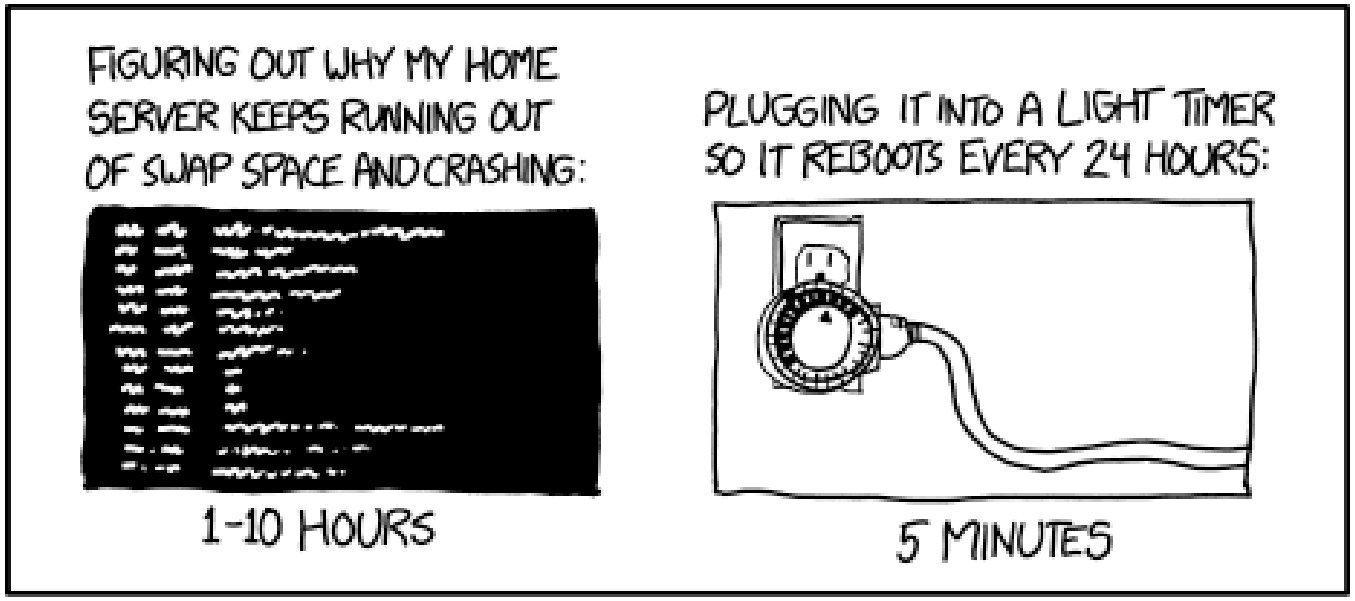
Virtual Memory II

CSE 351 Winter 2024

Instructor:
Justin Hsia

Teaching Assistants:
Adithi Raghavan
Aman Mohammed
Connie Chen
Eyoel Gebre
Jiawei Huang
Malak Zaki
Naama Amiel
Nathan Khuat
Nikolas McNamee
Pedro Amarante
Will Robertson

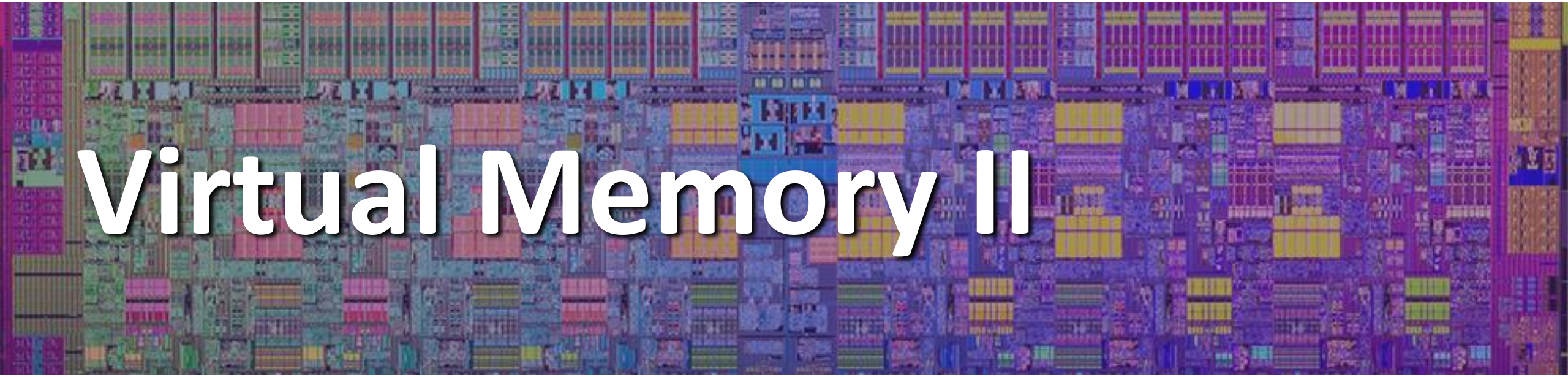


https://xkcd.com/1495/

Relevant Course Information

- HW22 due tonight, HW23 due Wednesday, HW24 due Friday
- Today is the last day to submit Lab 4
- Lab 5 due Friday (3/8)
  - The most significant amount of C programming you will do in this class – combines lots of topics from this class: pointers, bit manipulation, structs, examining memory
  - Understanding the concepts *first* and efficient *debugging* will save you lots of time
  - Light style grading

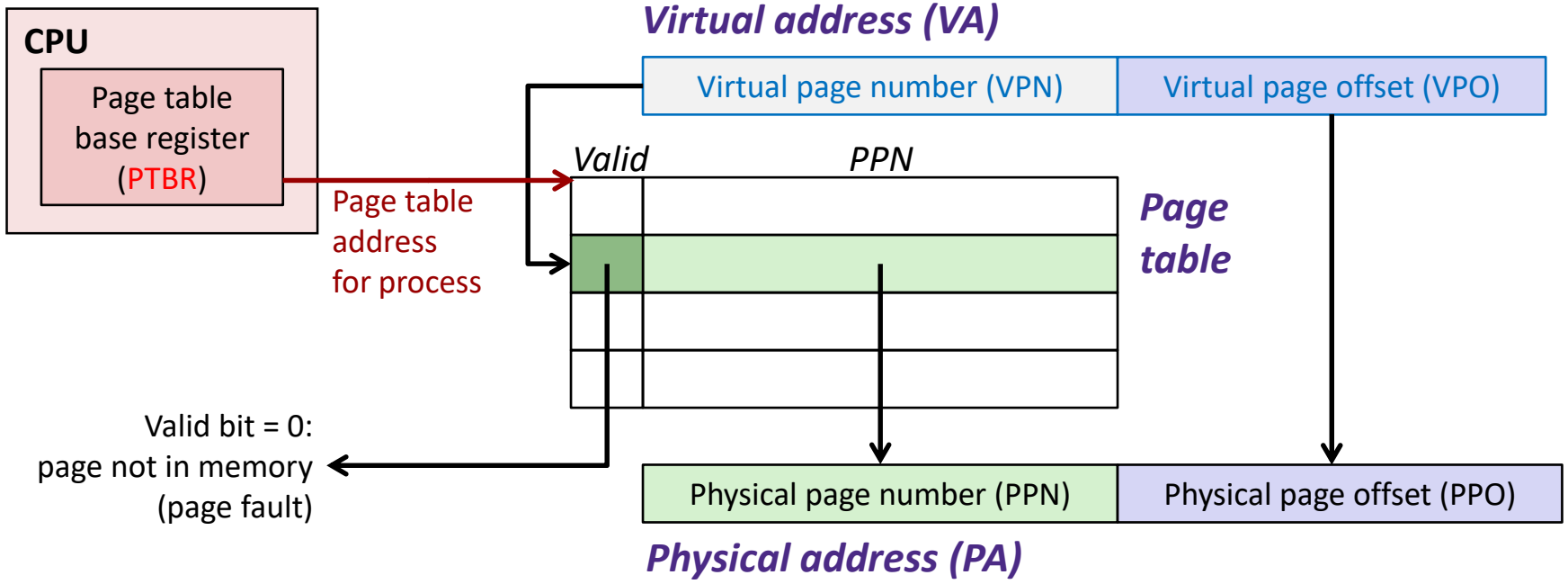
- No lessons for last two lectures – "normal" lectures
- Final exam: 3/11-13
  - Final review session on 3/8 @ 4:30 pm in CSE2 G01 and on Zoom



Virtual Memory II

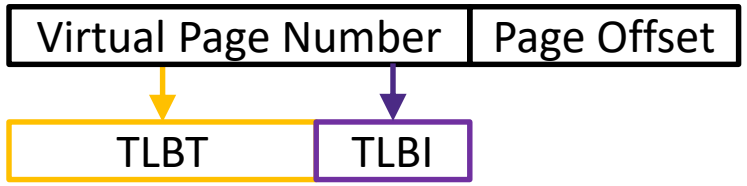
Lesson Summary (1/2)

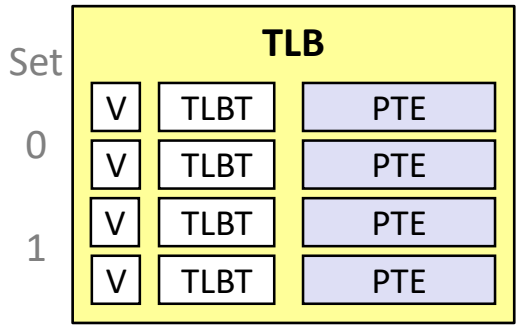
- Address translation done via ***page tables***
  - Lookup tables (one per process) that map VPN → PPN
  - Uses management bits: valid bit, access rights (read, write, execute)
  - Stored in memory – page table for currently-running process is pointed to by ***page table base register*** (PTBR)



Lesson Summary (2/2)

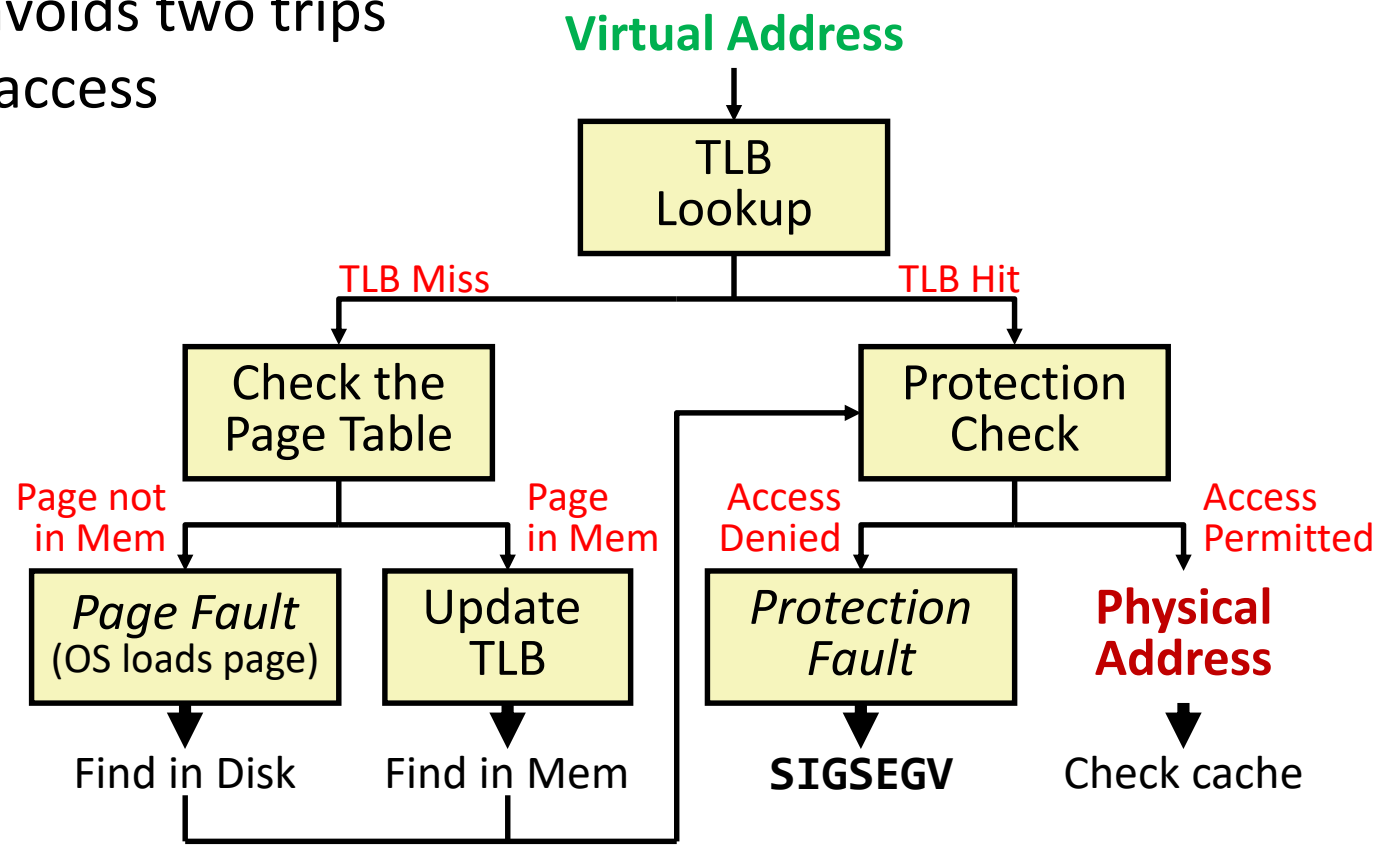
- Introduced the ***translation lookaside buffer*** (TLB) as a cache for page table entries (PTEs)
  - Try to avoid accessing the page table in memory
  - Split VPN into **TLB Tag** and **TLB Index** based on # of sets in TLB
  - Management bits include valid (like $, not PT) and TLB tag





Address Translation

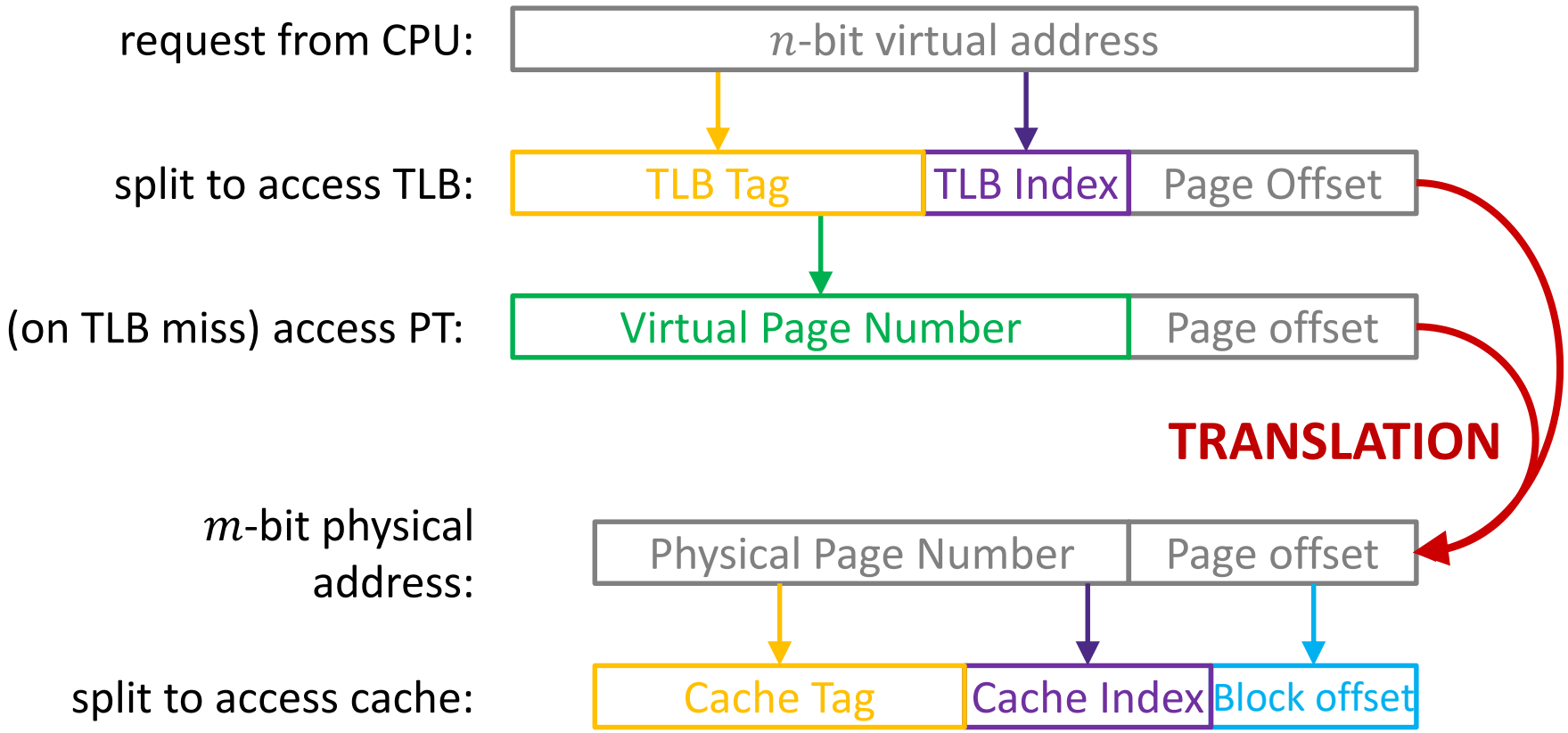
- VM is complicated, but also elegant and effective
  - Level of indirection to provide isolated memory & caching
  - TLB as a cache of page tables avoids two trips to memory for every memory access



Fetching Data on a Memory Read

1) Address Translation (check TLB)
   - Input: VPN, Output: PPN
   - *TLB Hit:* Fetch translation, return PPN
   - *TLB Miss:* Check page table (in memory)
     - *Page Table Hit:* Load page table entry into TLB
     - *Page Fault:* Fetch page from disk to memory, update corresponding page table entry, then load entry into TLB
2) Fetch Data (check cache)
   - Input: physical address, Output: data
   - *Cache Hit:* Return data value to processor
   - *Cache Miss:* Fetch data value from memory, store it in cache, return it to processor

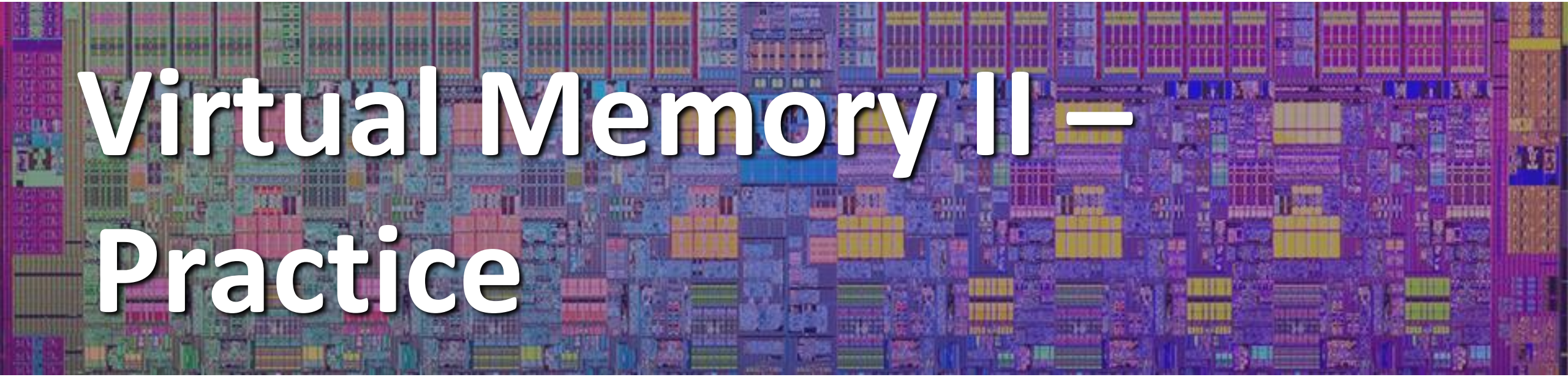
Address Manipulation



Lesson Q&A

- Learning Objectives:
  - Determine virtual memory parameters related to addresses, page tables, and TLBs.
  - Perform address translations (virtual address → physical address).
  - Describe the relationships between virtual memory parameters and policies.

- What lingering questions do you have from the lesson?
  - Chat with your neighbors about the lesson for a few minutes to come up with questions

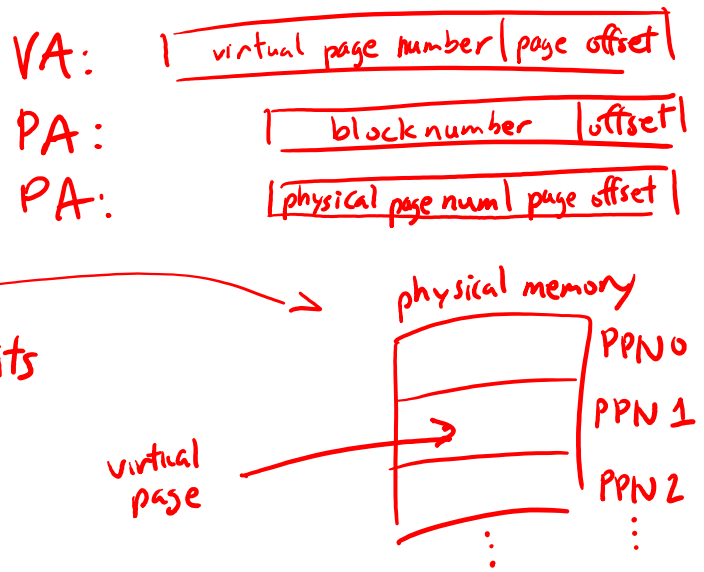


Virtual Memory II – Practice

Virtual Memory Concept Questions

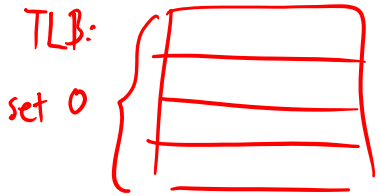
- Which terms from caching are most similar/analogous to the new virtual memory terms?
  - block #, block size, cache line, cache set, index width, management bits, offset width, tag width

  - page size
    block size
  - page offset width
    (block) offset width
  - virtual page number
    block number
  - physical page number
    block number or cache set
  - page table entry
    cache line: data of interest + management bits
  - access rights
    management bits



VM Parameters Questions

- Our system has the following properties
  - 1 MiB of physical address space — $m = 20$ bits
  - 4 GiB of virtual address space — $n = 32$ bits
  - 32 KiB page size — $p = 15$ bits
  - 4-entry fully associative TLB with LRU replacement — 1 set



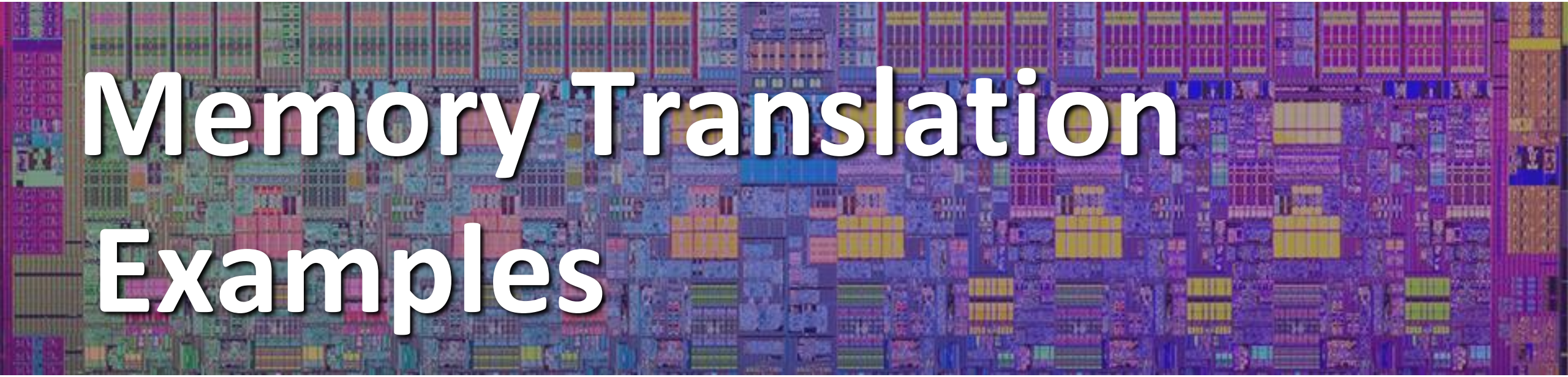
a) Fill in the following blanks:

2^{17} Entries in a page table (2^{n-p} ← # of virtual pages)

20 Minimum bit-width of PTBR ← physical address of PT (m)

17 TLBT bits (VPN → TLBT/TLBI, here TLBI = 0)

2^5 Max # of valid entries in a page table (2^{m-p} ← # of pages in physical memory)



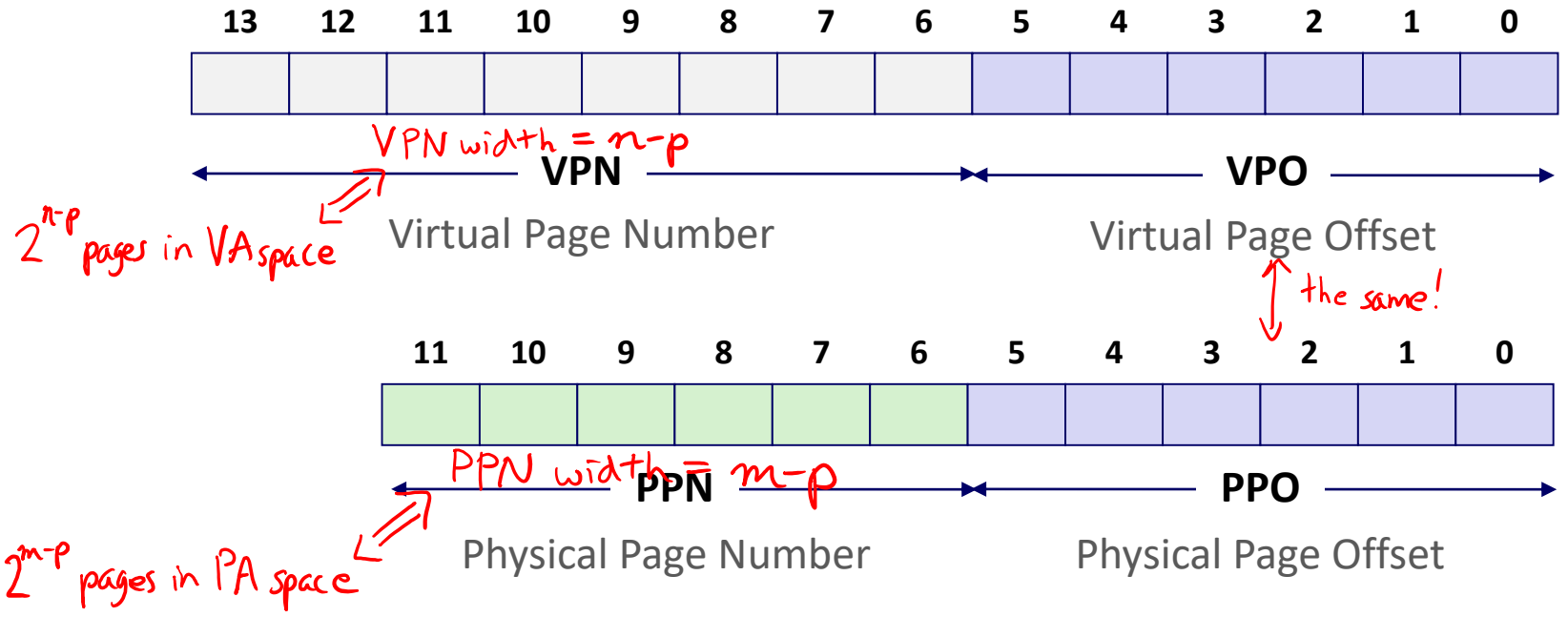
Memory Translation Examples

Summary of Address Translation Symbols

- Basic Parameters
  - $\mathrm{N} = 2^n$ Number of addresses in virtual address space
  - $\mathrm{M} = 2^m$ Number of addresses in physical address space
  - $\mathrm{P} = 2^p$ Page size (bytes)
- Components of the virtual address (VA)
  - **VPO** Virtual page offset
  - **VPN** Virtual page number
  - **TLBI** TLB index
  - **TLBT** TLB tag
- Components of the physical address (PA)
  - **PPO** Physical page offset (same as VPO)
  - **PPN** Physical page number

Simple Memory System Example (small)

- Addressing
  - 14-bit virtual addresses $n = 14$ bits $\iff$ $N = 16$ KiB VA space
  - 12-bit physical address $m = 12$ bits $\iff$ $M = 4$ KiB PA space
  - Page size = 64 bytes $P = 64$ B $\iff$ $p = 6$ bits



Simple Memory System: Page Table

- Only showing first 16 entries (out of 2^{n-p} = 2^8 = 256) one for every virtual page
  - **Note:** showing 2 hex digits for PPN even though only 6 bits
  - **Note:** other management bits not shown, but part of PTE (D, R, W, X)

VPN	PPN	Valid
0	28	1
1	–	0
2	33	1
3	02	1
4	–	0
5	16	1
6	–	0
7	–	0

VPN	PPN	Valid
8	0x13	1
9	17	1
A	09	1
B	–	0
C	–	0
D	2D	1
E	–	0
F	0D	1
⋮	⋮	⋮

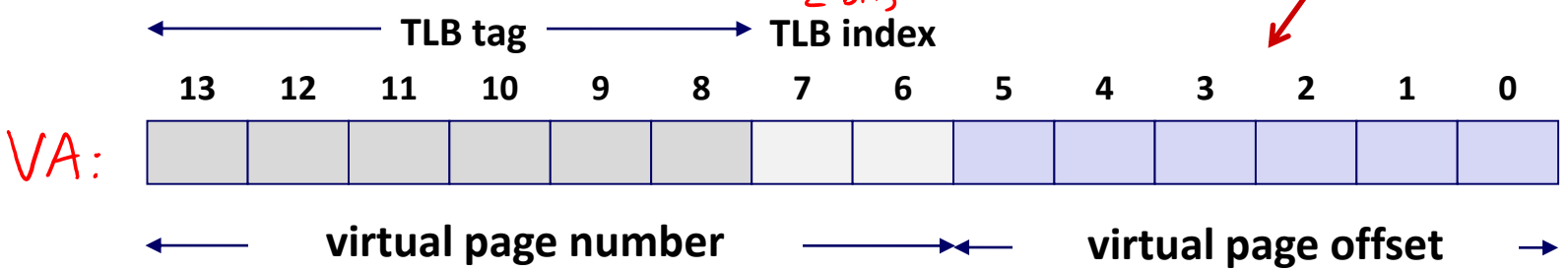
Simple Memory System: TLB

- 16 entries total
- 4-way set associative

16/4 = 4 sets ⇔ 2 bits

Why does the TLB ignore the page offset?

not part of its job! (address translation)

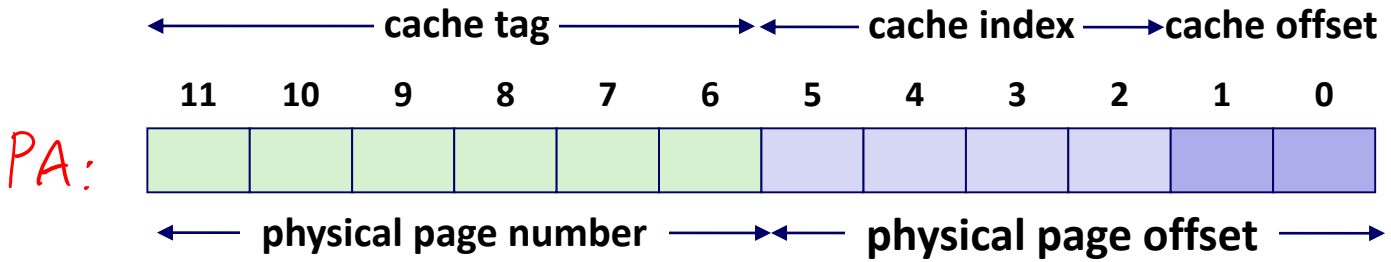


	Way 0			Way 1			Way 2			Way 3		
Set	***Tag***	***PPN***	***V***	***Tag***	***PPN***	***V***	***Tag***	***PPN***	***V***	***Tag***	***PPN***	***V***
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Simple Memory System: Cache

- Direct-mapped with $K = 4$ B, $C/K = 16$
- Physically addressed

Note: It is just coincidence that the PPN is the same width as the cache Tag



Index	Tag	Valid	B0	B1	B2	B3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

Index	Tag	Valid	B0	B1	B2	B3
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

Current State of Memory System

Circled #s refer to Memory Request Example #

TLB:

	Set	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V
③	0	03	–	0	09	0D	1	00	–	0	07	02	1
④	1	03	2D	1	02	–	0	04	–	0	0A	–	0
②	2	02	–	0	08	–	0	06	–	0	03	–	0
①	3	07	–	0	03	0D	1	0A	34	1	02	–	0

Page table (partial):

	VPN	PPN	V	VPN	PPN	V
③	0	28	1	8	13	1
	1	–	0	9	17	1
	2	33	1	A	09	1
	3	02	1	B	–	0
	4	–	0	C	–	0
	5	16	1	D	2D	1
	6	–	0	② E	–	0
	7	–	0	F	0D	1

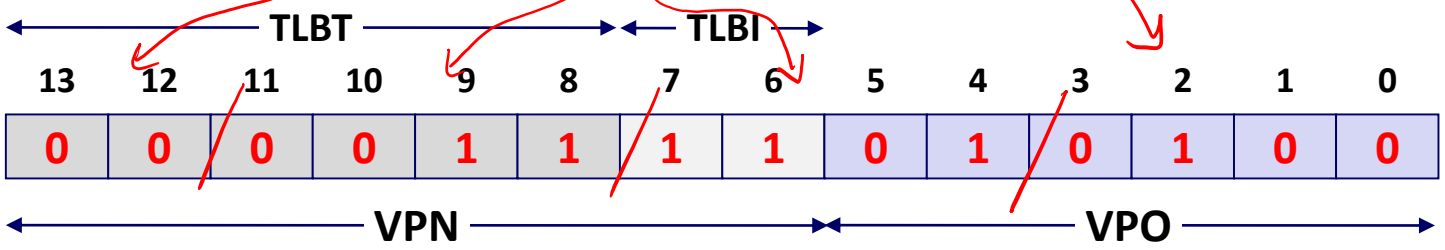
Cache:

	Index	Tag	V	B0	B1	B2	B3		Index	Tag	V	B0	B1	B2	B3
	0	19	1	99	11	23	11	③	8	24	1	3A	00	51	89
	1	15	0	–	–	–	–		9	2D	0	–	–	–	–
	2	1B	1	00	02	04	08	④	A	2D	1	93	15	DA	3B
	3	36	0	–	–	–	–		B	0B	0	–	–	–	–
	4	32	1	43	6D	8F	09		C	12	0	–	–	–	–
①	5	0D	1	36	72	F0	1D		D	16	1	04	96	34	15
	6	31	0	–	–	–	–		E	13	1	83	77	1B	D3
	7	16	1	11	C2	DF	03		F	14	0	–	–	–	–

Memory Request Example #1

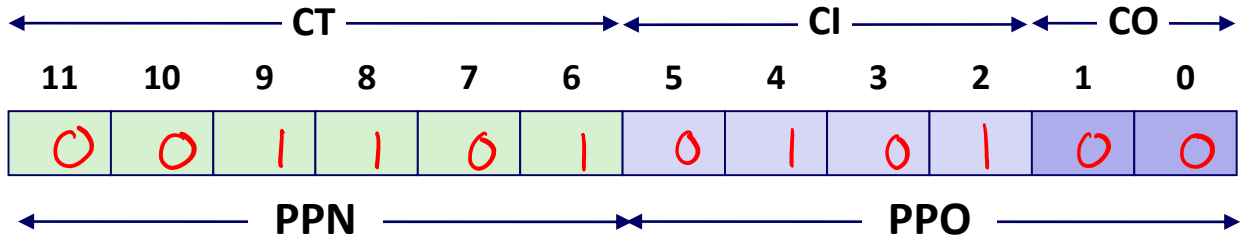
Note: It is just coincidence that the PPN is the same width as the cache Tag

- Virtual Address: 0x03D4



VPN 0xF (check this entry of the page table) TLBT 0x03 (look for this tag within TLB set) TLBI 3 (check this set of the TLB) TLB Hit? Y Page Fault? N PPN 0x0D

- Physical Address:

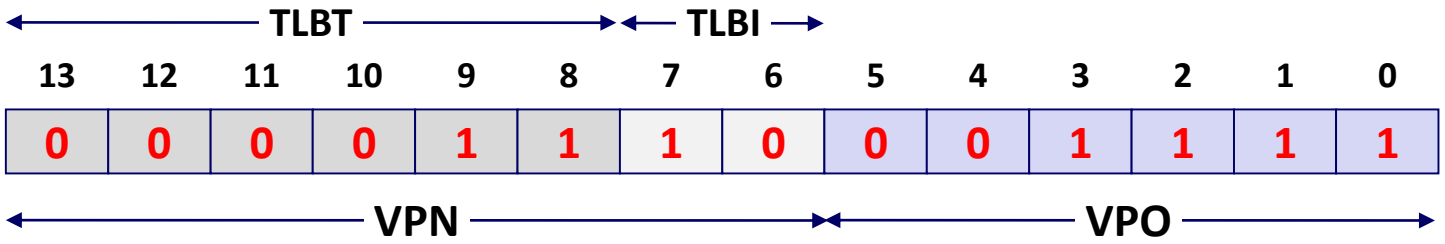


CT 0x0D (look for this tag within cache set) CI 5 (check this set of the cache) CO 0 (which byte of cache block) Cache Hit? Y Data (byte) 0x36

Memory Request Example #2

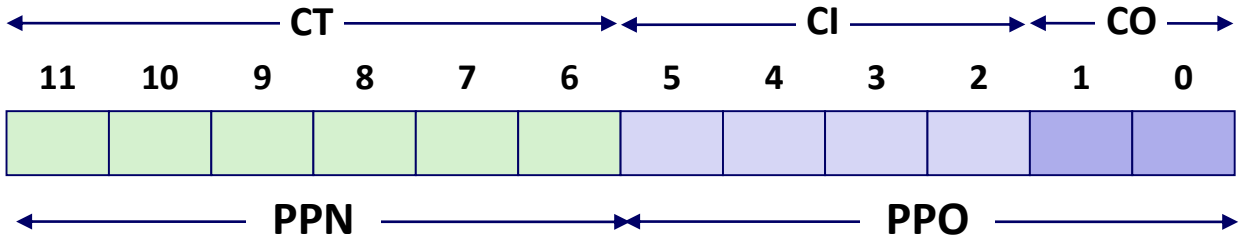
Note: It is just coincidence that the PPN is the same width as the cache Tag

- Virtual Address: 0x038F



VPN 0x0E TLBT 0x03 TLBI 2 TLB Hit? N Page Fault? Y PPN n/a

- Physical Address:

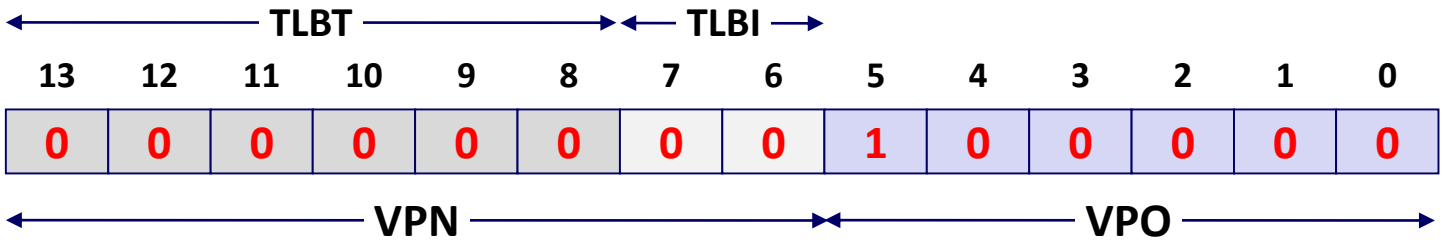


CT _____ CI _____ CO _____ Cache Hit? ___ Data (byte) _____

Memory Request Example #3

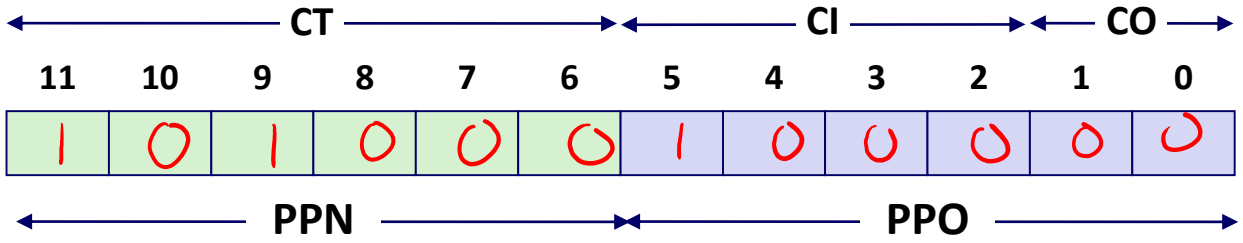
Note: It is just coincidence that the PPN is the same width as the cache Tag

- Virtual Address: 0x0020



VPN 0x00 TLBT 0x06 TLBI 0 TLB Hit? N Page Fault? N PPN 0x28

- Physical Address:

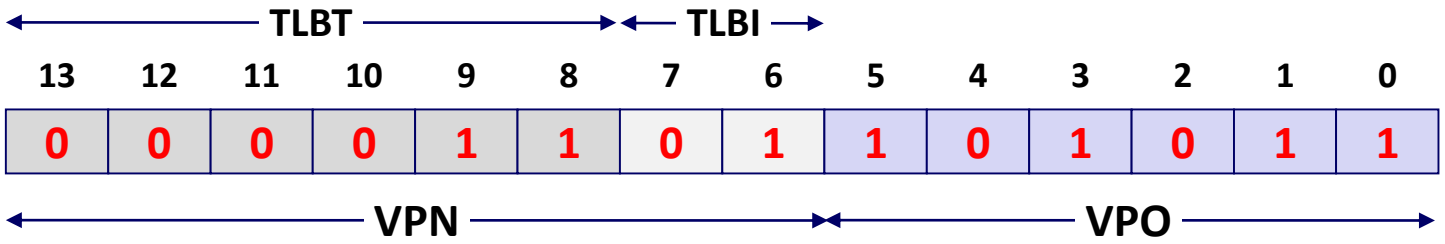


CT 0x28 CI 8 CO 0 Cache Hit? N Data (byte) n/a

Memory Request Example #4

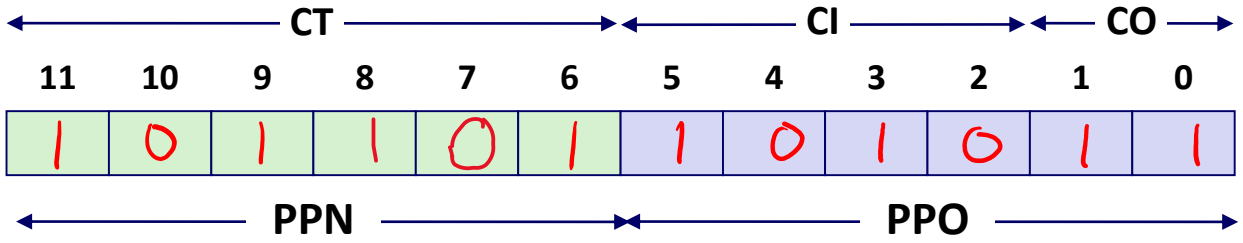
Note: It is just coincidence that the PPN is the same width as the cache Tag

- Virtual Address: 0x036B



VPN 0x0D TLBT 0x03 TLBI 1 TLB Hit? Y Page Fault? N PPN 0x2D

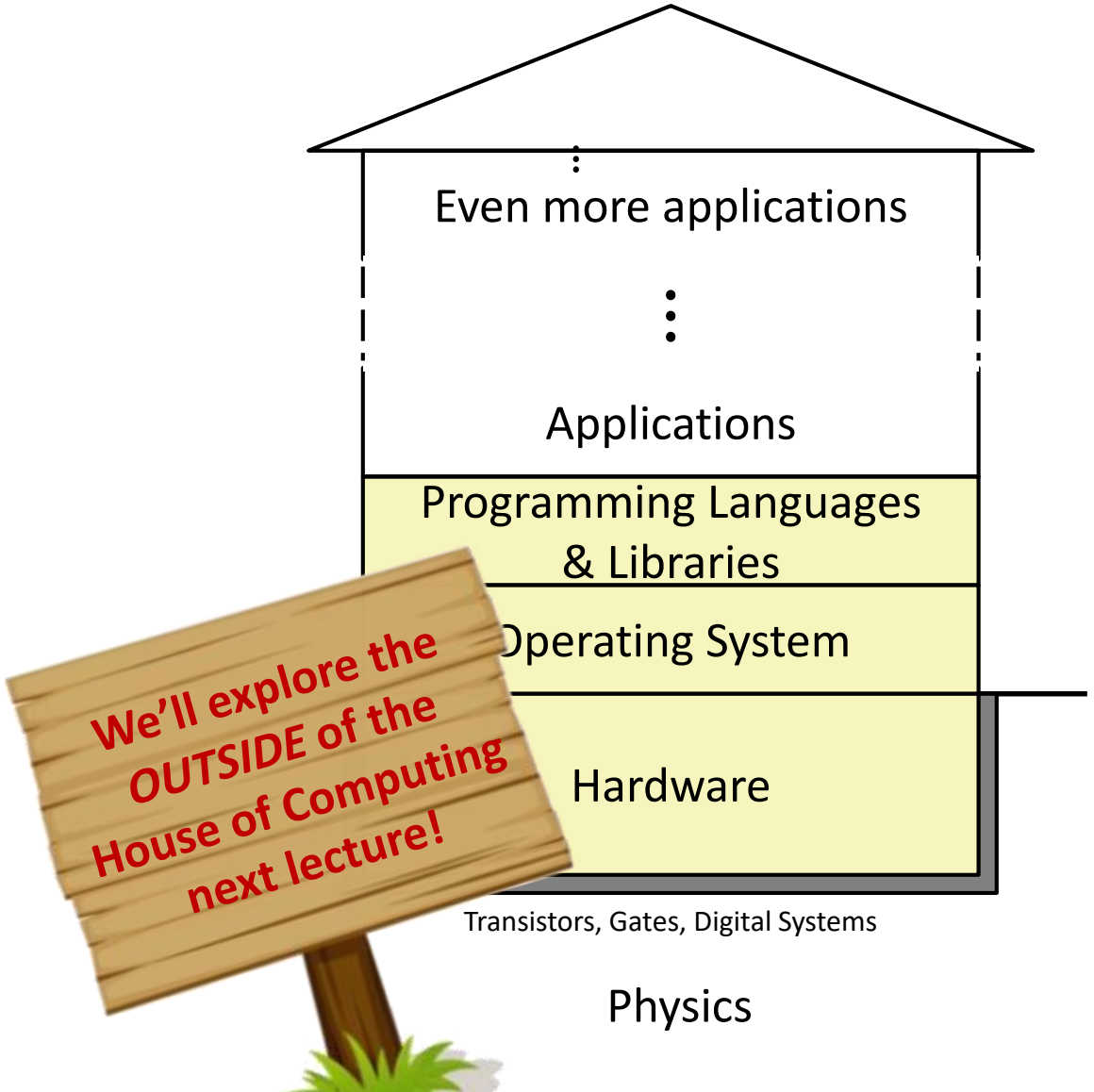
- Physical Address:



CT 0x2D CI A CO 3 Cache Hit? Y Data (byte) 0x3B

We made it! 😉 😎 😂

- Topic Group 1: **Data**
  - Memory, Data, Integers, Floating Point, Arrays, Structs
- Topic Group 2: **Programs**
  - x86-64 Assembly, Procedures, Stacks, Executables
- Topic Group 3: **Scale & Coherence**
  - Caches, Memory Allocation, Processes, Virtual Memory



Group Work Time

- During this time, you are encouraged to work on the following:
  1) If desired, continue your discussion
  2) Work on the homework problems
  3) Work on the current lab

- Resources:
  - You can revisit the lesson material
  - Work together in groups and help each other out
  - Course staff will circle around to provide support