Major Assignment COL216 Date:06-09-2020

LOKESH ACHARYA

2018CS10352

Replacement Policy:

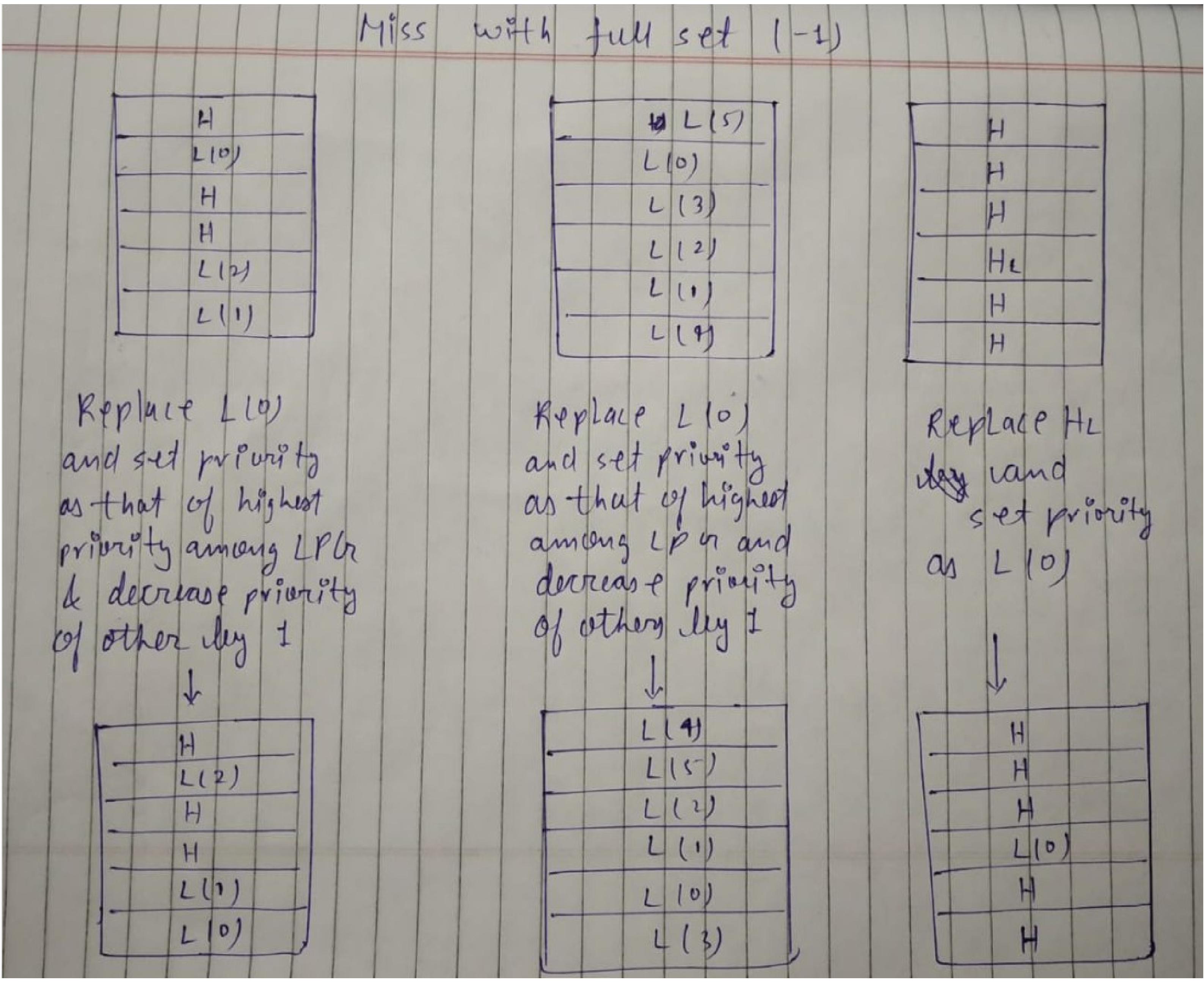
Every block can be treated as a High Priority or as a Low Priority block. i.e. Variable size of HPG and LPG which can change at runtime. This implemetation do not discart any unsed element form any priority set which may have been used for spacial locality.

Case: Miss

Subcase 1: Set full

If: all blocks are of HPG then replace with block lowest among them after cheking Dirty status with priority as 0.

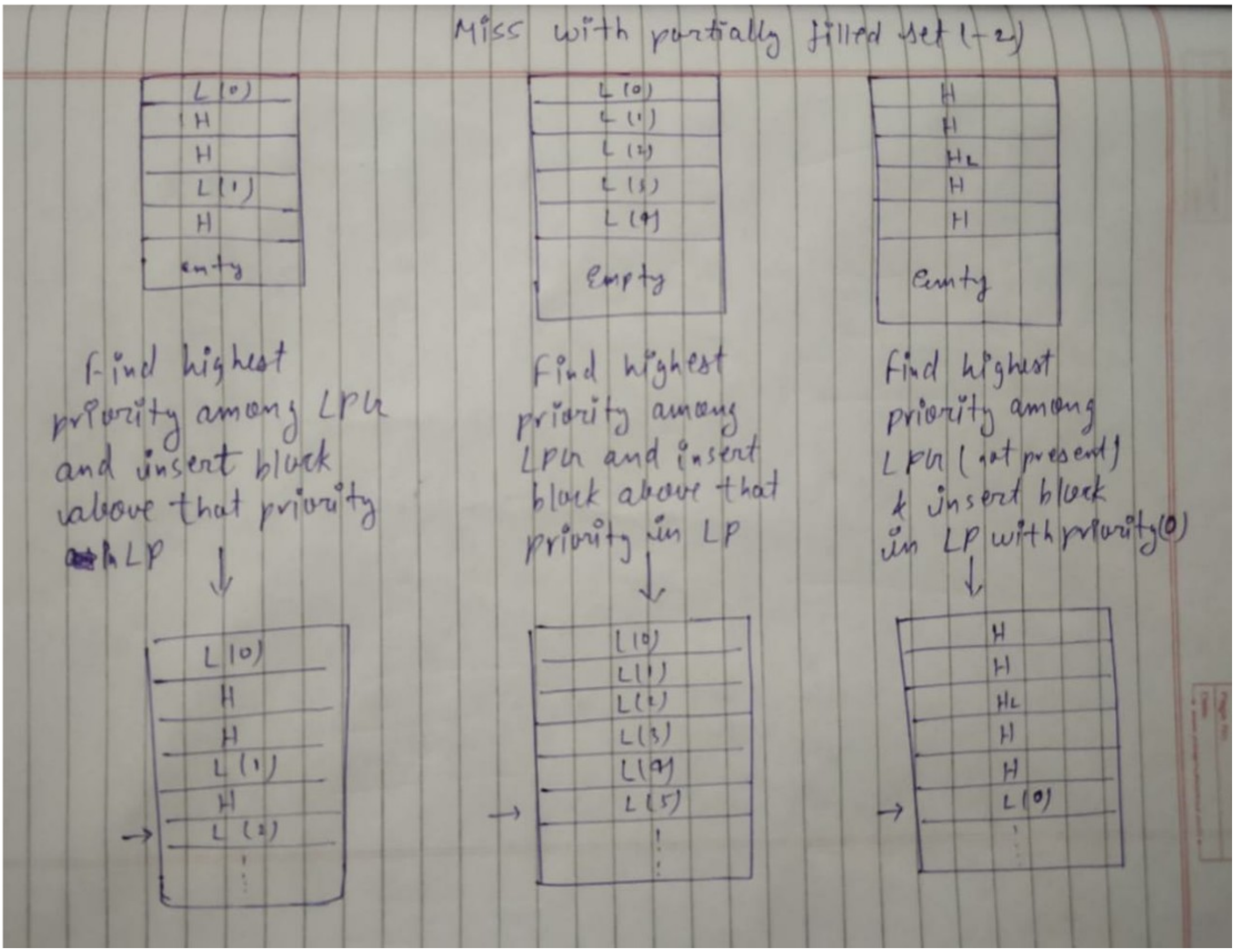
If: Replace element of LPG with priority 0 and set its priority highest among the LPG.



Miss with full set (1-1)
H
L(0)
H
H
L(2)
L(1)
Replace L(0) and set priority as that of highest priority among LPG & decrease priority of other by 1
H
L(2)
H
H
L(1)
L(0)
L(5)
L(0)
L(3)
L(2)
L(1)
L(4)
Replace L(0) and set priority as that of highest among LPG and decrease priority of others by 1
L(4)
L(5)
L(2)
L(1)
L(0)
L(3)
H
H
H
HL
H
H
Replace HL and set priority as L(0)
H
H
H
L(0)
H
H

Subcase 2: Partially filled

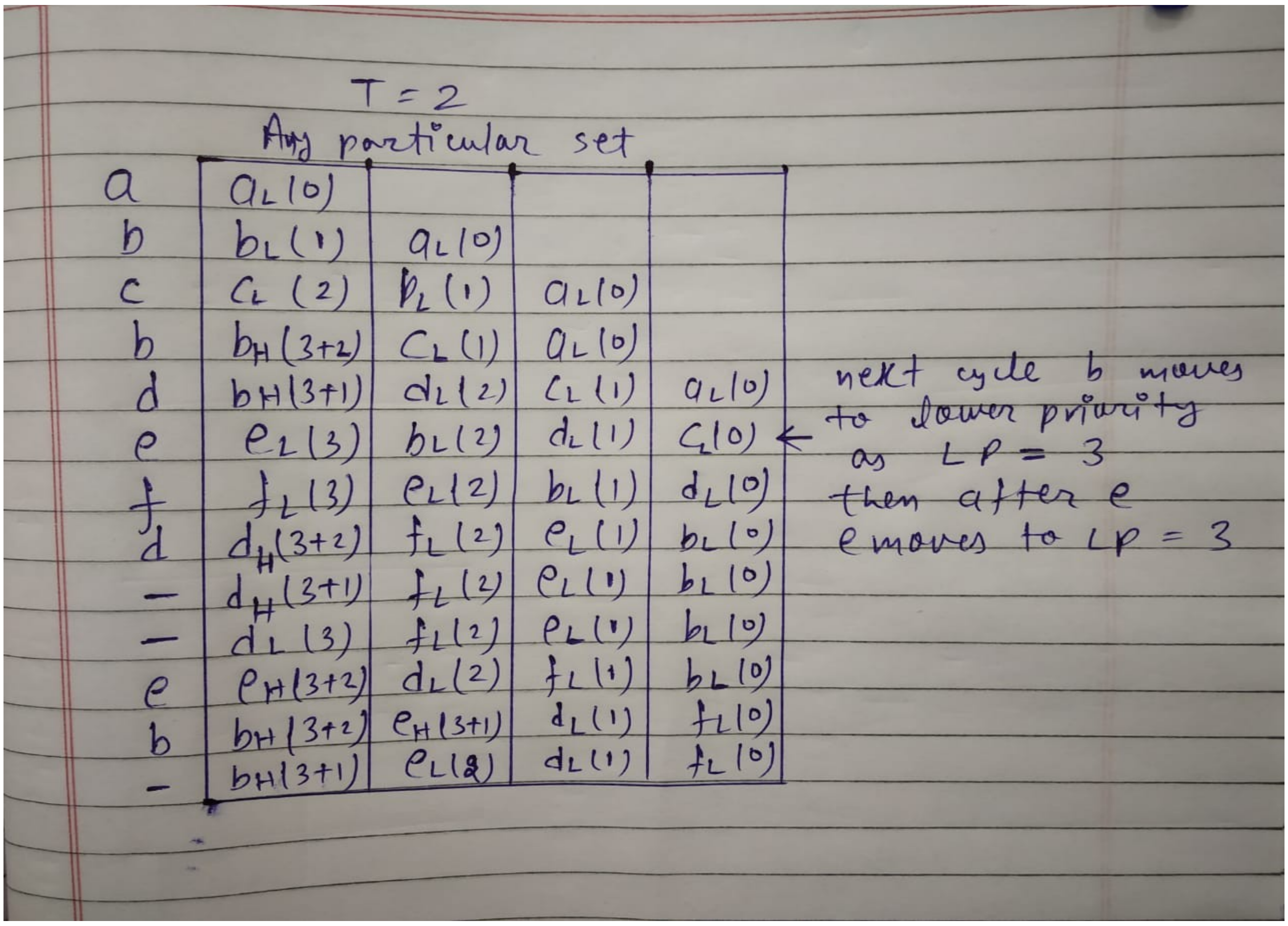
If: all valid blocks are in HPG then place the new one as LP with priority 0 in a new block.
If: place in new block with priority highest among LPG.


Miss with partially filled set (-2)
L(0)
H
H
L(1)
H
emty
Find highest priority among LPG and insert block above that priority in LP
L(0)
H
H
L(1)
H
L(2)
L(0)
L(1)
L(2)
L(3)
L(4)
Empty
Find highest priority among LPG and insert block above that priority in LP
L(0)
L(1)
L(2)
L(3)
L(4)
L(5)
H
H
HL
H
H
Emty
Find highest priority among LPG (not present) & insert block in LP with priority(0)
H
H
HL
H
H
L(0)

Case: Hit

If: Read is present then just retain the dirty status and add it to then Highest among HPG.
If: Write then overwrite the data section and dirty status as true.

Sample example:



T = 2

Any particular set

a	$a_L(0)$			
b	$b_L(1)$	$a_L(0)$		
c	$c_L(2)$	$b_L(1)$	$a_L(0)$	
b	$b_H(3+2)$	$c_L(1)$	$a_L(0)$	
d	$b_H(3+1)$	$d_L(2)$	$c_L(1)$	$a_L(0)$
e	$e_L(3)$	$b_L(2)$	$d_L(1)$	$c_L(0)$ ←
f	$f_L(3)$	$e_L(2)$	$b_L(1)$	$d_L(0)$
d	$d_H(3+2)$	$f_L(2)$	$e_L(1)$	$b_L(0)$
—	$d_H(3+1)$	$f_L(2)$	$e_L(1)$	$b_L(0)$
—	$d_L(3)$	$f_L(2)$	$e_L(1)$	$b_L(0)$
e	$e_H(3+2)$	$d_L(2)$	$f_L(1)$	$b_L(0)$
b	$b_H(3+2)$	$e_H(3+1)$	$d_L(1)$	$f_L(0)$
—	$b_H(3+1)$	$e_L(2)$	$d_L(1)$	$f_L(0)$

next cycle b moves to lower priority as LP = 3 then after e e moves to LP = 3

In this example when was not used for T(=2) times then it was put in LPG with priority 3 but as we got new request **e** it was placed above **b** in LPG.

When **d** was put in HPG and not used T(=2) times it was placed in LPG with highest priority among them.

When **b** was accessed again it was placed above e in HPG.

Functions used are:

int **powerOfTwo**(int &x)
:- *power of 2 in x*

int **check_tag**(**vector**<**tuple**<bool,int,int,bool,int>> cache,int s_index,int e_index, int tag)
:- *return index if tag matched and negative on no match (-1 if set is set is full and -2 if set is partially filled)*

int **find_invalid**(**vector**<**tuple**<bool,int,int,bool,int>> cache,int s_index,int e_index)
:- *ind index of first invalid index in a set*

tuple<bool,int,int,int,int,int> **INFO**(**vector**<**tuple**<bool,int,int,bool,int>> cache,int s_index,int e_index) :- *return (filled status, highest priority index among LPG, highest priority among LPG, lowest priority index among HPG, lowest priority among HPG)*

int **find_highest_among_LP**(**vector**<**tuple**<bool,int,int,bool,int>> cache,int s_index,int e_index)
:- *ind highest prority among LPG*

void **updateWB**(long long int* memo, **queue**<**tuple**<int,int,int>> &**WB**, int wbz, int P)
:- *update write buffer called on end of every cycle*

bool **wbfreed**(long long int* memo, **queue**<**tuple**<int,int,int>> WB, int wbz)
:- *check if the buffer is free or not*

bool **readStall**(int &sc, int P):- *facilitate stalling of read*

void **refreshCache**(**vector**<**tuple**<bool,int,int,bool,int>> &cache, int c_assoc, int T)
:- *refresh status of cache on end of every cycle -- updating priorty among HPG*

void **PRINTCACHE**(**vector**<**tuple**<bool,int,int,bool,int>> cache, int c_assoc, int num_sets)
:- Print the cache