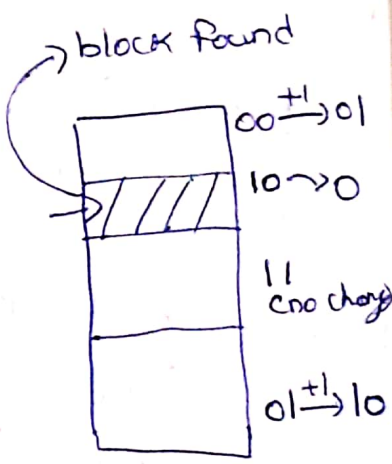
✱) Replacement Algorithm (Hamacher)

Least Recently Used (LRU)

referenced
•) Replace a cache line (block) which has not been used for longest period of time.

•) Assuming k-way cache

✱) Hit occurs, Set the Counter value zero(0) for the newly referenced block and increment the counter value of all blocks which have value lesser than the 'old' value of the referenced page (block)



Here K=4

Above is a figure of set selected.

.) Miss occurs and set is not full :-

Counter associated with ~~block is set~~ just referenced block is set to zero (0) and counters of all other blocks are incremented by 1.

.) Miss occurs and set is full :-

Identify the block associated with highest value (above 11 in 4-way cache) and replace it, Subsequently counter is set to zero (0) and Counters of all other blocks are incremented by one.

(Another replacement policy: FIFO (first in-first out)

"Recent past is good indicator of near future" gives us idea that LRU is better than FIFO

Dt-11-03-2020

Example of Mapping Techniques and LRU Replacement Policy (Hamacher)

Problem: A Computer has separate data and instruction caches The data cache has 8 blocks. Each block can hold 1 word of 16 bits.
Consider the program given in figure to identify the Cache content in various mapping schemes.

Sol:)

```
int A[4][10];
int sum=0;
for(j=0; j<10; j++)
   sum= sum + A[0][j];
AVG= sum/10;
for(i=9; i>=0; i--)
   A[0][i]= A[0][i] / AVG;
```

Normalizing first row.

•) Consider the Array A is organized (in main memory) in column order, and first element A[0][0] is at location 7A00

•) Variable sum, i, j are in processor's registers (Available, not required to access MM)

•) It is a 4-way set Associative mapping.

39

10+10+7

*) In Binary (1st location) 7A00 => 0111 1010 0000 0000

7A00	0111 1010 0000 0000	A[0][0]
7A01	0111 1010 0000 0001	A[1][0]
7A02	0111 1010 0000 0010	A[2][0]
7A03	0111 1010 0000 0011	A[3][0]
7A04	0111 1010 0000 0100	A[0][1]
7A05	0111 1010 0000 0101	A[1][2]
7A06	0111 1010 0000 0110	A[2][2]
7A07	0111 1010 0000 0111	A[3][1]
		A[0][2]
		A[1][2]
		⋮
7A27	0111 1010 0010 0111	A[3][9]

39th element

Total 40 elements

Mapping scheme:-

7A22 0111 1010 0010 0111

Direct mapping :- ← Tag (13 bits) → ↔ cache line (3 bits) → 8 blocks in cache

Associative :- ← Tag (16 bits) →

set-Associative :- ← Tag (15 bits) → ↔ #set = (1 bit)

(4-way) set-Associative
#set = 8/4 = 2.

Here:- As 1 block contains 1 word :

∴ No of bits for word field :- $(2)^0 = 0$

Content of cache after Pass → Content of cache after j = 1 and so on

Cache block	j = 1	j = 3	j = 5	j = 7	j = 9	i = 9
0	A[0][0]	A[0][2]	A[0][4]	A[0][6]	A[0][8]	
1						
2						
3						
4	A[0][1]	A[0][3]	A[0][5]	A[0][7]	A[0][9]	
5						
6						
7						

Eg:- A[0][0] = 0 000 → 0th block

+ 100 (Add 4) => As Column-order

A[0][1] = 0 100 → 4th block

+ 100

A[0][2] = 0 000 → 0th block

+ 100

A[0][3] = 0 100 → 4th block. and so on.

Now:-

next loop: i=9 (initially)

(we want location A[0][9] → It is directly in the 4th block (so its a hit)

•) This is the 1st hit as all were brought from mm so there was cache misses in last 10 times in bringing A[0][0] .... A[0][9] to cache.

•) next i=8 :- we want A[0][8] → we have A[0][8] is 0th block.

•) i=7 ; we want A[0][7] → we have A[0][7] in 4th block

•) Either we bring it back from mm

-> Or it is already there in 4th block, as it is direct mapping.

Total : 2 bits then miss

Associative mapping

•) A[0][0] can be kept anywhere, A[0][1] can be kept anywhere and so on.

•) while replacing, replace the least recently used location.

Content of cache after pass.

Cache block	j=7	j=9	i=0
0	A[0][0]	A[0][8]	A[0][0]
1	A[0][1]	A[0][9]	A[0][1]
2	A[0][2]	A[0][2]	A[0][2]
3	A[0][3]	A[0][3]	A[0][3]
4	A[0][4]	A[0][4]	A[0][4]
5	A[0][5]	A[0][5]	A[0][5]
6	A[0][6]	A[0][6]	A[0][6]
7	A[0][7]	A[0][7]	A[0][7]

Set 0

For A[0][8] in j=9 colum we replace A[0][0] as it is least recently used. It is victim page.
and for A[0][9] we use A[0][1] as victim page.

*) For i=9; A[0][9] is directly available in 1st block
So bit

Similary: for i=8, 7, 6, 5, 4, 3, 2 → bits

So total 8 bits.

i=1 we need A[0][1]; (replace with A[0][1] least recently used is A[0][9] (victim page)

i=0 we need A[0][0]; least recently used is A[0][8] (victim page) is replaced with A[0][0]

Set. Associative mapping

Content of Cache after pass

Cache block	j=3	j=7	j=8	i=5
Set 0: 0	A[0][0]	A[0][4]	A[0][8]	A[0][4]
1	A[0][1]	A[0][5]	A[0][9]	A[0][5]
2	A[0][2]	A[0][6]	A[0][6]	A[0][6]
3	A[0][3]	A[0][7]	A[0][7]	A[0][7]
Set 1: 4				
5				
6				
7				

Set field is last bit.

So all of A[0][0], A[0][1], ... A[0][9] are in set 0 as placed in column order.

↳ last bit of address is zero.

After j=3, we have the first column.

After j=7; for A[0][4] we have to replace A[0][0] as it was least recently used and so on.