Detecter\_Defaut\_Cache\_LRU( t\_cache mc , adresse @ ) : booleen

**Variables :**

index : entier

**Debut**

index := conv\_bin2dec (@(2 .. 4))

Pour i := 0 à 15 Faire

 Si mc[index][i].bit\_validite = '1' Alors

 Si mc[index][i].etiquette = @(5..10) Alors

 // MAJ du tag

 mc[index][i].tag\_LRU := 0

 // MAJ des tags

 Pour i := 0 à 15 Faire

 Si mc[index][i].bit\_validite = '1' Alors

 mc[index][i].tag\_LRU := mc[index][i].tag\_LRU + 1

 FinSi

 FinPour

 retourner Faux

 FinSi

 FinSi

FinPour

retourner Vrai

**Fin**

Detecter\_Defaut\_Cache\_LFU( t\_cache mc , adresse @) : booleen

**Variables :**

index : entier

**Debut**

index := conv\_bin2dec(@(2 .. 4))

Pour i := 0 à 15 Faire

 Si mc[index][i].bit\_validite = '1' Alors

 Si mc[index][i].etiquette = @(5..10) Alors

 // MAJ du tag

 mc[index][i].tag\_LRU := mc[index][i].tag\_LRU + 1

 retourner Faux

 FinSi

 FinSi

FinPour

retourner vrai

**Fin**

Charger\_En\_Memoire\_Cache\_LRU( t\_cache mc , main\_memory mm , adresse @ , t\_bloc b) :

**Variables :**

index , pos\_LRU : entier

@2 : adresse

**Debut**

 index := conv\_bin2dec(@(2..4))

pos\_LRU := 0

Pour i := 0 à 15 Faire

 // Cas ou la ligne n'est pas affmc[index]tée

 Si mc[index][i].bit\_validite = '0' Alors

 mc[index][i].bloc := b

 mc[index][i].etiquette := @(5..10)

 mc[index][i].tag\_LRU := 0

 mc[index][i].bit\_validite := '1'

 pos\_LRU := -1

 break // SORT DE LA BOUCLE POUR

 Sinon

 Si( mc[index][i].tag\_LRU > mc[index][pos\_LRU].tag\_LRU)Alors

 pos\_LRU := i

 FinSi

 FinSi

FinPour

// Si toutes les lignes étaient affectées

Si pos\_LRU <> -1 Alors

 // Si on a modifié le contenu du bloc, on le recharge en mm avant de l'écraser

 Si mc[index][pos\_LRU].maj = '1' Alors

 @2 := concat( mc[index][pos\_LRU].etiquette , @(2 .. 4) )

 pos\_mm := conv\_bin2dmc[index]( @2 )

 mm(pos\_mm) := mc[index][pos\_LRU].bloc

 FinSi

 // Ensuite on charge la nouvelle donnée

 mc[index][pos\_LRU].bloc := b

 mc[index][pos\_LRU].etiquette := @(5..10)

 mc[index][pos\_LRU].tag\_LRU := 0

FinSi

// MAJ des tags

Pour i := 0 à 15 Faire

 Si mc[index][i].bit\_validite = '1' Alors

 mc[index][i].tag\_LRU := mc[index][i].tag\_LRU + 1

 FinSi

FinPour

**Fin**

Charger\_En\_Memoire\_Cache\_LFU ( t\_cache mc , main\_memory mm , adresse @ ,

t\_bloc b) :

**variables :**

index , pos\_LFU : entier

@2 : adresse

**Debut**

 index := conv\_bin2dec(@(2..4))

pos\_LFU := 0

Pour i := 0 à 15 Faire

 // Cas ou la ligne n'est pas affectée

 Si mc[index][i].bit\_validite = '0' Alors

 mc[index][i].bloc := b

 mc[index][i].etiquette := @(5..10)

 mc[index][i].tag\_LFU := 1

 mc[index][i].bit\_validite := '1'

 pos\_LFU := -1

 break // SORT DE LA LOOP POUR

 Sinon

 Si( mc[index][i].tag\_LFU < mc[index][pos\_LFU].tag\_LFU)Alors

 pos\_LFU := i

 FinSi

 FinSi

FinPour

Si pos\_LFU <> -1 Alors

 // Si on a modifié le contenu du bloc, on le recharge en mm avant de l'écraser

 Si mc[index][pos\_LFU].maj = '1' Alors

 @2 := concat( mc[index][pos\_LFU].etiquette , @(2 .. 4) )

 pos\_mm := conv\_dmc[index]2bin( @2 )

 mm(pos\_mm) := mc[index][pos\_LFU].bloc

 FinSi

 // Ensuite on charge la nouvelle donnée

 mc[index][pos\_LFU].bloc := b

 mc[index][pos\_LFU].etiquette := @(5..10)

 mc[index][pos\_LFU].tag\_LFU := 1

FinSi

**Fin**