HW6

author: 111062272 蕭登鴻

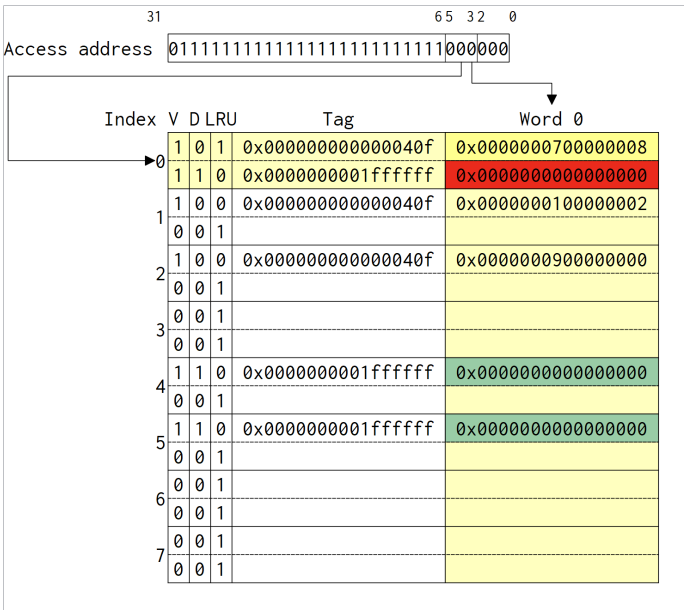
1

Notice: Since cache access happens at stage `MEM`, instruction executed in picture `after` is at stage `MEM`.

case a

before:

after:



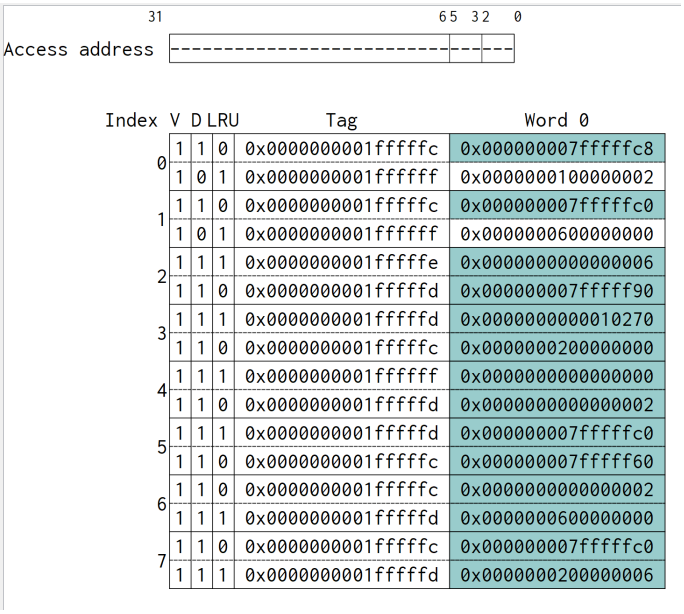
instructions:

```
1032c:      03010413        addi x8 x2 48
10330:      000107b7        lui x15 0x10
10334:      3c078793        addi x15 x15 960
10338:      0007b603        ld x12 0 x15
1033c:      0087b683        ld x13 8 x15
10340:      0107b703        ld x14 16 x15       WB
10344:      fcc43823        sd x12 -48 x8       MEM
10348:      fcd43c23        sd x13 -40 x8       EX
1034c:      fee43023        sd x14 -32 x8       ID
10350:      0187a783        lw x15 24 x15       IF
10354:      fef42423        sw x15 -24 x8
10358:      00700793        addi x15 x0 7
1035c:      fef42623        sw x15 -20 x8
```

Explanation: Set 0 has 1 already-occupied block when a write-allocate (write miss) happens. Therefore, dirty bit is set to 1, and the ref. bit of two ways is swapped.

case b

before:



Index	V	D	LRU	Tag	Word 0
0	1	1	0	0x0000000001fffffc	0x000000007fffffc8
	1	0	1	0x0000000001ffffff	0x0000000100000002
1	1	1	0	0x0000000001fffffc	0x000000007fffffc0
	1	0	1	0x0000000001ffffff	0x0000000600000000
2	1	1	1	0x0000000001fffffe	0x0000000000000006
	1	1	0	0x0000000001fffffd	0x000000007fffff90
3	1	1	1	0x0000000001fffffd	0x0000000000010270
	1	1	0	0x0000000001fffffc	0x0000000200000000
4	1	1	1	0x0000000001ffffff	0x0000000000000000
	1	1	0	0x0000000001fffffd	0x0000000000000002
5	1	1	1	0x0000000001fffffd	0x000000007fffffc0
	1	1	0	0x0000000001fffffc	0x000000007fffff60
6	1	1	0	0x0000000001fffffc	0x0000000000000002
	1	1	1	0x0000000001fffffd	0x0000000600000000
7	1	1	0	0x0000000001fffffc	0x000000007fffffc0
	1	1	1	0x0000000001fffffd	0x0000000200000006

after:

Index	V	D	LRU	Tag	Word 0
0	1	1	1	0x0000000001fffffc	0x000000007fffffc8
	1	1	0	0x0000000001ffffff	0x0000000100000002
1	1	1	0	0x0000000001fffffc	0x000000007fffffc0
	1	0	1	0x0000000001ffffff	0x0000000600000000
2	1	1	1	0x0000000001fffffe	0x0000000000000006
	1	1	0	0x0000000001fffffd	0x000000007fffff90
3	1	1	1	0x0000000001fffffd	0x0000000000010270
	1	1	0	0x0000000001fffffc	0x0000000200000000
4	1	1	1	0x0000000001ffffff	0x0000000000000000
	1	1	0	0x0000000001fffffd	0x0000000000000002
5	1	1	1	0x0000000001fffffd	0x000000007fffffc0
	1	1	0	0x0000000001fffffc	0x000000007fffff60
6	1	1	0	0x0000000001fffffc	0x0000000000000002
	1	1	1	0x0000000001fffffd	0x0000000600000000
7	1	1	0	0x0000000001fffffc	0x000000007fffffc0
	1	1	1	0x0000000001fffffd	0x0000000200000006

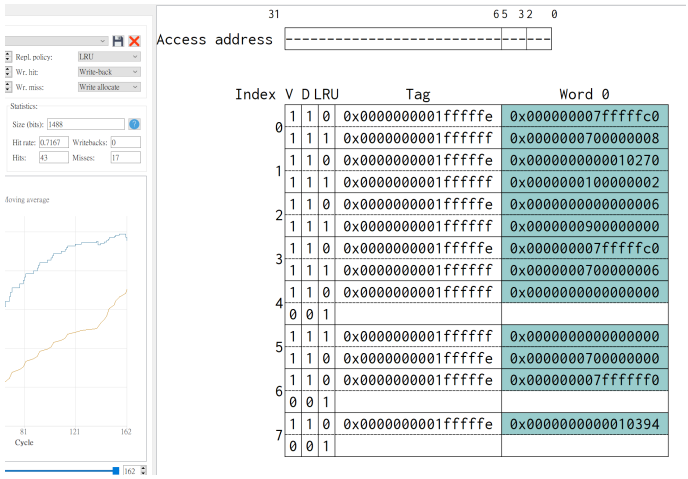
instructions:

```
100cc:      fef42623      sw x15 -20 x8
100d0:      fd043783      ld x15 -48 x8
100d4:      0007a703      lw x14 0 x15
100d8:      fd843783      ld x15 -40 x8
100dc:      00e7a023      sw x14 0 x15          MEM
100e0:      fd043783      ld x15 -48 x8          EX
100e4:      fec42703      lw x14 -20 x8          ID
100e8:      00e7a023      sw x14 0 x15           IF
100ec:      00000013      addi x0 x0 0
100f0:      02813403      ld x8 40 x2
```

Explanation: Way 1 of set 0 becomes dirty when a write hit happens. Therefore, dirty bit is changed from 0 to 1, and the ref. bit of two ways is swapped.

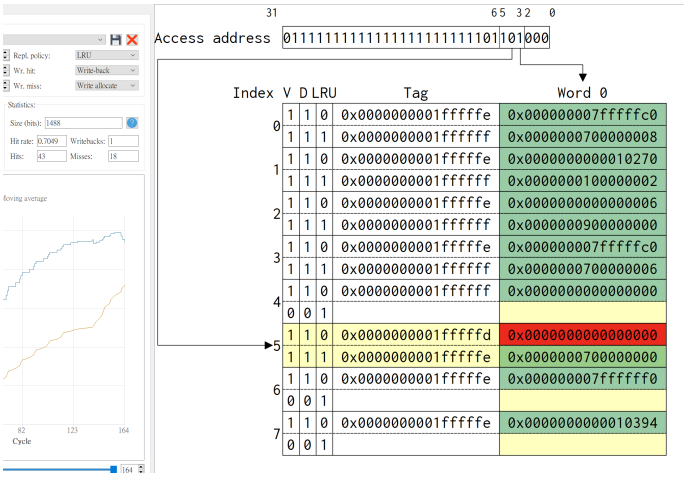
case c

before:



Index	V	D	LRU	Tag	Word 0
0	1	1	0	0x0000000001fffffe	0x000000007fffffc0
	1	1	1	0x0000000001ffffff	0x0000000700000008
1	1	1	0	0x0000000001fffffe	0x0000000000010270
	1	1	1	0x0000000001ffffff	0x0000000100000002
2	1	1	0	0x0000000001fffffe	0x0000000000000006
	1	1	1	0x0000000001ffffff	0x0000000900000000
3	1	1	0	0x0000000001fffffe	0x000000007fffffc0
	1	1	1	0x0000000001ffffff	0x0000000700000006
4	1	1	0	0x0000000001ffffff	0x0000000000000000
	0	0	1		
5	1	1	1	0x0000000001ffffff	0x0000000000000000
	1	1	0	0x0000000001fffffe	0x0000000700000000
6	1	1	0	0x0000000001fffffe	0x000000007ffffff0
	0	0	1		
7	1	1	0	0x0000000001fffffe	0x0000000000010394
	0	0	1		

after:



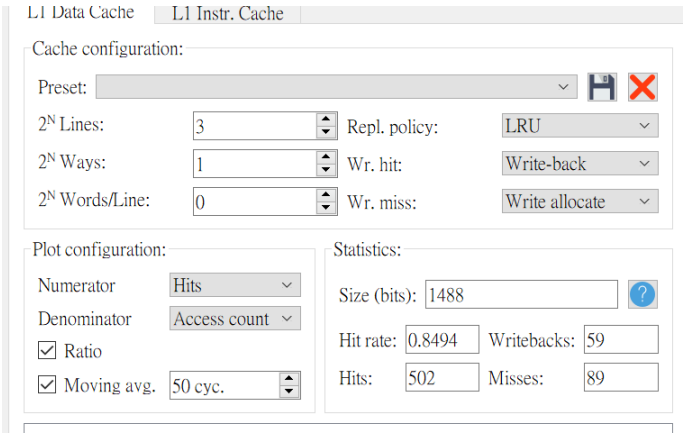
instructions:

```
100f4:      03010113        addi x2 x2 48
100f8:      00008067        jalr x0 x1 0
100fc:      fd010113        addi x2 x2 -48
10100:      02113423        sd x1 40 x2
10104:      02813023        sd x8 32 x2
10108:      03010413        addi x8 x2 48       WB
1010c:      fca43c23        sd x10 -40 x8       MEM
10110:      00058793        addi x15 x11 0      EX
10114:      00060713        addi x14 x12 0      ID
10118:      fcf42a23        sw x15 -44 x8       IF
1011c:      00070793        addi x15 x14 0
10120:      fcf42823        sw x15 -48 x8
10124:      fd042783        lw x15 -48 x8
```

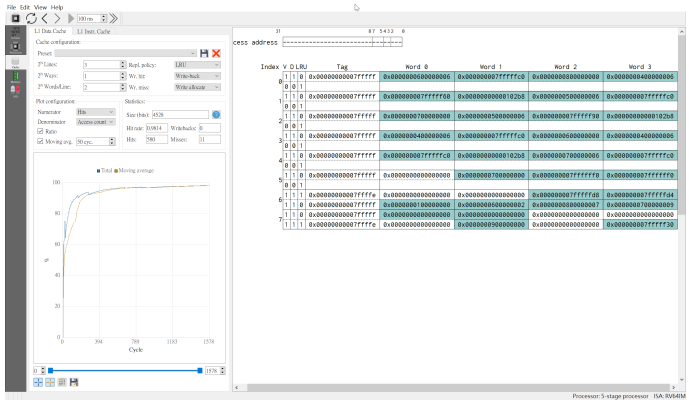
Explanation: A write miss happens at way 0 of set 5, which is marked dirty. Therefore, a write back is performed according to the WB counter on the left; dirty bit is changed from 0 to 1; the ref. bit of two ways is swapped.

d

original:



own-design:



Explanation: Owning to the frequent-array-accessing nature in quicksort algorithm, increasing the parameter `Words/Line` accords to the increasing ability of handling spacial locality; therefore increase the hit rate from 84.94% to 98.14%.

2

a

encoding	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10
bit pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14
bit value			1		0	0	1		0	1	1	0	0	1
p1	x		x		x		x		x		x		x	
p2		x	x			x	x			x	x			x
p4				x	x	x	x					x	x	x
p8								x	x	x	x	x	x	x

(under xor addition)

p1 = 1 + 0 + 1 + 0 + 1 + 0 = 1

p2 = 1 + 0 + 1 + 1 + 1 + 1 = 1

p4 = 0 + 0 + 1 + 0 + 0 + 1 = 0

p8 = 0 + 1 + 1 + 0 + 0 + 1 = 1

C\{p11} = 11 1 0 001 1 011001

p11 = sum(11 1 0 001 1 011001) = 0

=> C = 11 1 0 001 1 011001 0

b

```
No errors
```

decoding	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	p11
bit pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bit value	1	1	1	0	0	0	1	1	0	1	1	0	0	1	0
p1	x		x		x		x		x		x		x		
p2		x	x			x	x			x	x			x	
p4				x	x	x	x					x	x	x	
p8								x	x	x	x	x	x	x	

h1 = 1 + 1 + 0 + 1 + 0 + 1 + 0 = 0

h2 = 1 + 1 + 0 + 1 + 1 + 1 + 1 = 0

h4 = 0 + 0 + 0 + 1 + 0 + 0 + 1 = 0

h8 = 1 + 0 + 1 + 1 + 0 + 0 + 1 = 0

hn = 0

=> No errors

c

```
Suppose d5 of C is inverted
```

decoding	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	p11
bit pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bit value	1	1	1	0	0	0	1	1	1	1	1	0	0	1	0
p1	x		x		x		x		x		x		x		
p2		x	x			x	x			x	x			x	
p4				x	x	x	x					x	x	x	
p8								x	x	x	x	x	x	x	

h1 = 1 + 1 + 0 + 1 + 0 + 1 + 0 = 1

h2 = 1 + 1 + 0 + 1 + 1 + 1 + 1 = 0

h4 = 0 + 0 + 0 + 1 + 0 + 0 + 1 = 0

h8 = 1 + 0 + 1 + 1 + 0 + 0 + 1 = 1

hn = 1

=> Single error in pos 1001_2 = 9: d5

d

```
Suppose p1 and d8 of C are inverted
```

decoding	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	p11
bit pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bit value	0	1	1	0	0	0	1	1	0	1	1	1	0	1	0
p1	x		x		x		x		x		x		x		
p2		x	x			x	x			x	x			x	
p4				x	x	x	x					x	x	x	
p8								x	x	x	x	x	x	x	

h1 = 1 + 1 + 0 + 1 + 0 + 1 + 0 = 1

h2 = 1 + 1 + 0 + 1 + 1 + 1 + 1 = 0

h4 = 0 + 0 + 0 + 1 + 0 + 0 + 1 = 1

h8 = 1 + 0 + 1 + 1 + 0 + 0 + 1 = 1

hn = 0

=> Double error detected

e

decoding	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	p11
bit pos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bit value	0	1	1	0	0	0	1	1	0	1	1	1	0	1	1
p1	x		x		x		x		x		x		x		
p2		x	x			x	x			x	x			x	
p4				x	x	x	x					x	x	x	
p8								x	x	x	x	x	x	x	

h1 = 1 + 1 + 0 + 1 + 0 + 1 + 0 = 1

h2 = 1 + 1 + 0 + 1 + 1 + 1 + 1 = 0

h4 = 0 + 0 + 0 + 1 + 0 + 0 + 1 = 1

h8 = 1 + 0 + 1 + 1 + 0 + 0 + 1 = 1

hn = 1

=> Single error at pos 1101_2 = 13: d9

=> In fact, however, not at d9 (error correction/detection failed)

f

Since p = 8 is the minimum solution to:

$$2^p >= p + 128 + 1$$

plus one additional parity bit => a total of 9 parity bits.

3

a

Size of a single block: $2^{3\ 2} \cdot 2^{1\ 0}\text{bytes} = 2^4\text{bytes} = 16\text{bytes} \equiv A$

b

Number of blocks: $2^{8\ 4} = 2^5 = 32 \equiv B$

c

Total cache size: $A \cdot B = 512\text{bytes}$

d

	V	D	Tag	Data
size	1b	1b	5b	16B

=> $2^5 \cdot (1 + 1 + 5 + 16 \cdot 8)\text{bits} = 540\text{bytes}$

e

Tag	Index	Block offset	byte offset
13:8	7:4	3:2	1:0

f

	R	V	D	Tag	Data
size	1b	2 * 1b	2 * 1b	2 * 6b	2 * 16B

=> $2^4 \cdot (1 + 2 \cdot (1 + 1 + 6 + 16 \cdot 8))\text{bits} = 546\text{bytes}$

4

3-level designs:

$$\text{L1} \rightarrow \left\{ \begin{matrix} \text{L2-DM} \\ \text{L2-4Way} \end{matrix} \right\} \rightarrow \text{L3-8Way} \rightarrow \text{Mem.}$$

2-level designs:

$$\text{L}_1 \rightarrow \left\{ \begin{matrix} \text{L2-DM} \\ \text{L2-4Way} \end{matrix} \right\} \rightarrow \text{Mem.}$$

```
Design symbol

A: L1 -> L2-DM -> L3-8Way
B: L1 -> L2-4Way -> L3-8Way
C: L1 -> L2-DM
D: L1 -> L2-4Way
```

a

```
Effective CPI

A: 1 + 5%(16 + 4%(50 + 2%(200))) = 1.908
B: 1 + 5%(20 + 3.5%(50 + 2%(200))) = 2.0945
C: 1 + 5%(16 + 4%(200)) = 2.2
D: 1 + 5%(20 + 3.5%(200)) = 2.35
```

b

denote $r = 4\%(50 + 2\%(200))$

=> $1 + 5\%(x + r) \leq 1.8$ => $x \leq 13.84$ => $x_{\max} = 13$

5

of bits for:

- offset: lg(16) = 4
- Index: lg(4) = 2
- Tag: 12 - 4 - 2 = 6

initial:

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	0			
1		0				0			
2		0				0			
3		0				0			

1. (# of no.) Read 0x340

   Tag/Index/Offset: 001101 00 0000

   => $miss, load

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	0	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1		0				0			
2		0				0			
3		0				0			

2. Read 0x000

   Tag/Index/Offset: 000000 00 0000

   => $hit

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1		0				0			
2		0				0			
3		0				0			

3. Read 0x1d8

   Tag/Index/Offset: 000111 01 1000

=> $miss, load

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	1	1	0	000111	Mem[0x1d0-1df]	0			
2		0				0			
3		0				0			

4. Write 0x354

   Tag/Index/Offset: 001101 01 0100

   => $miss, write allo., mark dirty

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2		0				0			
3		0				0			

5. Read 0xa61

   Tag/Index/Offset: 101001 10 0001

   => $miss, load

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2	1	1	0	101001	Mem[0xa60-a6f]	0			
3		0				0			

6. Write 0xa61

   Tag/Index/Offset: 101001 10 0001

=> $hit, mark dirty

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2	1	1	1	101001	Mem[0xa60-a6f]'	0			
3		0				0			

7. Read 0x3ec

   Tag/Index/Offset: 001111 10 1100

   => $miss, load

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2	0	1	1	101001	Mem[0xa60-a6f]'	1	0	001111	Mem[0x3e0-3ef]
3		0				0			

8. Read 0xa62

   Tag/Index/Offset: 101001 10 0010

   => $hit

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2	1	1	1	101001	Mem[0xa60-a6f]'	1	0	001111	Mem[0x3e0-3ef]
3		0				0			

9. Read 0x3ea

Tag/Index/Offset: 001111 10 1010

=> $hit

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2	0	1	1	101001	Mem[0xa60-a6f]'	1	0	001111	Mem[0x3e0-3ef]
3		0				0			

10. Read 0x422

    Tag/Index/Offset: 010000 10 0010

    => $miss, block replacement: way 0, write back happens; after: load

set	R	V0	D0	Tag0	Data0	V1	D1	Tag1	Data1
0	1	1	0	000000	Mem[0x000-00f]	1	0	001101	Mem[0x340-34f]
1	0	1	0	000111	Mem[0x1d0-1df]	1	1	001101	Mem[0x350-35f]'
2	1	1	0	010000	Mem[0x420-42f]	1	0	001111	Mem[0x3e0-3ef]
3		0				0			

6

a

page size: 512 B = 2^9 B

=> offset: 9 bits

b

16 - 9 = 7 => $2^7 = 128$ virtual pages

c

14 - 9 = 5 => $2^5 = 32$ virtual pages

d

$2^7 \cdot 4$ B = 2^9 B = 512 B

e

tag bits: 7 - 4 = 3

	V	D	R	Tag	phy-page num.
size	1b	1b	1b	3b	5b

=> $2^4(11) = 176$ b

7

4KB = 2^2*2^10 Bytes = 2^12 bytes => page offset = 12

=> Tag size = 4*4 - 12 = 4

initial state:

TLB:

V	Tag	phy-page num.	R	Replacement Pointer
1	0x4	6	1	V
1	0x1	2	0	
1	0xa	3	1	
0	0x3	5	0	

Page table:

index	V	phy-page num or in disk
0	0	D
1	1	2
2	0	D
3	1	5
4	1	6
5	1	11
6	1	7
7	0	D

index	V	phy-page num or in disk
8	0	D
9	0	D
a	1	3
b	0	D

1. 0x5368

   tag: 0x5

   => $miss, no page fault

   section in page table:

index	V	phy-page num or in disk
5	1	11

   TLB:

V	Tag	phy-page num.	R	Replacement Pointer
1	0x4	6	1	V
1	0x1	2	0	
1	0xa	3	1	
1	0x5	11	1	

2. 0x02c3

   tag: 0x0

   => $miss + replacement, page fault

   updated section in page table:

index	V	phy-page num or in disk
0	1	1

   TLB:

V	Tag	phy-page num.	R	Replacement Pointer
1	0x4	6	0	
1	0x0	1	1	

V	Tag	phy-page num.	R	Replacement Pointer
1	0xa	3	1	V
1	0x5	11	1	

3. 0x434b

   tag: 0x4

   => $hit, reset others's R

V	Tag	phy-page num.	R	Replacement Pointer
1	0x4	6	1	
1	0x0	1	0	
1	0xa	3	0	V
1	0x5	11	0	

4. 0x6812

   tag: 0x6

   => $miss + replacement, no page fault

   section in page table:

index	V	phy-page num or in disk
6	1	7

   TLB:

V	Tag	phy-page num.	R	Replacement Pointer
1	0x4	6	1	
1	0x0	1	0	
1	0x6	7	1	
1	0x5	11	0	V

5. 0xaf50

   tag: 0xa

   => $miss + replacement, no page fault

   section in page table:

index	V	phy-page num or in disk
a	1	3

TLB:

V	Tag	phy-page num.	R	Replacement Pointer
1	0x4	6	1	V
1	0x0	1	0	
1	0x6	7	1	
1	0xa	3	1	