Processor
VA
Trans-
lation
Cache/
memory
PA
5
Data
6

Worksheet!

TLB Tag bits

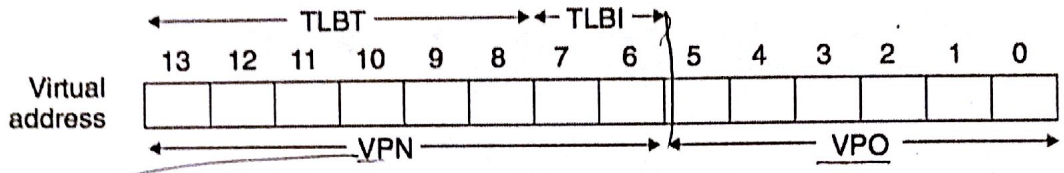
In-Class Problem Walkthrough

Will be solved In class on April 20, 2016

TLB Index bits.

Consider the following memory system snapshot,

Figure A. TLB: Four sets, 16 entries, four-way set associative



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Virtual Page Number

PPN - Physical Page Number

Figure B. Page table: Only the first 16 PTEs are shown

VPN	PPN	Valid
00	28	1
01	—	0
02	33	1
03	02	1
04	—	0
05	16	1
06	—	0
07	—	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

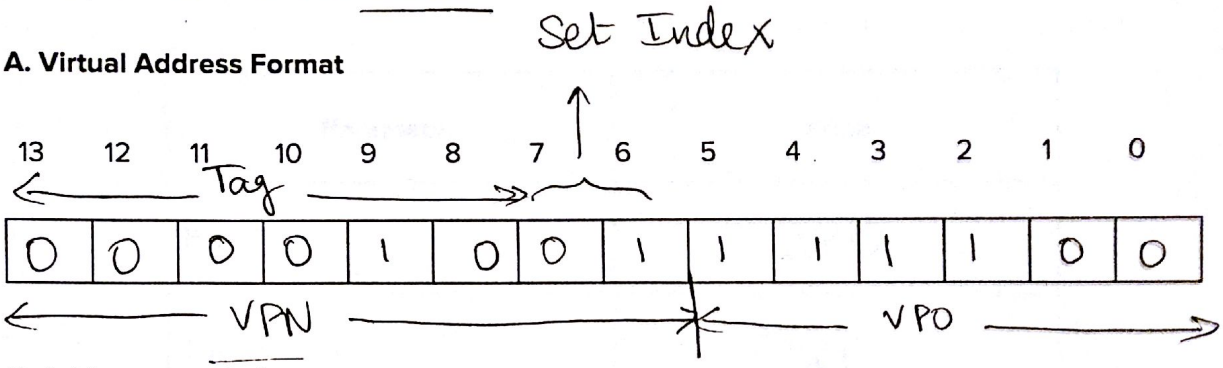
→ 0

TLB entry will be updated

* | 02 | 17 | 1 |

Given Virtual Address: **0x027c**

A. Virtual Address Format

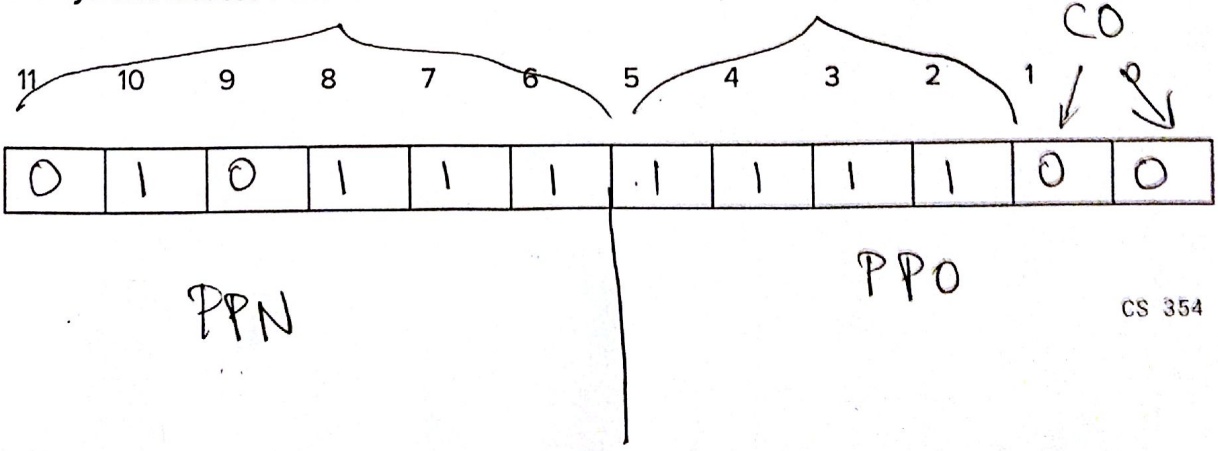


B. Address translation

Parameter	Value
VPN	0x09
TLB Index	0x1
TLB tag	0x2
TLB Hit? (Y/N)	No
Page Fault? (Y/N)	No
PPN	0x17

If the valid bit in the PTE was not set
⇒ Page Fault

C. Physical Address Format



D. Physical Memory Reference

Parameter	Value
Byte Offset	0x0
Cache Index	0xF
Cache tag	0x17
Cache Hit? (Y/N)	No.
Cache byte returned	—

Byte Addressable

TLB → tag, set and block offset.

Each block → PTE

VPN → Virtual Page Number (tag + set).

So far → PPN = 0x17

How do we get the Physical Address ?

PA → PPN + PPO

PA → PPN + VPO