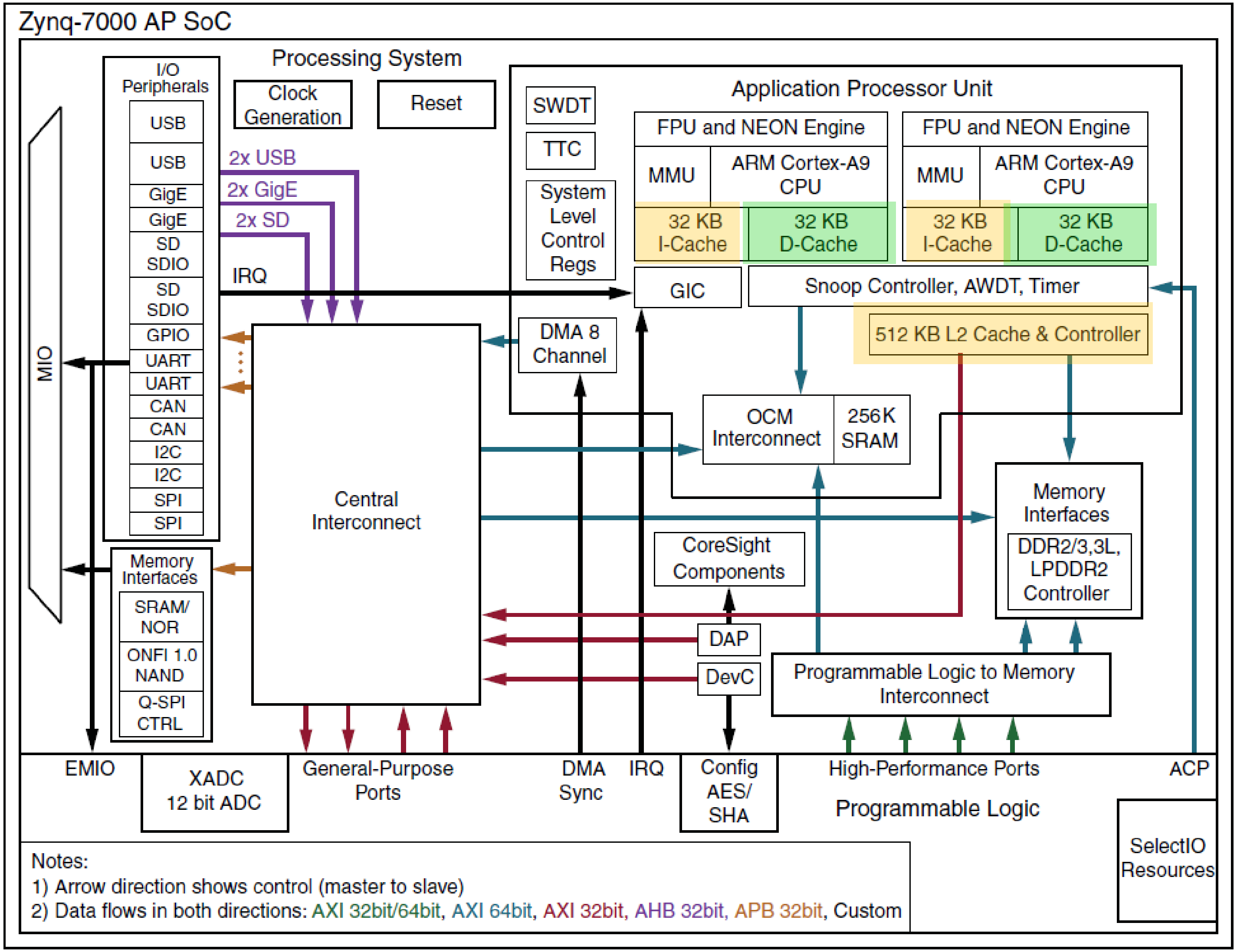
고려대학교
KOREA UNIVERSITY

COSE321 Computer Systems Design

Lecture 10. Caches in Zynq-7000 (Cortex-A9)

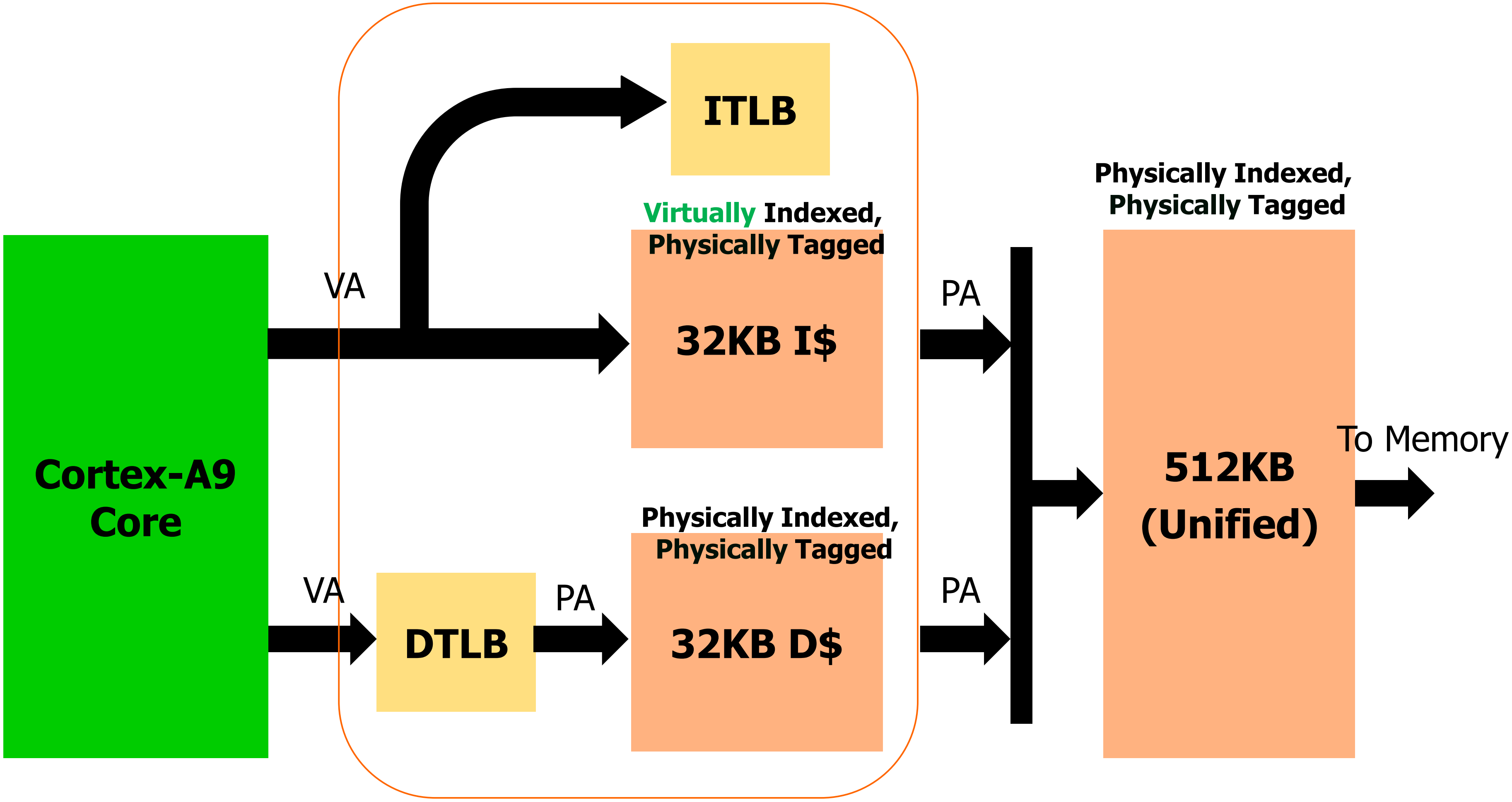
Prof. Taeweon Suh
Computer Science & Engineering
Korea University

Caches in Zynq



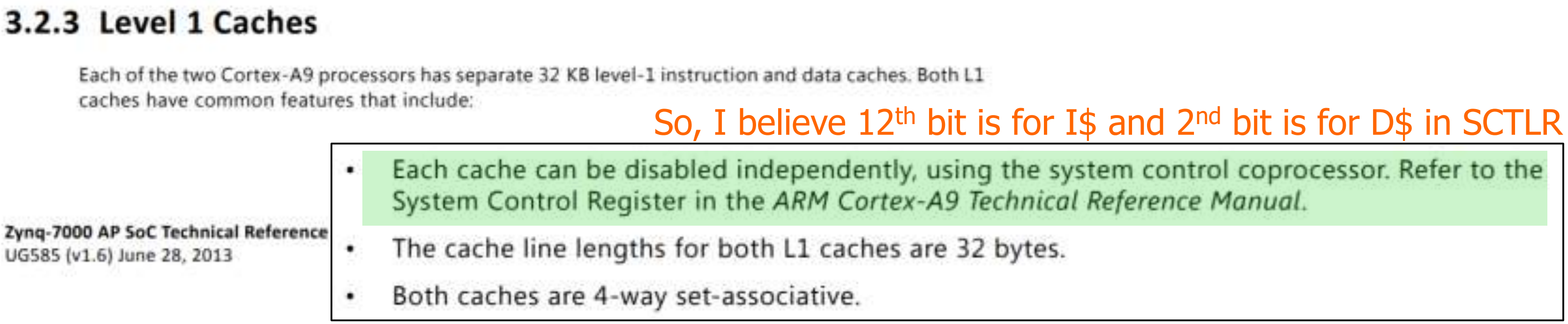
Notes:
1) Arrow direction shows control (master to slave)
2) Data flows in both directions: AXI 32bit/64bit, AXI 64bit, AXI 32bit, AHB 32bit, APB 32bit, Custom

Caches in Cortex-A9


Cortex-A9 Core
VA
ITLB
Virtually Indexed, Physically Tagged
32KB I$
VA
DTLB
PA
Physically Indexed, Physically Tagged
32KB D$
PA
PA
Physically Indexed, Physically Tagged
512KB (Unified)
To Memory

L1 Caches

- 32KB L1 I$ & 32KB L1 D$
  - I$: Virtually Indexed & Physically Tagged
  - D$: Physically Indexed & Physically Tagged
- Line size: 32B (=8 words)
- 4-way set associative
- Replacement policies
  - Pseudo round-robin or pseudo random
- Critical word first filling



3.2.3 Level 1 Caches

Each of the two Cortex-A9 processors has separate 32 KB level-1 instruction and data caches. Both L1 caches have common features that include:

So, I believe 12th bit is for I$ and 2nd bit is for D$ in SCTLR

- Each cache can be disabled independently, using the system control coprocessor. Refer to the System Control Register in the *ARM Cortex-A9 Technical Reference Manual*.
- The cache line lengths for both L1 caches are 32 bytes.
- Both caches are 4-way set-associative.

Zynq-7000 AP SoC Technical Reference
UG585 (v1.6) June 28, 2013

L2 Cache

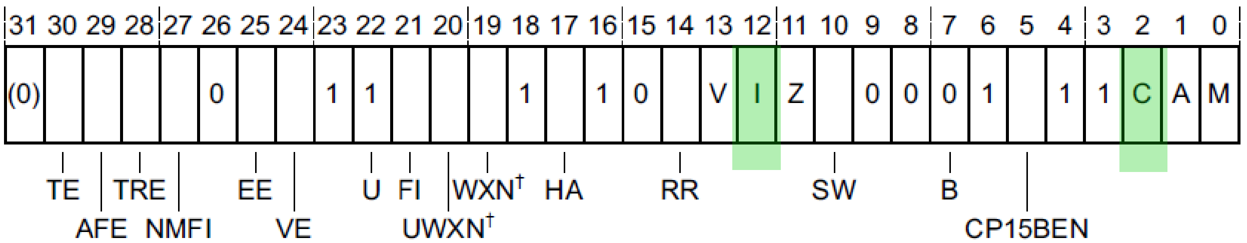
- 512KB Unified & Shared by both Cortex-A9s
  - Physically Indexed & Physically Tagged
  - MESI protocol for coherency
- Line size: 32B (=8 words)
- 8-way set associative
- Replacement policy
  - Pseudo random victim selection
- Critical word first filling
- L1/L2 exclusive mode support (i.e., data exists in either, but no both) – see `L2 auxilary register` in Zynq

http://esca.korea.ac.kr/teaching/cose321_CSD/labs/Zynq-7000-TRM.pdf

System Control Register (SCTLR)

- SCTLR provides the top level control of the system

In a VMSAv7 implementation, the SCTLR bit assignments are:



Bit	Value	Field
31	(0)	
30		TE
29		AFE
28		TRE
27		NMFI
26	0	
25		EE
24		VE
23	1	
22	1	U
21		FI
20		UWXN†
19		WXN†
18	1	
17		HA
16	1	
15	0	
14		RR
13	V	
12	I	
11	Z	
10		SW
9	0	
8	0	
7	0	B
6	1	
5		CP15BEN
4	1	
3	1	
2	C	
1	A	
0	M	

† Reserved before the introduction of the Virtualization Extensions, see text for more information.

- **TE: Thumb Exception Enable**
- **AFE: Access Flag Enable**

  0: In the translation table descriptors, AP[0] is an access permission bit

  1: In the translation table descriptors, AP[0] is an access flag
- **TRE: TEX remap enable**

  0: TEX remap disabled. TEX[2:0] are used with the C and B bits to describe memory region attributes

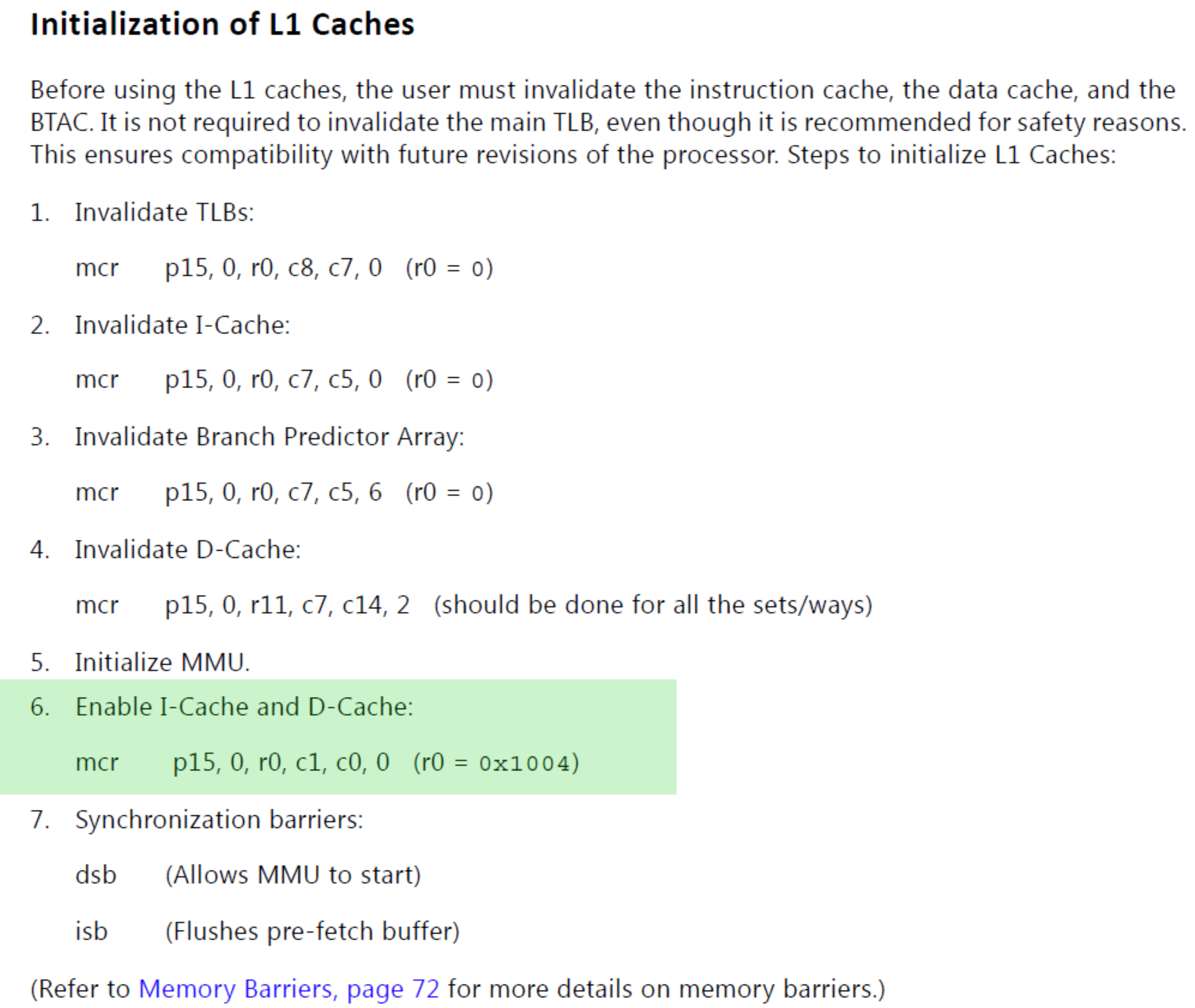
  1: TEX remap enabled. TEX[2:1] are reassigned for use as bits managed by OS. The TEX[0], C and B bits, with the MMU remap registers describe the memory region attributes
- **VE: Interrupt Vectors Enable**
- **RR: Round Robin select. Cache replacement policy**
- **V: Vectors bit. 0: Low vectors 0x0, 1: High vectors (Hivecs): 0xFFFF0000**
- **I: I$ enable**
- **Z: Branch prediction enable**
- **CP15BEN: CP15 barrier enable**
- **C: D$ enable, and Unified$ enable for data access.**
- ***Unified$ use for instruction access (Implementation defined)**
- **A: Alignment check enable**
- **M: MMU enable**

Non-Deterministic Cache Behavior

- Assume that the cache line size is 32B (=8 words)

Code	Worst-case scenario in $
0x091C: add r0, r1, r2	I$ miss (8-word line fill from 0x0900)
0x0920: mov r12, #0xA000	I$ miss (8-word line fill from 0x0920)
0x0924: ldr r3, [r12, #0]	D$ miss 1. D$ dirty line eviction 2. 8-word line fill from 0xA000
0x0928: ldm sp!, {r0-r9}	3 D$ misses 1. 3 D$ dirty line evictions 2. 3 Line fills (miss for r0, miss for r1~r8, and miss for r9)

L1 Caches



Initialization of L1 Caches

Before using the L1 caches, the user must invalidate the instruction cache, the data cache, and the BTAC. It is not required to invalidate the main TLB, even though it is recommended for safety reasons. This ensures compatibility with future revisions of the processor. Steps to initialize L1 Caches:

1. Invalidate TLBs:

   ```
   mcr     p15, 0, r0, c8, c7, 0   (r0 = 0)
   ```

2. Invalidate I-Cache:

   ```
   mcr     p15, 0, r0, c7, c5, 0   (r0 = 0)
   ```

3. Invalidate Branch Predictor Array:

   ```
   mcr     p15, 0, r0, c7, c5, 6   (r0 = 0)
   ```

4. Invalidate D-Cache:

   ```
   mcr     p15, 0, r11, c7, c14, 2   (should be done for all the sets/ways)
   ```

5. Initialize MMU.
6. Enable I-Cache and D-Cache:

   ```
   mcr     p15, 0, r0, c1, c0, 0   (r0 = 0x1004)
   ```

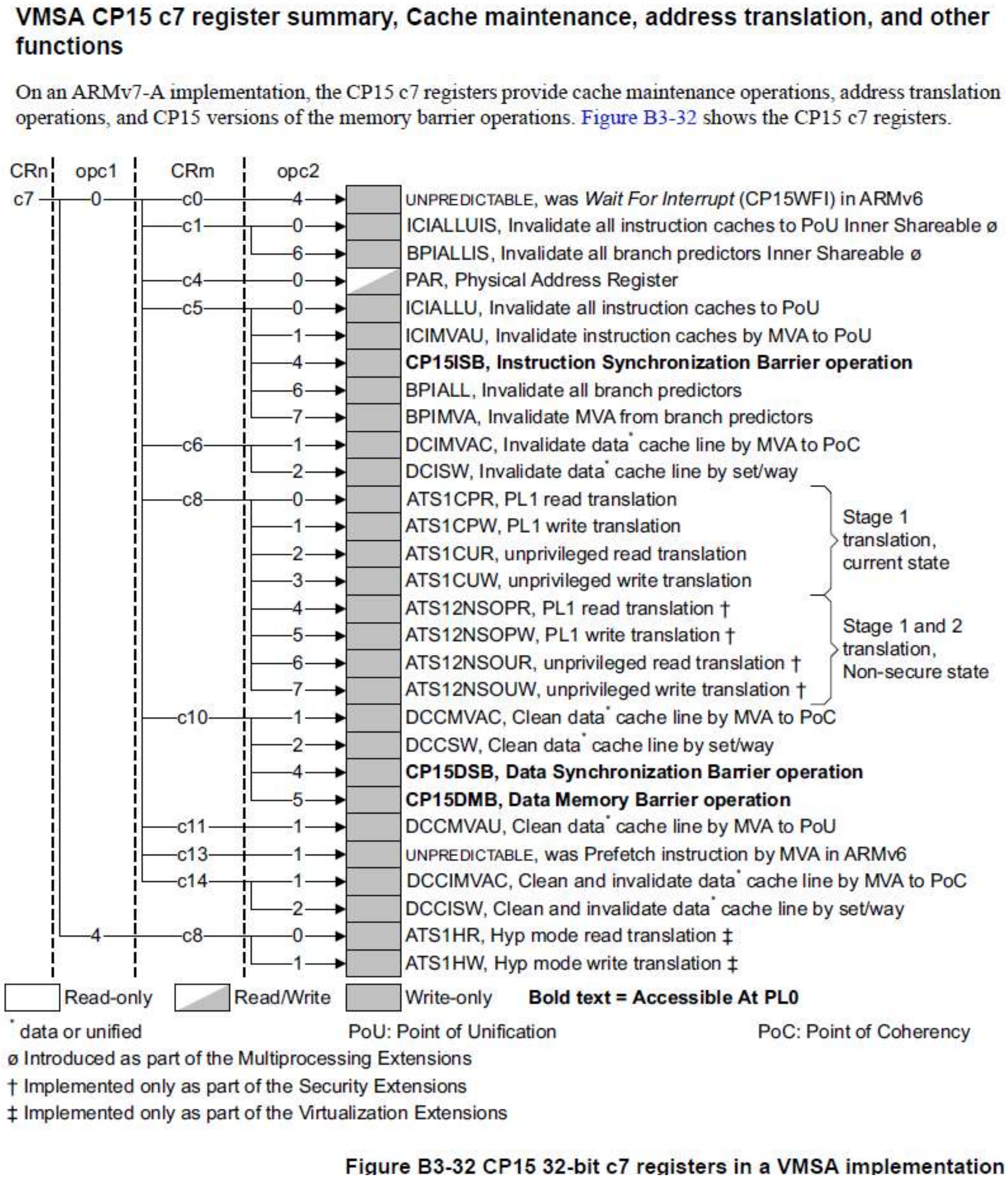
7. Synchronization barriers:

   ```
   dsb     (Allows MMU to start)
   isb     (Flushes pre-fetch buffer)
   ```

(Refer to Memory Barriers, page 72 for more details on memory barriers.)

Source: Zynq-7000 TRM

Cache Maintenance Operations



VMSA CP15 c7 register summary, Cache maintenance, address translation, and other functions

On an ARMv7-A implementation, the CP15 c7 registers provide cache maintenance operations, address translation operations, and CP15 versions of the memory barrier operations. Figure B3-32 shows the CP15 c7 registers.

CRn	opc1	CRm	opc2			
c7	0	c0	4	■	UNPREDICTABLE, was *Wait For Interrupt* (CP15WFI) in ARMv6	
		c1	0	■	ICIALLUIS, Invalidate all instruction caches to PoU Inner Shareable ø	
			6	■	BPIALLIS, Invalidate all branch predictors Inner Shareable ø	
		c4	0	◪	PAR, Physical Address Register	
		c5	0	■	ICIALLU, Invalidate all instruction caches to PoU	
			1	■	ICIMVAU, Invalidate instruction caches by MVA to PoU	
			4	■	**CP15ISB, Instruction Synchronization Barrier operation**	
			6	■	BPIALL, Invalidate all branch predictors	
			7	■	BPIMVA, Invalidate MVA from branch predictors	
		c6	1	■	DCIMVAC, Invalidate data* cache line by MVA to PoC	
			2	■	DCISW, Invalidate data* cache line by set/way	
		c8	0	■	ATS1CPR, PL1 read translation	Stage 1 translation, current state
			1	■	ATS1CPW, PL1 write translation	
			2	■	ATS1CUR, unprivileged read translation	
			3	■	ATS1CUW, unprivileged write translation	
			4	■	ATS12NSOPR, PL1 read translation †	Stage 1 and 2 translation, Non-secure state
			5	■	ATS12NSOPW, PL1 write translation †	
			6	■	ATS12NSOUR, unprivileged read translation †	
			7	■	ATS12NSOUW, unprivileged write translation †	
		c10	1	■	DCCMVAC, Clean data* cache line by MVA to PoC	
			2	■	DCCSW, Clean data* cache line by set/way	
			4	■	**CP15DSB, Data Synchronization Barrier operation**	
			5	■	**CP15DMB, Data Memory Barrier operation**	
		c11	1	■	DCCMVAU, Clean data* cache line by MVA to PoU	
		c13	1	■	UNPREDICTABLE, was Prefetch instruction by MVA in ARMv6	
		c14	1	■	DCCIMVAC, Clean and invalidate data* cache line by MVA to PoC	
			2	■	DCCISW, Clean and invalidate data* cache line by set/way	
	4	c8	0	■	ATS1HR, Hyp mode read translation ‡	
			1	■	ATS1HW, Hyp mode write translation ‡	

□ Read-only ◪ Read/Write ■ Write-only **Bold text = Accessible At PL0**

* data or unified PoU: Point of Unification PoC: Point of Coherency

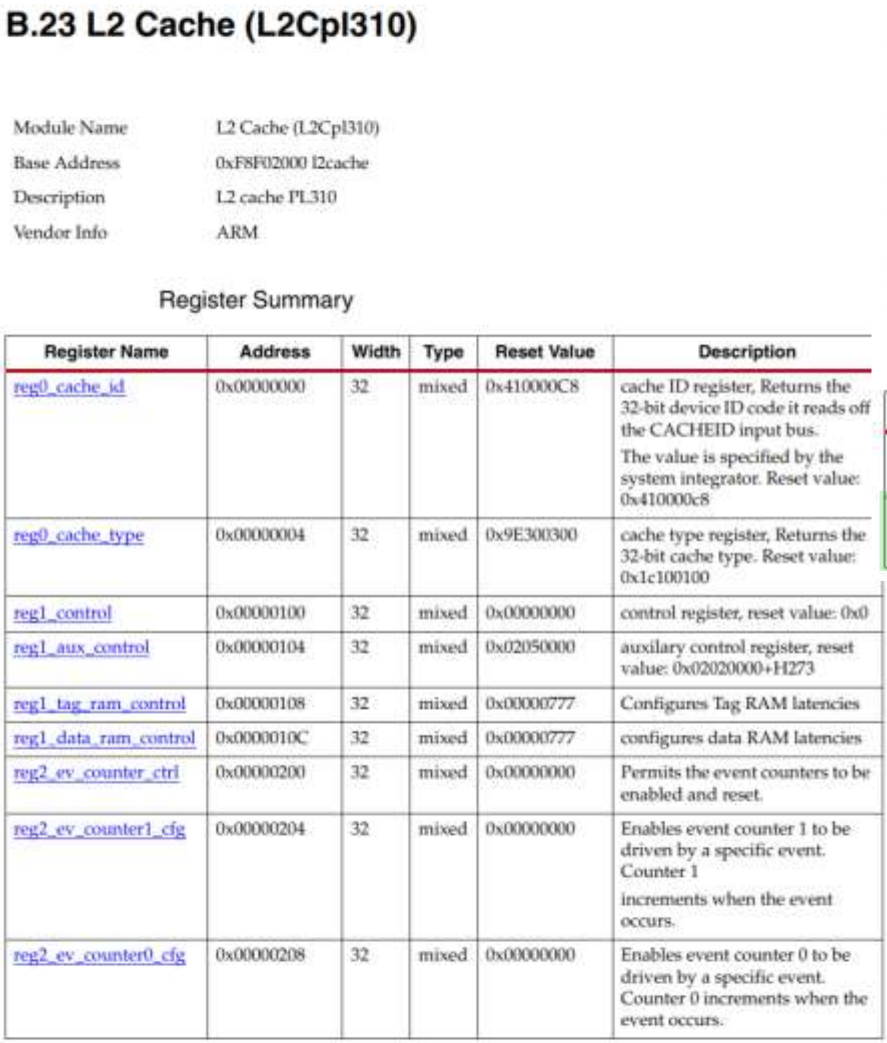
ø Introduced as part of the Multiprocessing Extensions

† Implemented only as part of the Security Extensions

‡ Implemented only as part of the Virtualization Extensions

Figure B3-32 CP15 32-bit c7 registers in a VMSA implementation

L2 Cache

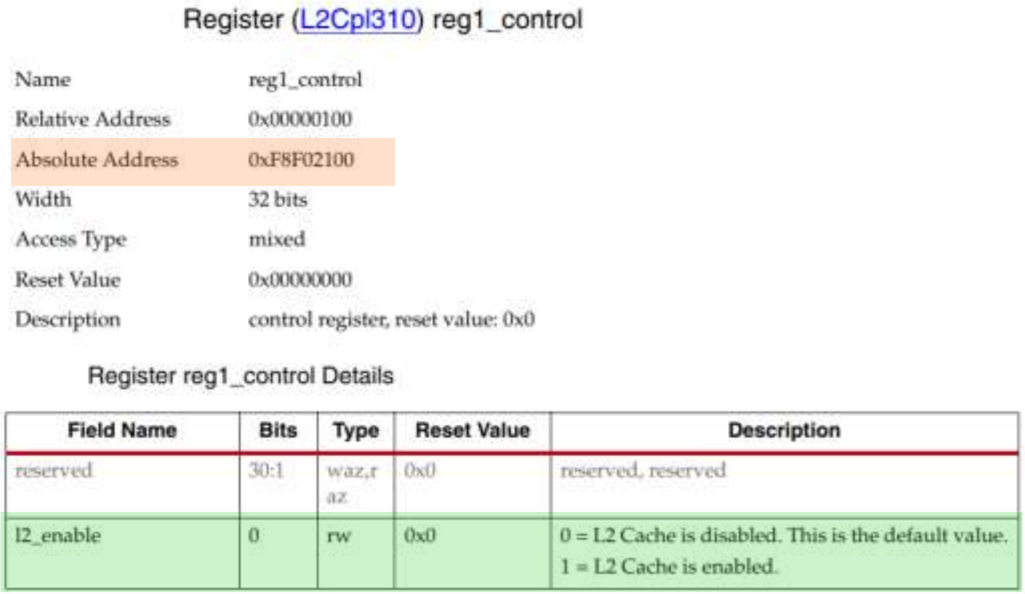


B.23 L2 Cache (L2Cpl310)

Module Name	L2 Cache (L2Cpl310)
Base Address	0xF8F02000 l2cache
Description	L2 cache PL310
Vendor Info	ARM

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
reg0_cache_id	0x00000000	32	mixed	0x410000C8	cache ID register, Returns the 32-bit device ID code it reads off the CACHEID input bus. The value is specified by the system integrator. Reset value: 0x410000c8
reg0_cache_type	0x00000004	32	mixed	0x9E300300	cache type register, Returns the 32-bit cache type. Reset value: 0x1c100100
reg1_control	0x00000100	32	mixed	0x00000000	control register, reset value: 0x0
reg1_aux_control	0x00000104	32	mixed	0x02050000	auxilary control register, reset value: 0x02020000+H273
reg1_tag_ram_control	0x00000108	32	mixed	0x00000777	Configures Tag RAM latencies
reg1_data_ram_control	0x0000010C	32	mixed	0x00000777	configures data RAM latencies
reg2_ev_counter_ctrl	0x00000200	32	mixed	0x00000000	Permits the event counters to be enabled and reset.
reg2_ev_counter1_cfg	0x00000204	32	mixed	0x00000000	Enables event counter 1 to be driven by a specific event. Counter 1 increments when the event occurs.
reg2_ev_counter0_cfg	0x00000208	32	mixed	0x00000000	Enables event counter 0 to be driven by a specific event. Counter 0 increments when the event occurs.



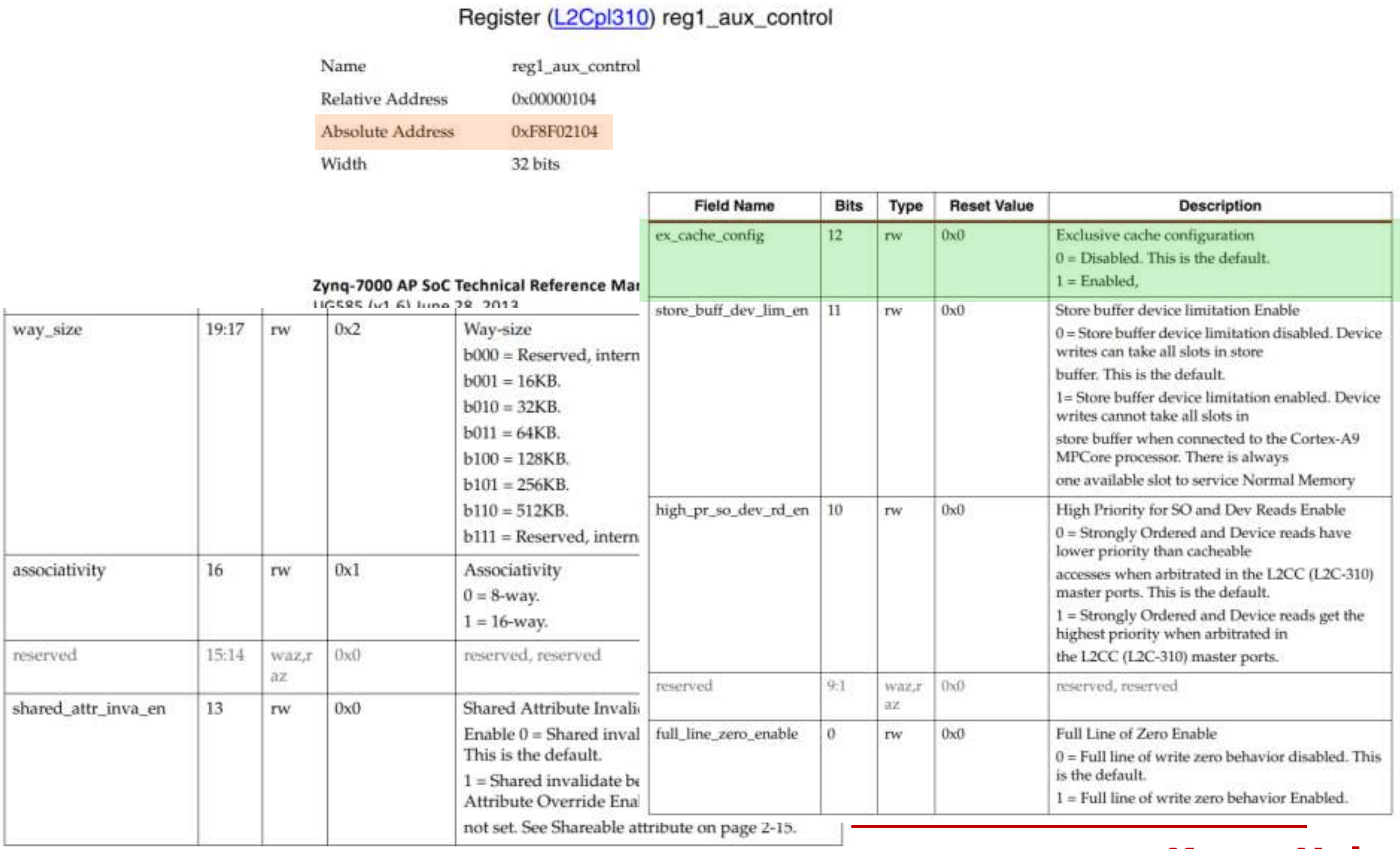
Register (L2Cpl310) reg1_control

Name	reg1_control
Relative Address	0x00000100
Absolute Address	0xF8F02100
Width	32 bits
Access Type	mixed
Reset Value	0x00000000
Description	control register, reset value: 0x0

Register reg1_control Details

Field Name	Bits	Type	Reset Value	Description
reserved	30:1	waz,raz	0x0	reserved, reserved
l2_enable	0	rw	0x0	0 = L2 Cache is disabled. This is the default value. 1 = L2 Cache is enabled.

Auxiliary Control Register



Register (L2Cpl310) reg1_aux_control

Name	reg1_aux_control
Relative Address	0x00000104
Absolute Address	0xF8F02104
Width	32 bits

Zynq-7000 AP SoC Technical Reference Mar
UG585 (v1.6) June 28, 2013

way_size	19:17	rw	0x2	Way-size
				b000 = Reserved, intern
				b001 = 16KB.
				b010 = 32KB.
				b011 = 64KB.
				b100 = 128KB.
				b101 = 256KB.
				b110 = 512KB.
				b111 = Reserved, intern

associativity	16	rw	0x1	Associativity 0 = 8-way. 1 = 16-way.
reserved	15:14	waz,raz	0x0	reserved, reserved
shared_attr_inva_en	13	rw	0x0	Shared Attribute Invali Enable 0 = Shared inval This is the default. 1 = Shared invalidate be Attribute Override Enal not set. See Shareable attribute on page 2-15.

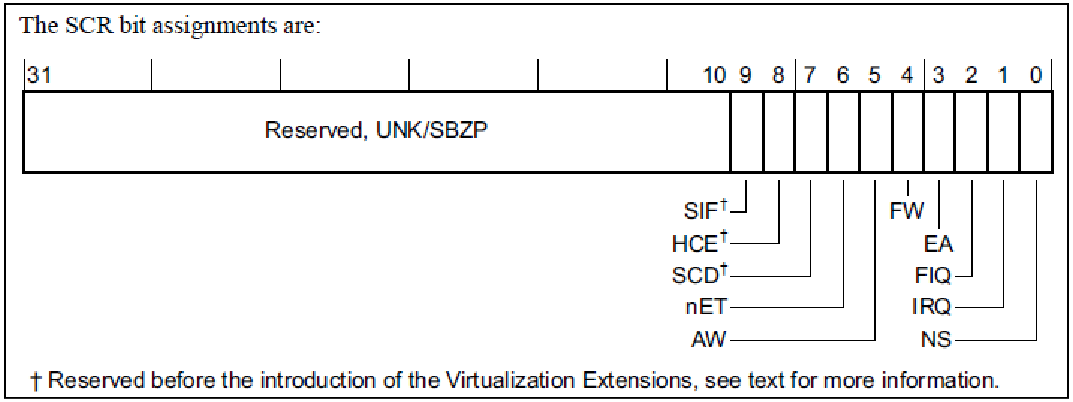
Field Name	Bits	Type	Reset Value	Description
ex_cache_config	12	rw	0x0	Exclusive cache configuration 0 = Disabled. This is the default. 1 = Enabled,
store_buff_dev_lim_en	11	rw	0x0	Store buffer device limitation Enable 0 = Store buffer device limitation disabled. Device writes can take all slots in store buffer. This is the default. 1= Store buffer device limitation enabled. Device writes cannot take all slots in store buffer when connected to the Cortex-A9 MPCore processor. There is always one available slot to service Normal Memory
high_pr_so_dev_rd_en	10	rw	0x0	High Priority for SO and Dev Reads Enable 0 = Strongly Ordered and Device reads have lower priority than cacheable accesses when arbitrated in the L2CC (L2C-310) master ports. This is the default. 1 = Strongly Ordered and Device reads get the highest priority when arbitrated in the L2CC (L2C-310) master ports.
reserved	9:1	waz,raz	0x0	reserved, reserved
full_line_zero_enable	0	rw	0x0	Full Line of Zero Enable 0 = Full line of write zero behavior disabled. This is the default. 1 = Full line of write zero behavior Enabled.

Backup Slides

Secure Configuration Register (SCR)

- SCR defines the configuration of the current security state
  - Security state of the CPU: Secure or Non-secure
  - What mode CPU branches to if IRQ, FIQ or external abort occurs
  - Whether the CPSR.{F,A} can be modified when SCR.NS == 1

- **SIF: Secure Instruction Fetch**
- **HCE: Hyp Call Enable**
- **SCD: Secure Monitor Call disable**
- **nET: Not Early Termination**
- **AW: A bit writable**
- **FW: F bit writable**
- **EA: External Abort handler**
- **FIQ: FIQ handler**
- **IRQ: IRQ handler**
- **NS: Non-Secure bit**

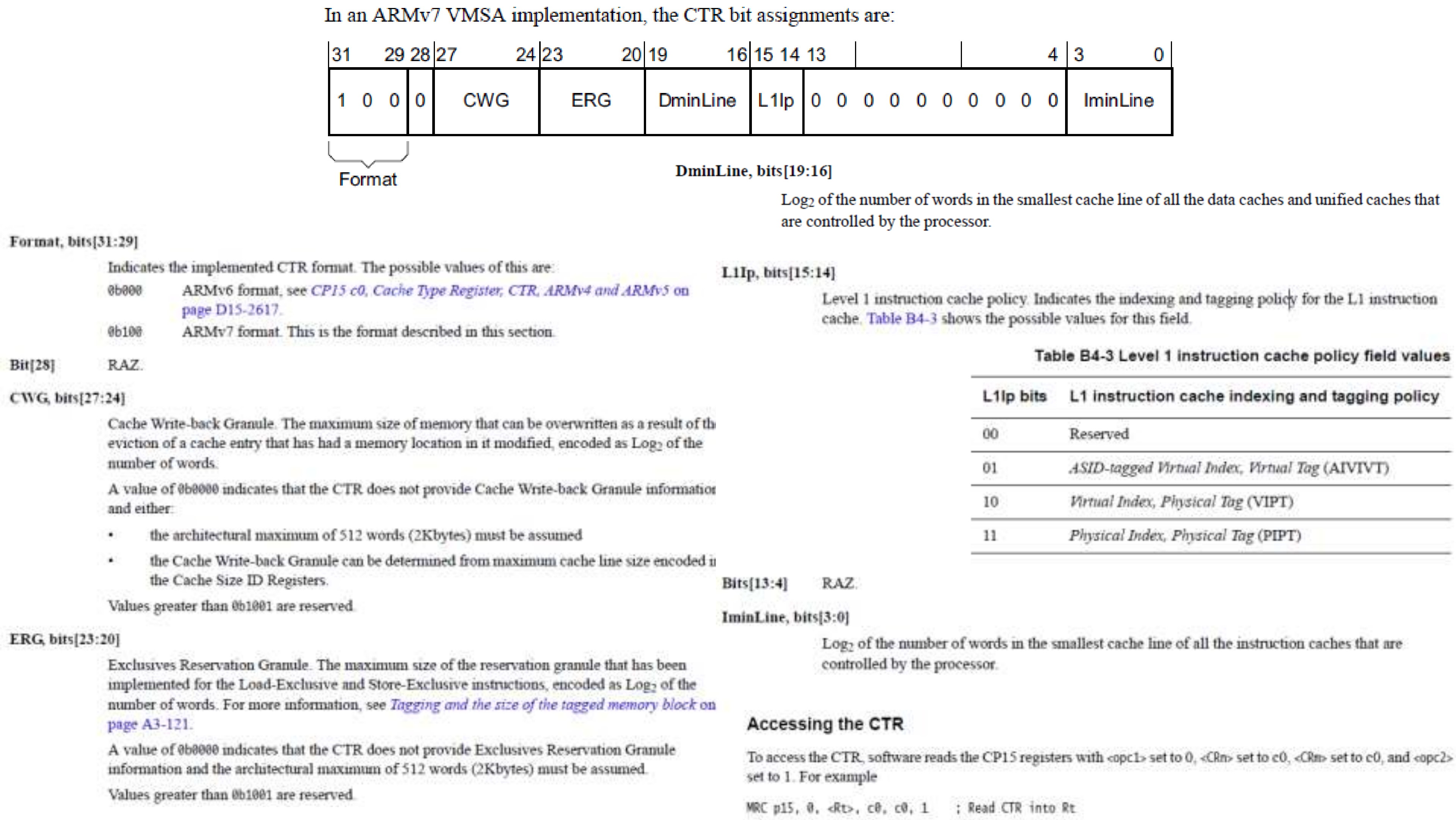


The SCR bit assignments are:

† Reserved before the introduction of the Virtualization Extensions, see text for more information.

Cache Type Register (CTR)

- CTR provides information about the architecture of the caches



In an ARMv7 VMSA implementation, the CTR bit assignments are:

31 29	28	27 24	23 20	19 16	15 14	13 4	3 0
1 0 0	0	CWG	ERG	DminLine	L1Ip	0 0 0 0 0 0 0 0 0 0	IminLine

Format

Format, bits[31:29]

Indicates the implemented CTR format. The possible values of this are:

0b000 ARMv6 format, see *CP15 c0, Cache Type Register, CTR, ARMv4 and ARMv5* on page D15-2617.

0b100 ARMv7 format. This is the format described in this section.

Bit[28] RAZ.

CWG, bits[27:24]

Cache Write-back Granule. The maximum size of memory that can be overwritten as a result of th eviction of a cache entry that has had a memory location in it modified, encoded as Log$_2$ of the number of words.

A value of 0b0000 indicates that the CTR does not provide Cache Write-back Granule informatio and either:

- the architectural maximum of 512 words (2Kbytes) must be assumed
- the Cache Write-back Granule can be determined from maximum cache line size encoded i the Cache Size ID Registers.

Values greater than 0b1001 are reserved.

ERG, bits[23:20]

Exclusives Reservation Granule. The maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions, encoded as Log$_2$ of the number of words. For more information, see *Tagging and the size of the tagged memory block* on page A3-121.

A value of 0b0000 indicates that the CTR does not provide Exclusives Reservation Granule information and the architectural maximum of 512 words (2Kbytes) must be assumed.

Values greater than 0b1001 are reserved.

DminLine, bits[19:16]

Log$_2$ of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the processor.

L1Ip, bits[15:14]

Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Table B4-3 shows the possible values for this field.

Table B4-3 Level 1 instruction cache policy field values

L1Ip bits	L1 instruction cache indexing and tagging policy
00	Reserved
01	*ASID-tagged Virtual Index, Virtual Tag* (AIVIVT)
10	*Virtual Index, Physical Tag* (VIPT)
11	*Physical Index, Physical Tag* (PIPT)

Bits[13:4] RAZ.

IminLine, bits[3:0]

Log$_2$ of the number of words in the smallest cache line of all the instruction caches that are controlled by the processor.

Accessing the CTR

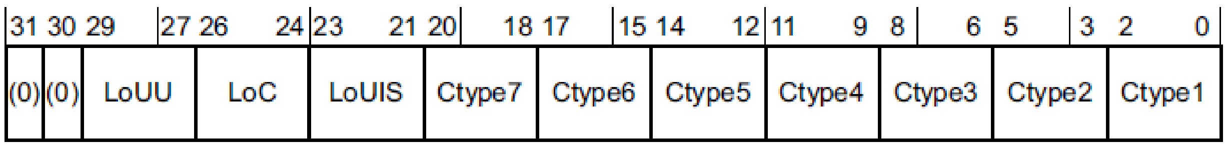
To access the CTR, software reads the CP15 registers with <opc1> set to 0, <CRn> set to c0, <CRm> set to c0, and <opc2> set to 1. For example

```
MRC p15, 0, <Rt>, c0, c0, 1    ; Read CTR into Rt
```

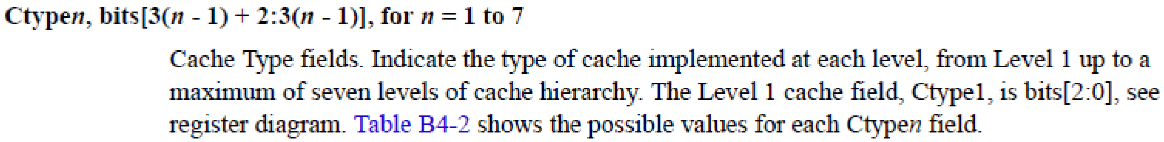
Cache Level ID Register (CLIDR)

- CLIDR identifies the type of caches implemented at each level, up to a max. of 7 levels

The CLIDR bit assignments are:



31	30	29 27	26 24	23 21	20 18	17 15	14 12	11 9	8 6	5 3	2 0
(0)	(0)	LoUU	LoC	LoUIS	Ctype7	Ctype6	Ctype5	Ctype4	Ctype3	Ctype2	Ctype1



Ctype*n*, bits[3(*n* - 1) + 2:3(*n* - 1)], for *n* = 1 to 7

Cache Type fields. Indicate the type of cache implemented at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. The Level 1 cache field, Ctype1, is bits[2:0], see register diagram. Table B4-2 shows the possible values for each Ctype*n* field.

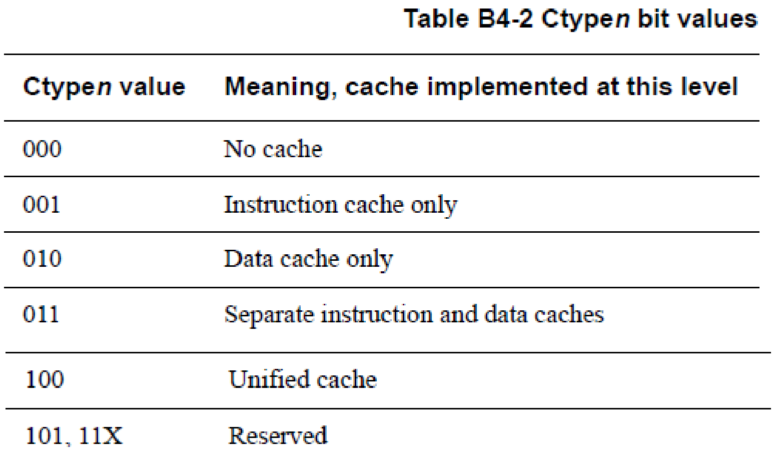
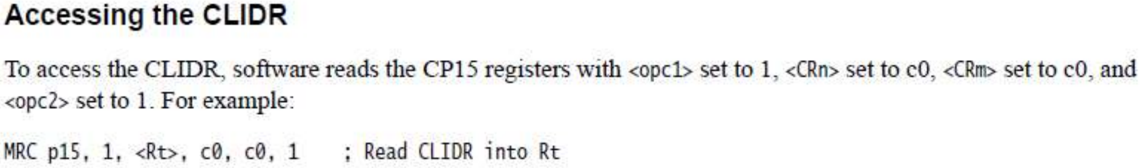


Table B4-2 Ctype*n* bit values

Ctype*n* value	Meaning, cache implemented at this level
000	No cache
001	Instruction cache only
010	Data cache only
011	Separate instruction and data caches
100	Unified cache
101, 11X	Reserved

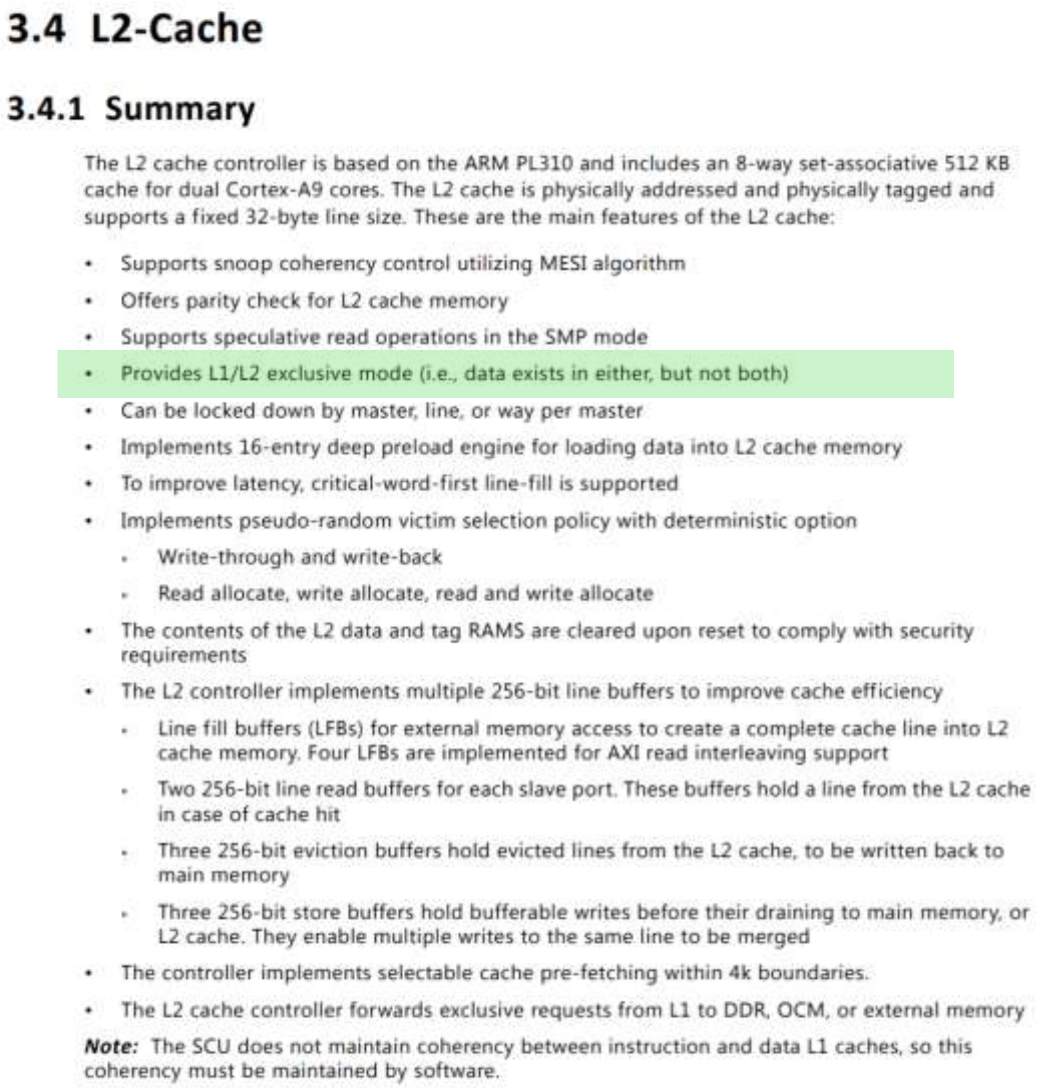


Accessing the CLIDR

To access the CLIDR, software reads the CP15 registers with <opc1> set to 1, <CRn> set to c0, <CRm> set to c0, and <opc2> set to 1. For example:

```
MRC p15, 1, <Rt>, c0, c0, 1    ; Read CLIDR into Rt
```

L2 Cache



3.4 L2-Cache

3.4.1 Summary

The L2 cache controller is based on the ARM PL310 and includes an 8-way set-associative 512 KB cache for dual Cortex-A9 cores. The L2 cache is physically addressed and physically tagged and supports a fixed 32-byte line size. These are the main features of the L2 cache:

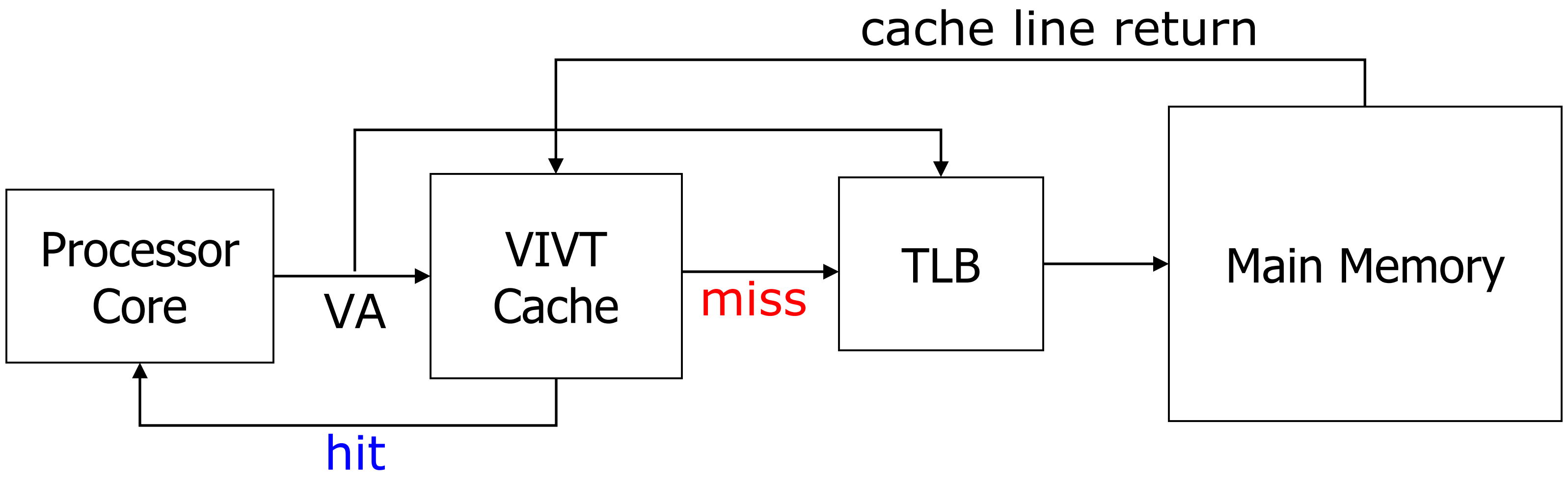
- Supports snoop coherency control utilizing MESI algorithm
- Offers parity check for L2 cache memory
- Supports speculative read operations in the SMP mode
- Provides L1/L2 exclusive mode (i.e., data exists in either, but not both)
- Can be locked down by master, line, or way per master
- Implements 16-entry deep preload engine for loading data into L2 cache memory
- To improve latency, critical-word-first line-fill is supported
- Implements pseudo-random victim selection policy with deterministic option
  - Write-through and write-back
  - Read allocate, write allocate, read and write allocate
- The contents of the L2 data and tag RAMS are cleared upon reset to comply with security requirements
- The L2 controller implements multiple 256-bit line buffers to improve cache efficiency
  - Line fill buffers (LFBs) for external memory access to create a complete cache line into L2 cache memory. Four LFBs are implemented for AXI read interleaving support
  - Two 256-bit line read buffers for each slave port. These buffers hold a line from the L2 cache in case of cache hit
  - Three 256-bit eviction buffers hold evicted lines from the L2 cache, to be written back to main memory
  - Three 256-bit store buffers hold bufferable writes before their draining to main memory, or L2 cache. They enable multiple writes to the same line to be merged
- The controller implements selectable cache pre-fetching within 4k boundaries.
- The L2 cache controller forwards exclusive requests from L1 to DDR, OCM, or external memory

Note: The SCU does not maintain coherency between instruction and data L1 caches, so this coherency must be maintained by software.

TLB and Caches

- Several Design Alternatives
  - VIVT: Virtually-indexed Virtually-tagged Cache
  - VIPT: Virtually-indexed Physically-tagged Cache
  - PIVT: Physically-indexed Virtually-tagged Cache
    - Not outright useful, R6000 is the only used this.
  - PIPT: Physically-indexed Physically-tagged Cache

Virtually-Indexed Virtually-Tagged (VIVT)

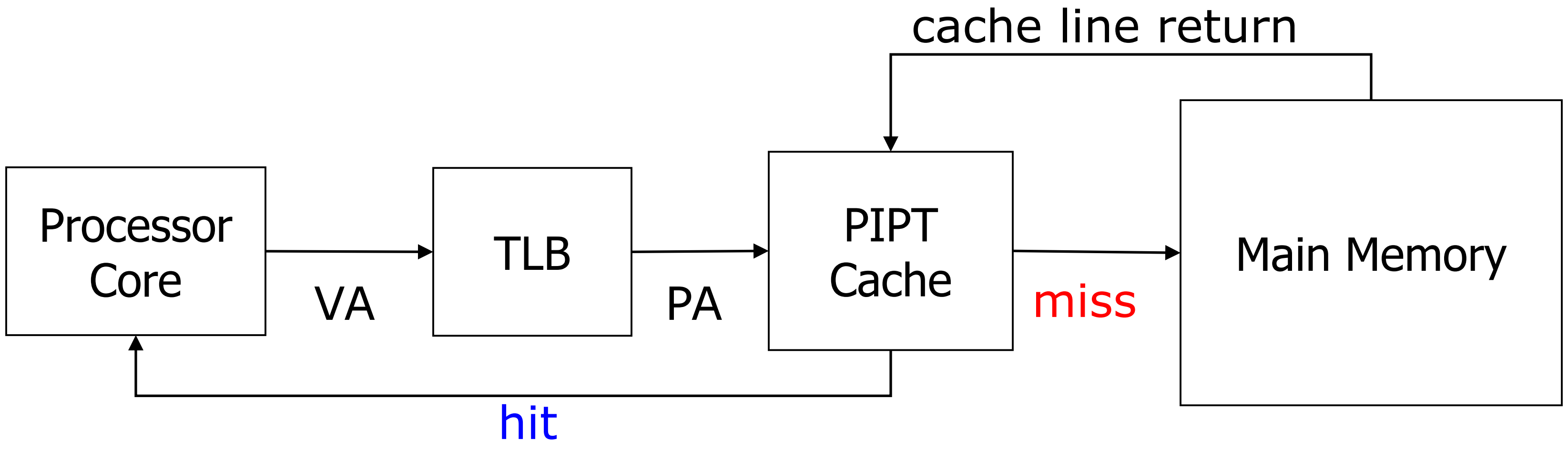


- Fast cache access
- Only require address translation when going to memory (miss)
- Issues?

VIVT Cache Issues - Aliasing

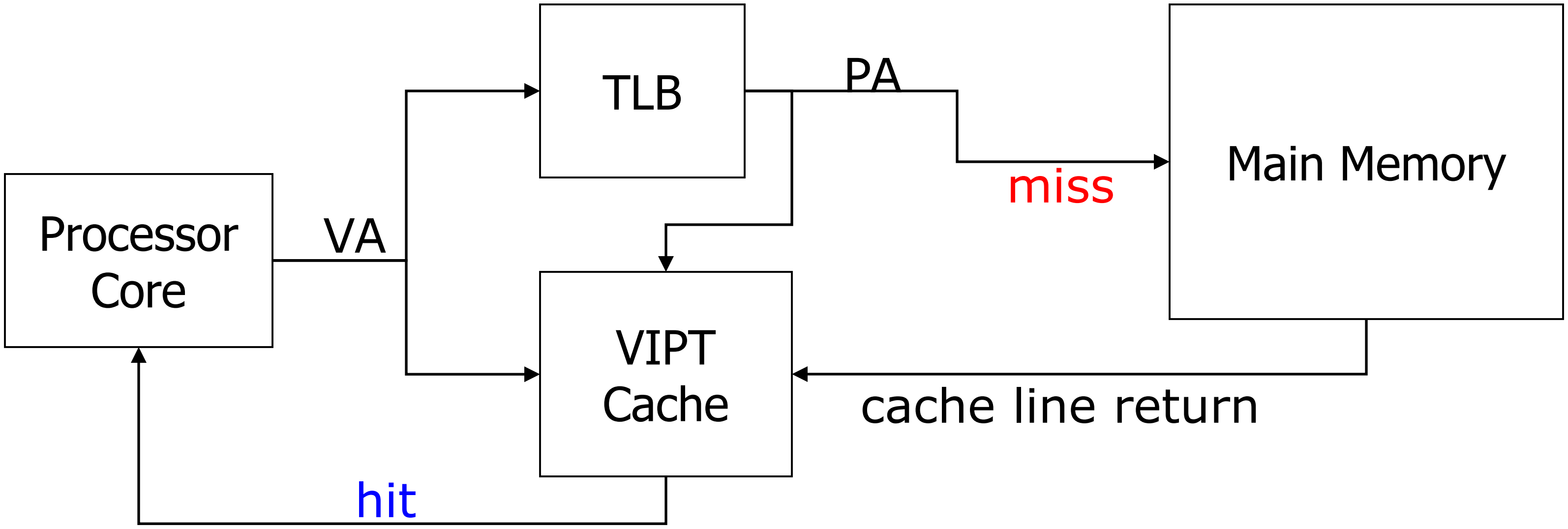
- Homonym
  - Same VA maps to different PAs
  - Occurs when there is a context switch
  - Solutions
    - Include process id (PID) in cache or
    - Flush cache upon context switches
- Synonym (also a problem in VIPT)
  - Different VAs map to the same PA
  - Occurs when data is shared by multiple processes
  - Duplicated cache line in VIPT cache and VIVT$ w/ PID
  - Data is inconsistent due to duplicated locations
  - Solution
    - Can Write-through solve the problem?
    - Flush cache upon context switch
    - If (index+offset) < page offset, can the problem be solved? (discussed later in VIPT)

Physically-Indexed Physically-Tagged (PIPT)



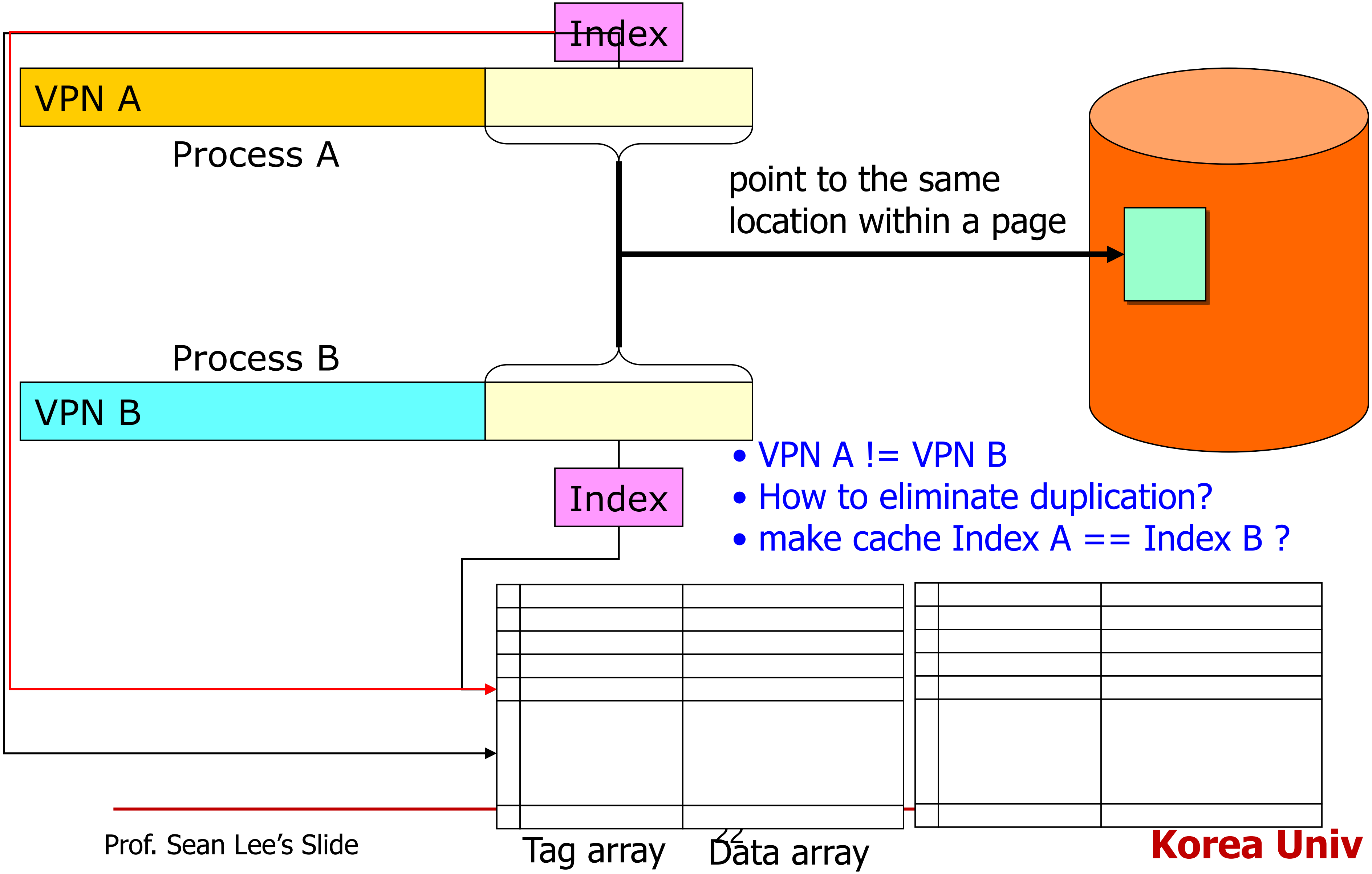
- Slower, always translate address before accessing memory
- Simpler for data coherence

Virtually-Indexed Physically-Tagged (VIPT)



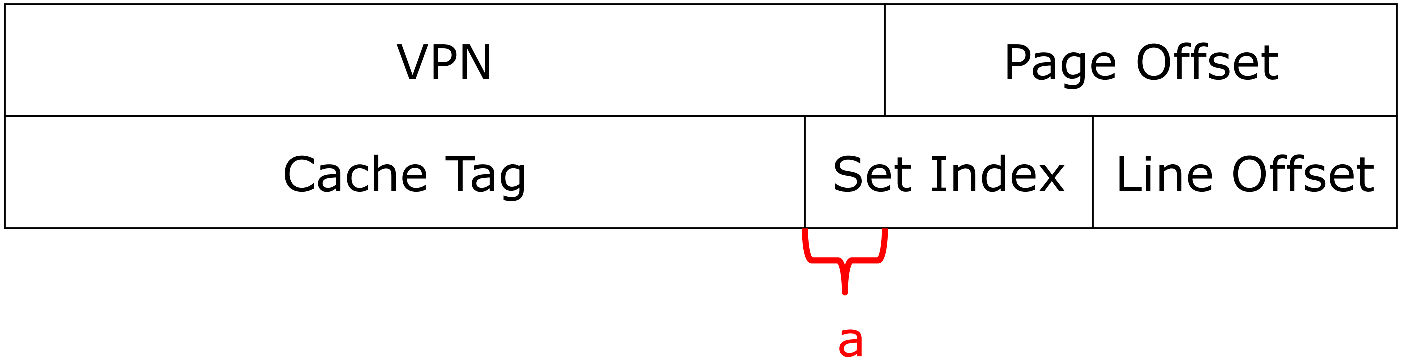
- Gain benefit of a VIVT and PIPT
- Parallel Access to TLB and VIPT cache
- No Homonym
- How about Synonym?

Deal w/ Synonym in VIPT Cache



- VPN A != VPN B
- How to eliminate duplication?
- make cache Index A == Index B ?

Synonym in VIPT Cache



VPN

Page Offset

Cache Tag

Set Index

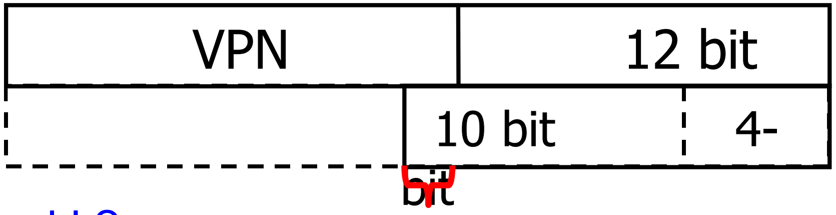
Line Offset

a

- If two VPNs do not differ in a then there is no synonym problem, since they will be indexed to the same set of a VIPT cache
- Imply # of sets cannot be too big
- Max number of sets = page size / cache line size
  - Ex: 4KB page, 32B line, max set = 128
- A complicated solution in MIPS R10000

R10000’s Solution to Synonym

- 32KB 2-Way Virtually-Indexed L1



a= VPN[1:0] stored as part of L2 cache Tag

- Direct-Mapped Physical L2
  - L2 is *Inclusive* of L1
  - VPN[1:0] is appended to the “tag” of L2
- Given two virtual addresses *VA1* and *VA2* that differs in *VPN[1:0]* and both map to the same physical address *PA*
  - Suppose *VA1* is accessed first so blocks are allocated in L1&L2
  - What happens when *VA2* is referenced?

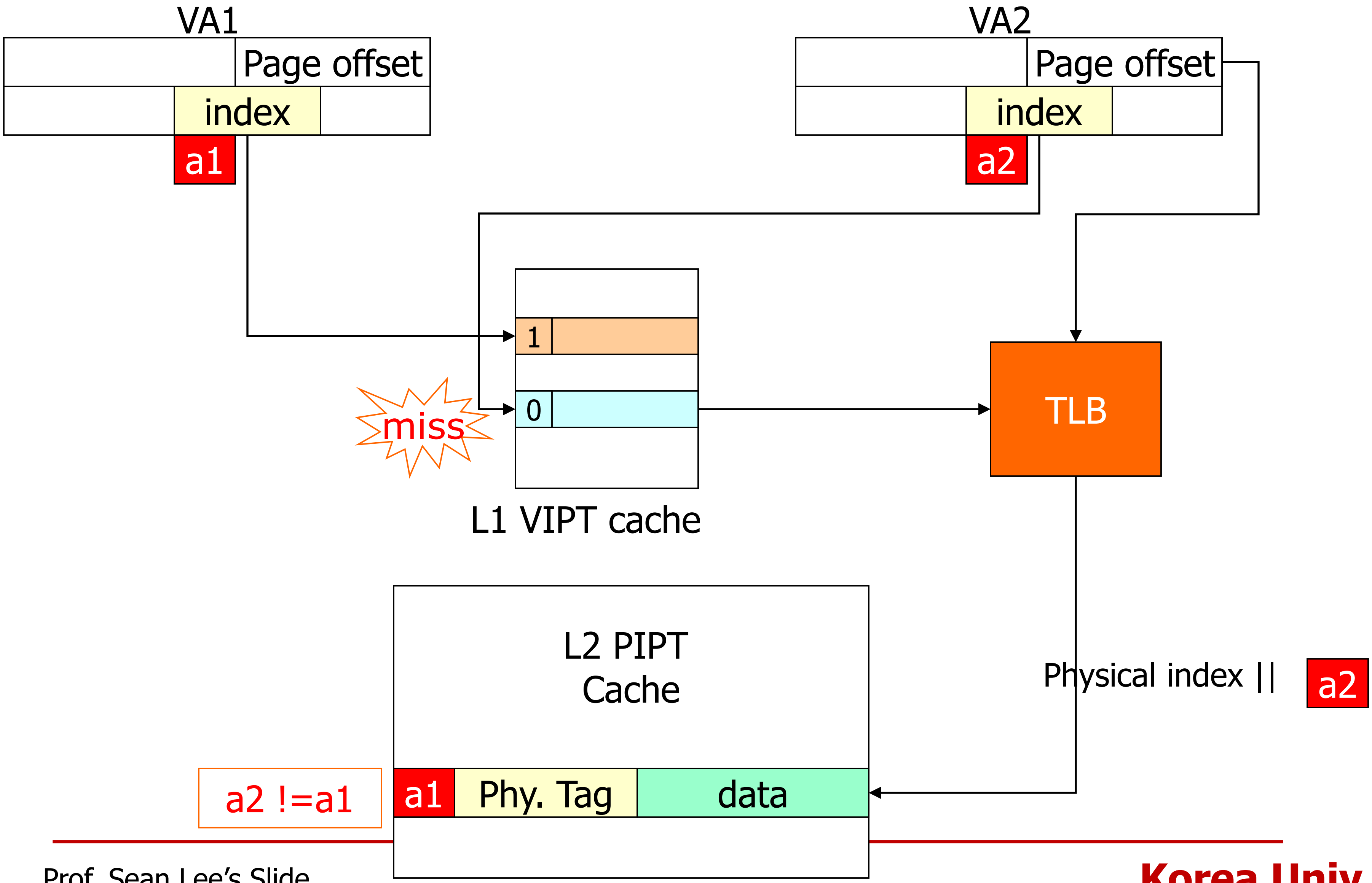
    1 *VA2* indexes to a different block in L1 and misses

    2 *VA2* translates to *PA* and goes to the same block as *VA1* in L2

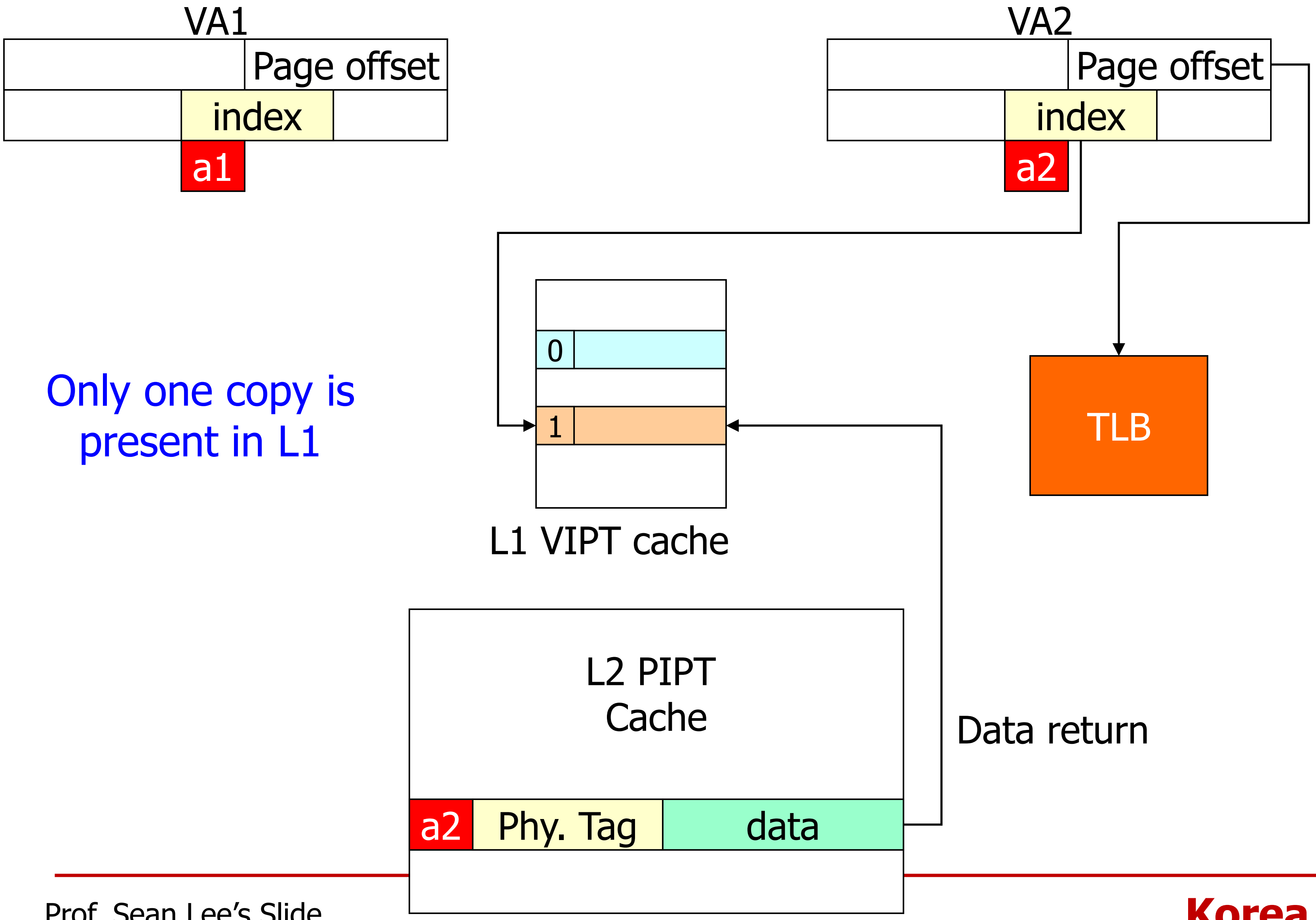
    3. Tag comparison fails (since *VA1*[1:0]≠*VA2*[1:0])

    4. Treated just like as a L2 conflict miss ⇒ *VA1*’s entry in L1 is ejected (or dirty-written back if needed) due to inclusion policy

Deal w/ Synonym in MIPS R10000



Deal w/ Synonym in MIPS R10000



Only one copy is present in L1