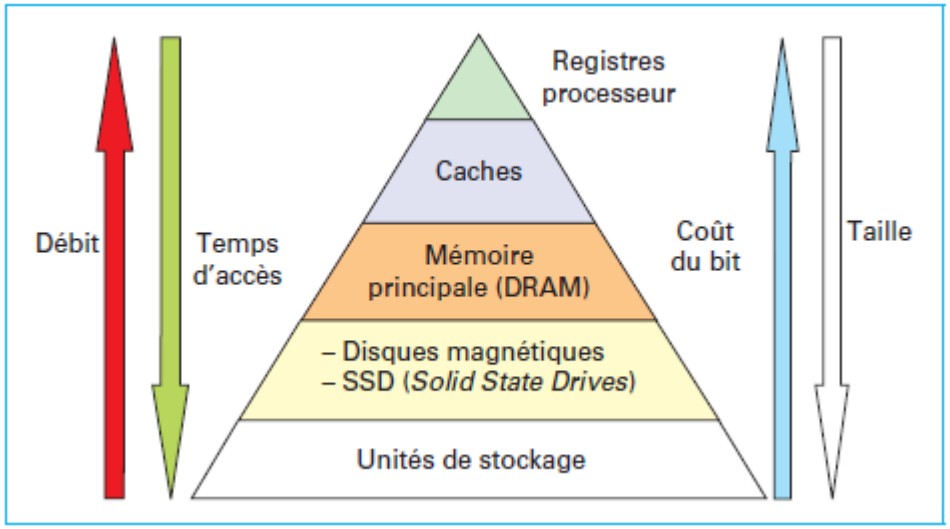
# Le principe de hiérarchie mémoire : les caches
# Les mémoires de l'ordinateur




*Techniques de l'ingénieur, H1002, hiérarchie mémoire : les caches*

# Hiérarchie Mémoire

vitesse — 6-35 ns — 70 - 120 ns

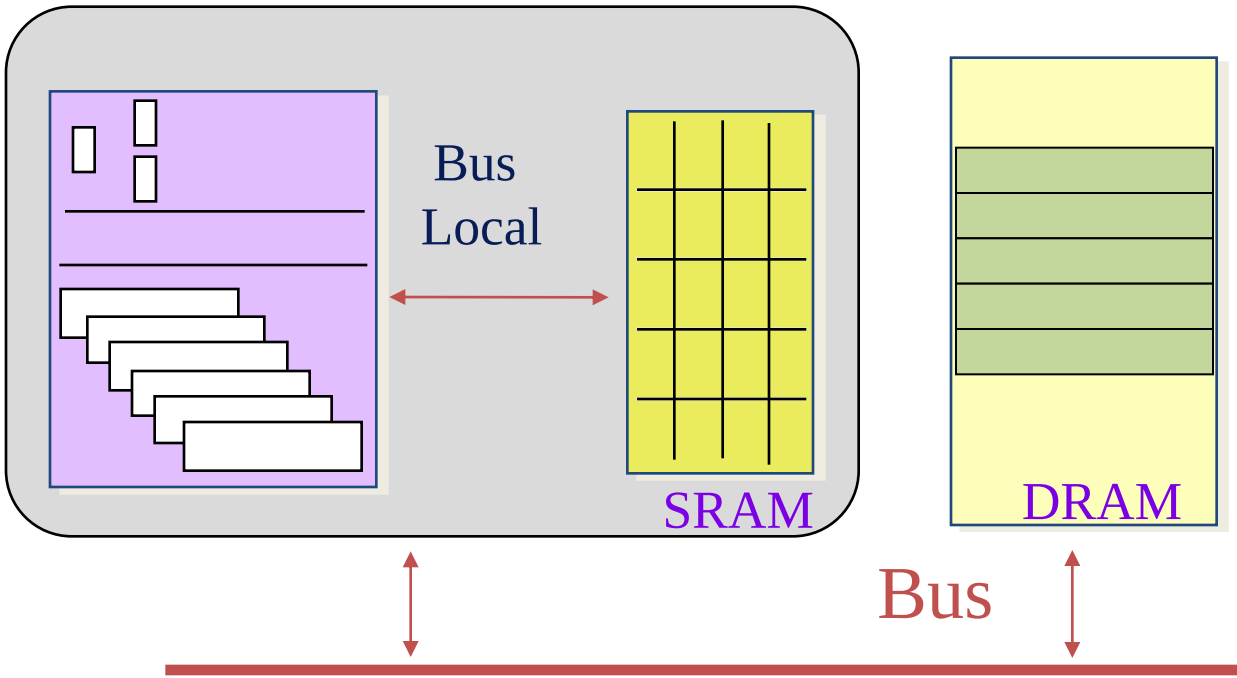
cout Le plus élevé — Le moins élevé

capacité La plus petite — La plus grande

Processeur Registres

Mémoire Cache

Mémoire centrale




La mémoire cache est une mémoire intermédiaire placée entre le processeur et la mémoire centrale dont le temps d'accès est de 4 à 20 fois inférieur à celui de la mémoire centrale.

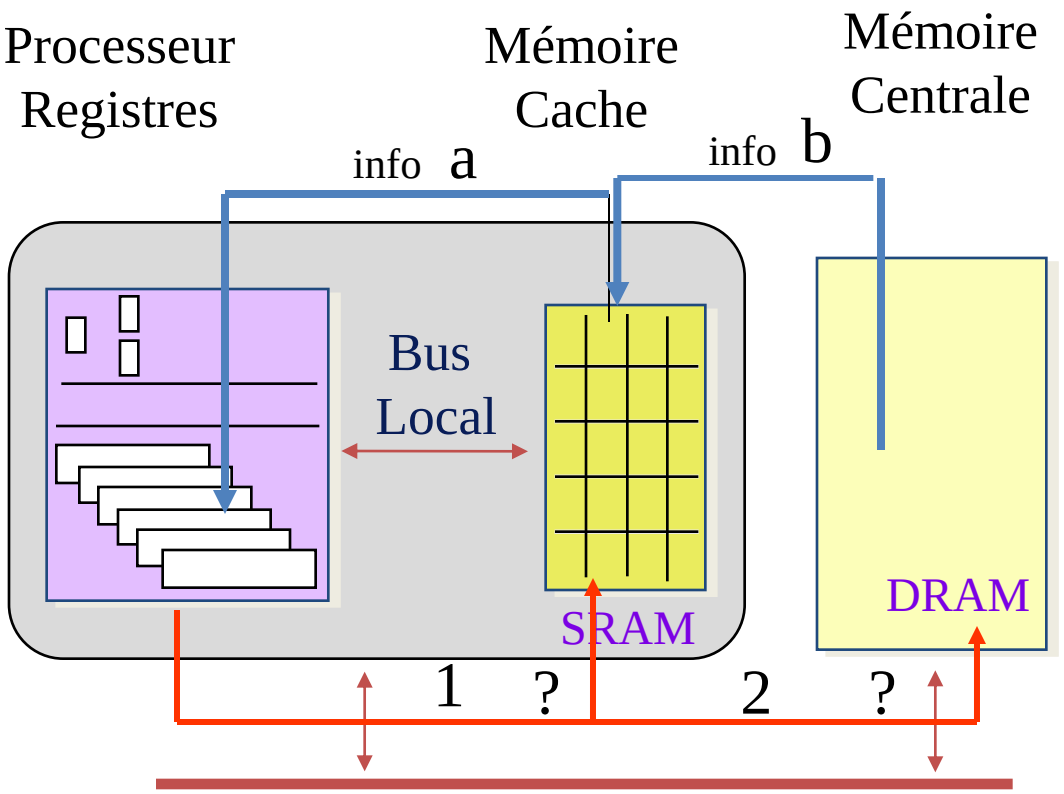
Elle comporte un nombre fini d'entrées. Une entrée contient n mot mémoire et s'appelle une ligne.

**Tableau 2 – Temps d'accès pour un Pentium M**

| Élément | Temps d'accès |
|---|---|
| Registre | ⩽ 1 cycle |
| Cache L1 | ~ 3 cycles |
| Cache L2 | ~ 14 cycles |
| Mémoire principale | ~ 240 cycles |

# Mémoire cache : principe

La stratégie suivie s'appuie sur le principe de localité

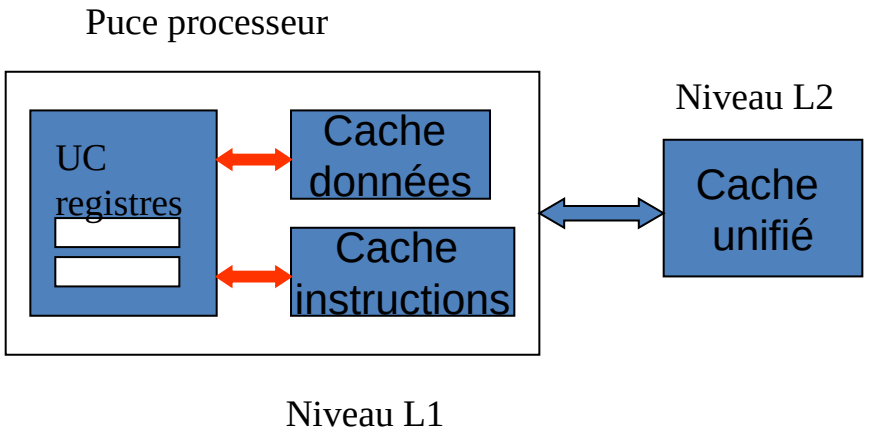



1. L'info cherchée est-elle dans le cache ?
OUI / Succès (a) : ramener l'info dans le processeur
NON / Défaut (2) : chercher l'info dans la mémoire centrale

2. L'info est-elle en mémoire centrale ?
OUI / Succès (b) : ramener l'info dans le cache , puis dans le processeur (a)
NON / Défaut

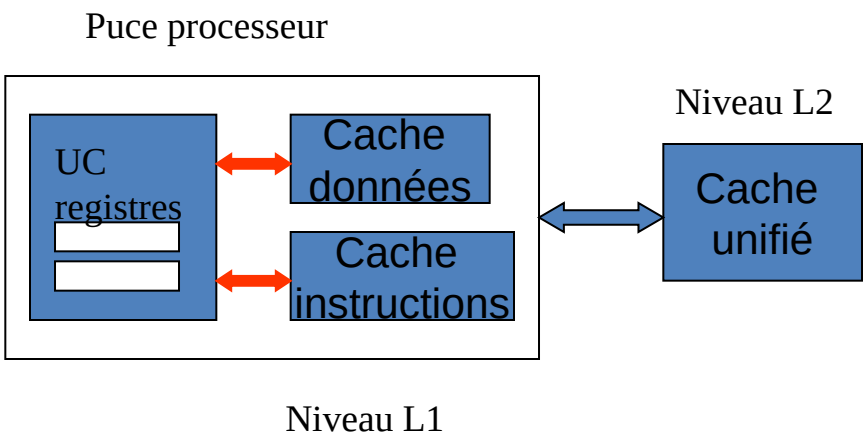
# Le principe de hiérarchie mémoire : les caches
# Les différents types de caches

## Les structures de caches



# Les mémoires de l'ordinateur
# Le principe de hiérarchie mémoire : les caches

## Les structures de caches



# Mémoire cache : structure

- La recherche d'un mot dans le cache s'effectue à partir de son adresse en mémoire centrale.

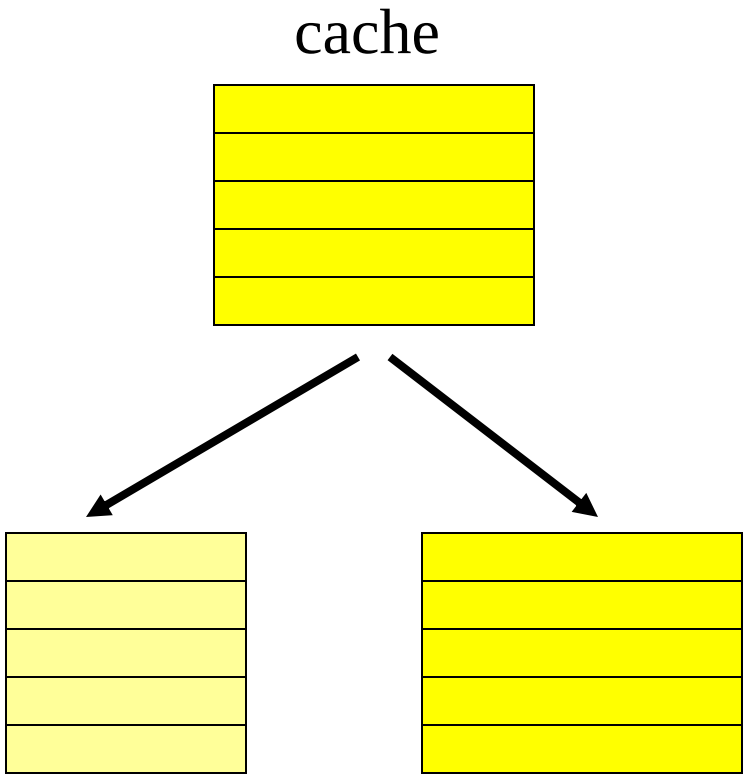
- Un cache est caractérisé :
  - sa capacité

    Nombre d'entrées * taille du bloc de données

    *128 * 16 octets*
  - son organisation
    - Cache associatif
    - Cache direct
    - Cache mixte

# Mémoire cache : structure

- Trois types de cache :
  - associatif : un bloc de mots de la mémoire centrale est placé dans n'importe quelle entrée (ligne) libre du cache
  - à correspondance directe : l'entrée (ligne) du cache occupée par un bloc de mots est fonction de l'adresse en mémoire centrale de ce bloc.
  - mixte

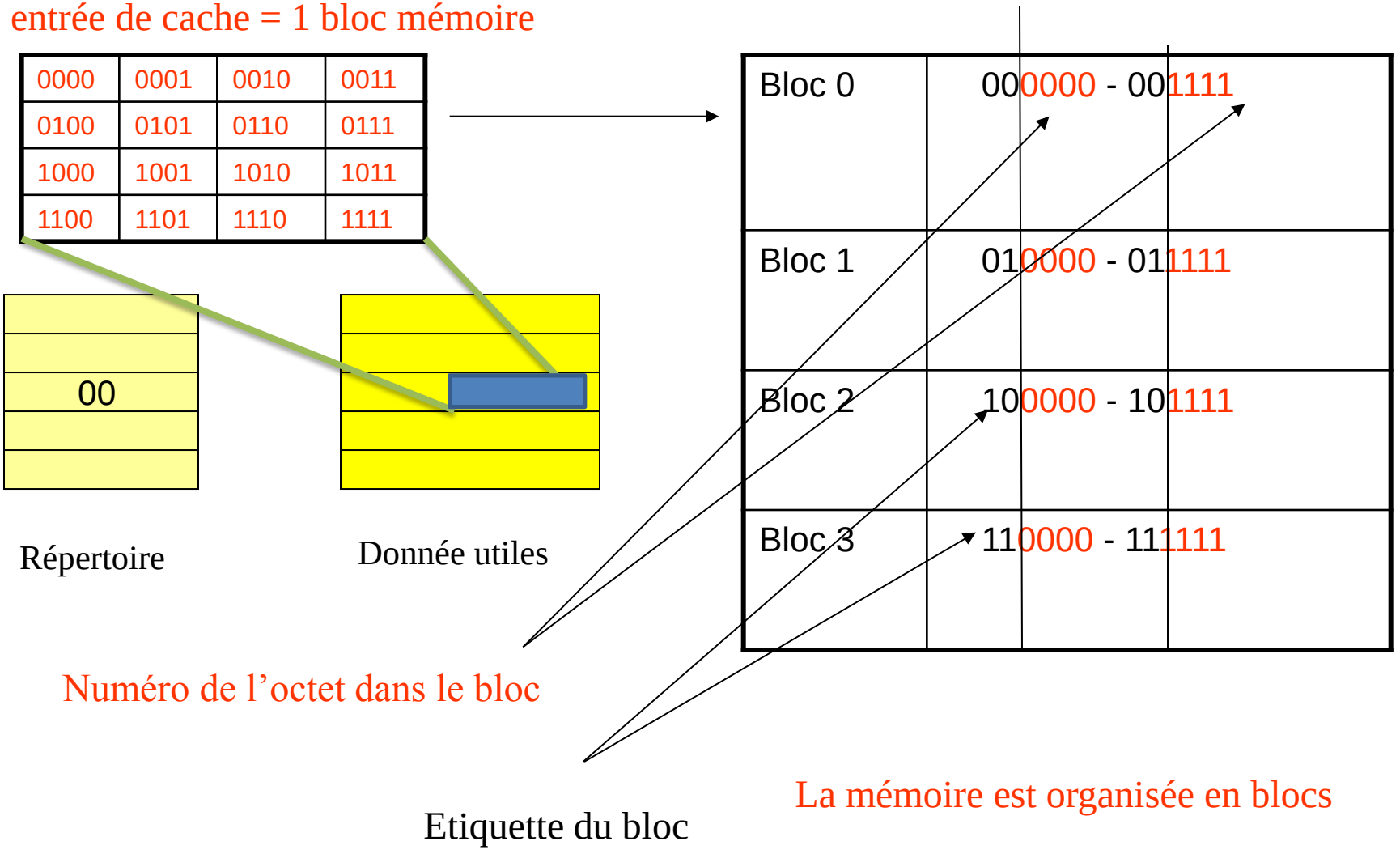



Répertoire :
Contient l'étiquette du bloc présent dans l'entrée du cache

Donnée utiles
Contient le bloc de mots (n octets Puissance de 2)

# Mémoire cache : structure

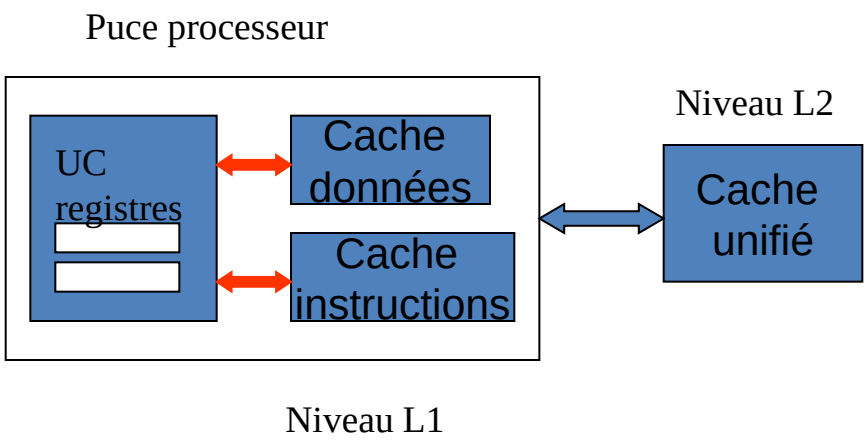
1 entrée de cache = 1 bloc mémoire



# Les mémoires de l'ordinateur
# Le principe de hiérarchie mémoire : les caches

## Les structures de caches

## Cache associatif



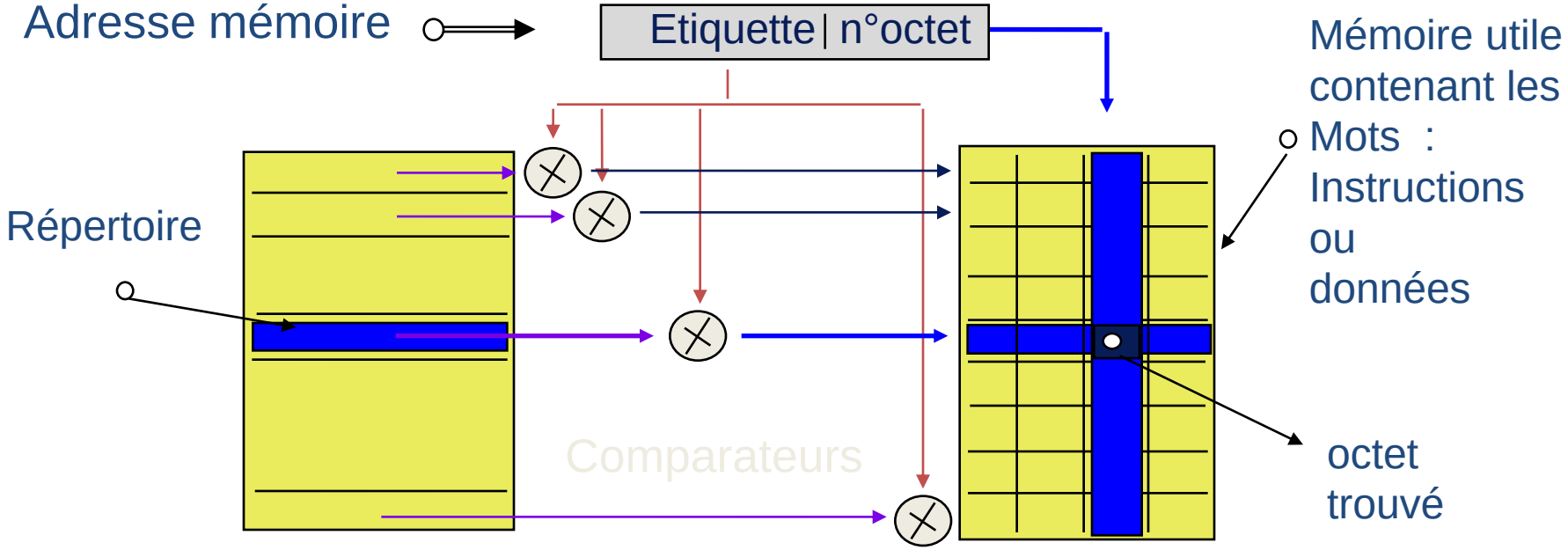
# Cache associatif

- Un bloc de mots de la mémoire centrale est placé dans n'importe quelle entrée libre du cache

- si le cache est plein, il faut libérer une entrée

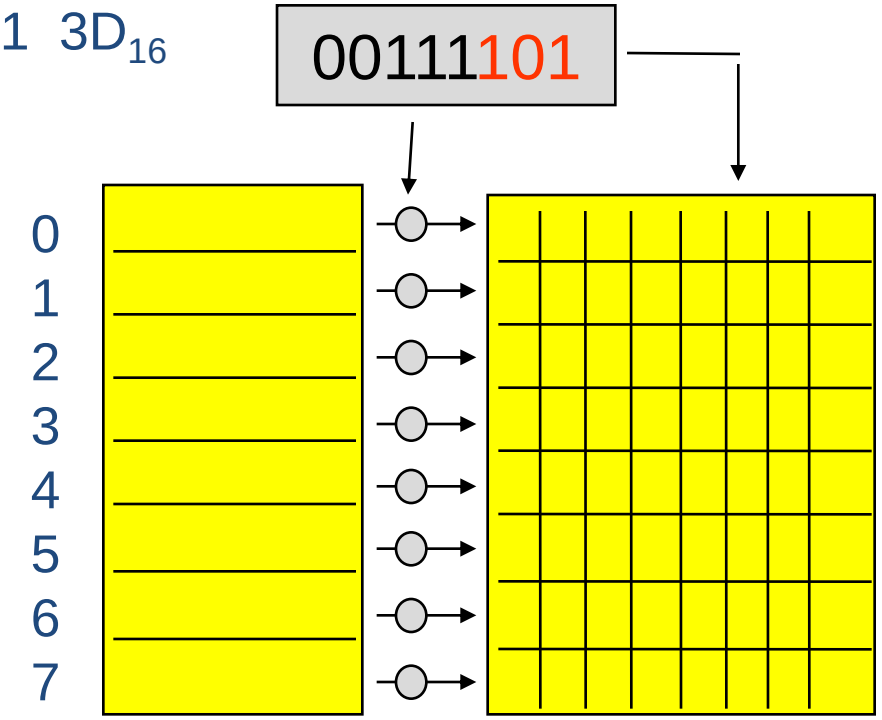
  - Algorithme de remplacement de ligne

# Cache purement associatif (lecture)

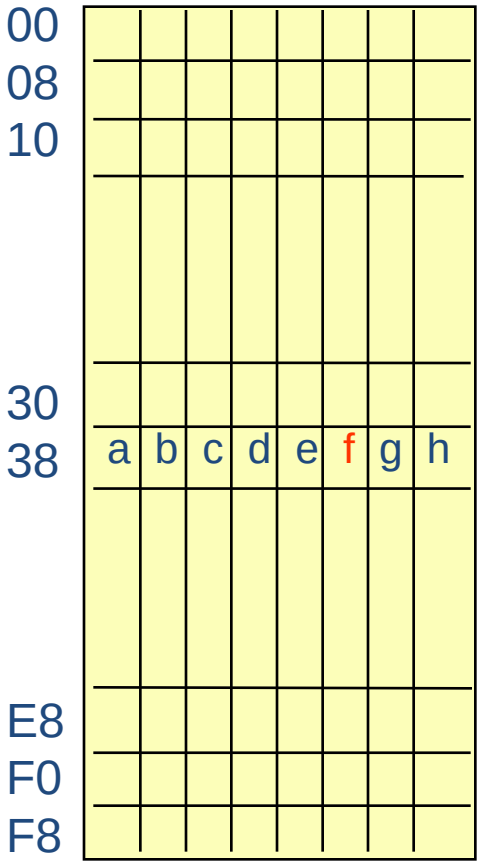



Si Répertoire contient Etiquette Alors bloc de mots trouvé
Charger le processeur avec ligne[n°octet]
Sinon Si Répertoire plein Alors Algorithme de remplacement de ligne
Remplir la ligne choisie
Charger le processeur avec ligne[n°octet]
Sinon Remplir une ligne libre
Charger le processeur avec ligne [n°octet]
FinSi
FinSi

Load D R1 $3D_{16}$




Mémoire centrale

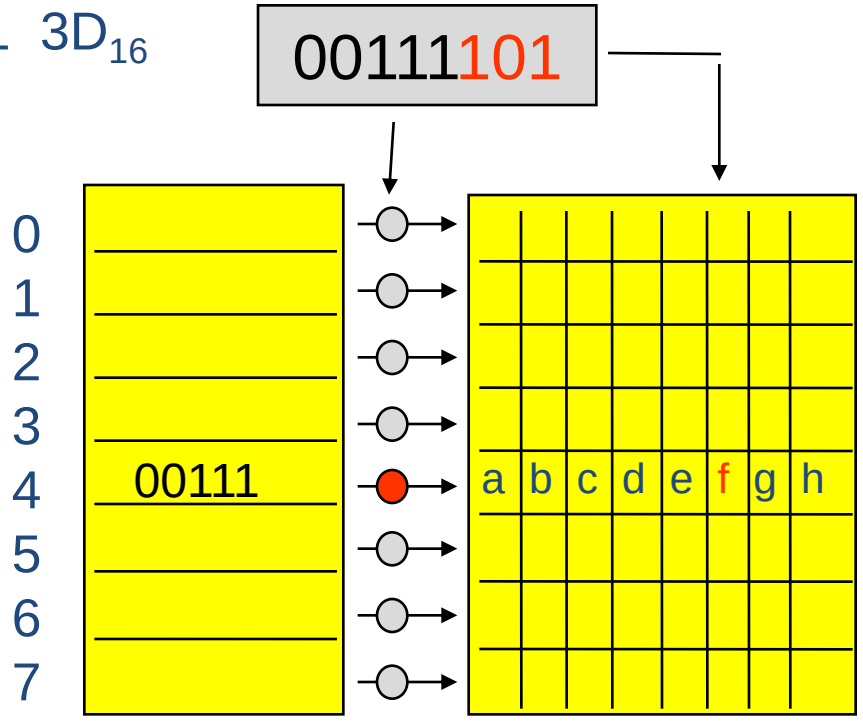



Chaque ligne contient 8 octets → 3 bits de poids faible pour les désigner
L'étiquette est formée des 5 bits de poids fort

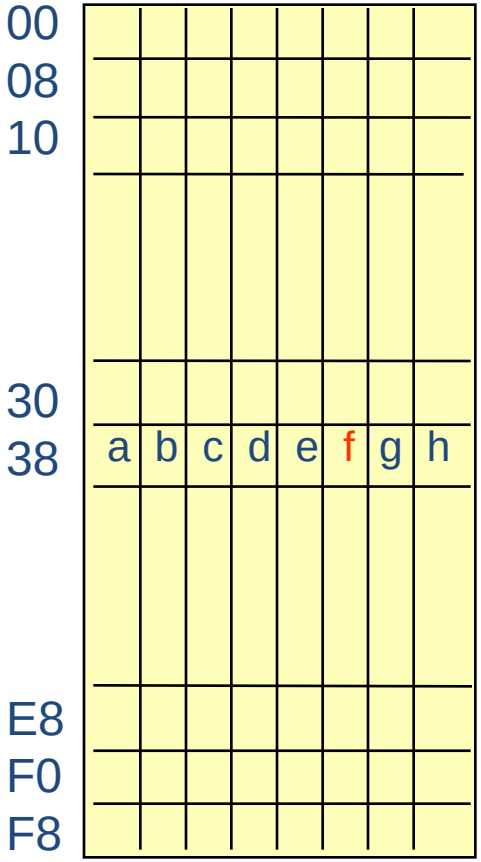
Si répertoire contient 00111 Alors Charger f dans processeur
Sinon Si répertoire plein Alors Remplacement ligne
Remplir ligne
Charger f dans processeur
Sinon Remplir une ligne
Charger f dans processeur

*Cache purement associatif : Lecture*

Load D R1 $3D_{16}$




Mémoire centrale

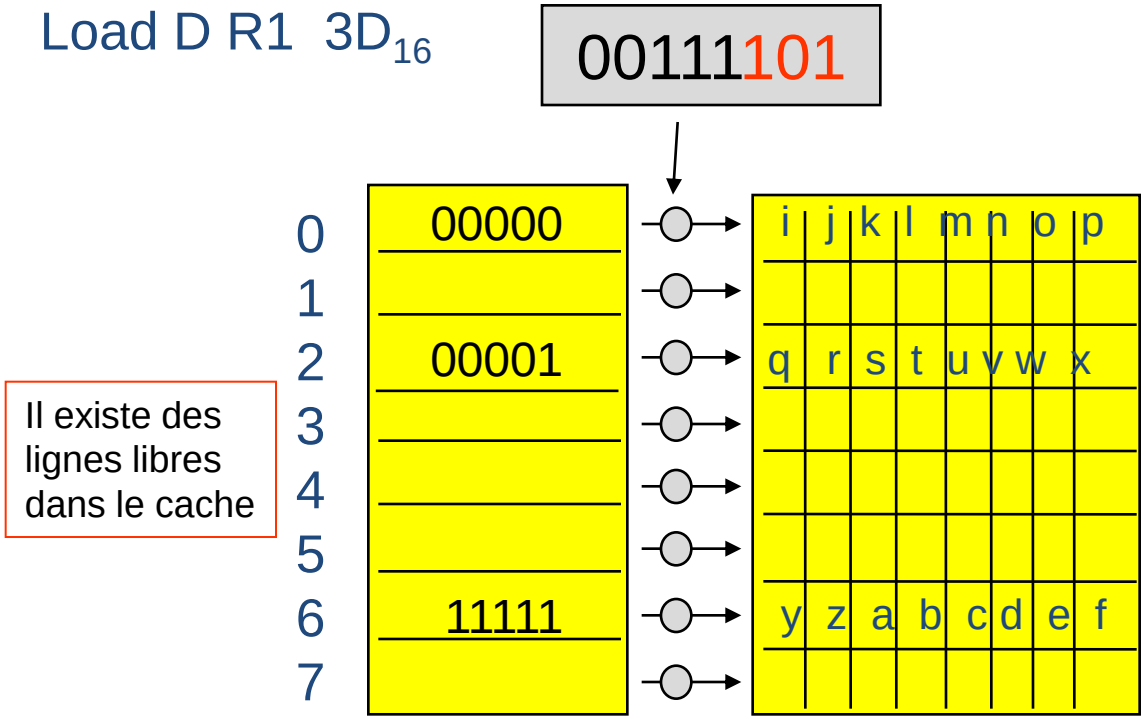



Chaque ligne contient 8 octets → 3 bits de poids faible pour les désigner
L'étiquette est formée des 5 bits de poids fort

Si répertoire contient 00111 Alors Charger f dans processeur
Sinon Si répertoire plein Alors Remplacement ligne
Remplir ligne
Charger f dans processeur
Sinon Remplir une ligne
Charger f dans processeur

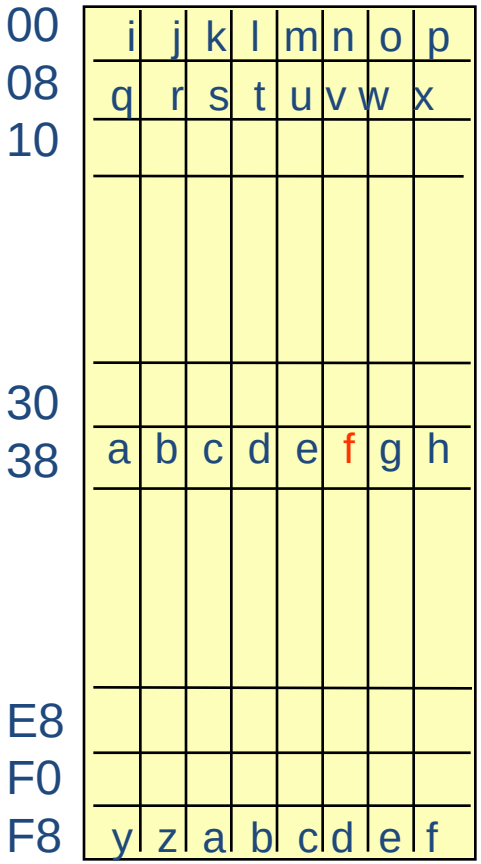
*Cache purement associatif : SUCCES*

Load D R1 $3D_{16}$




Il existe des lignes libres dans le cache

Mémoire centrale

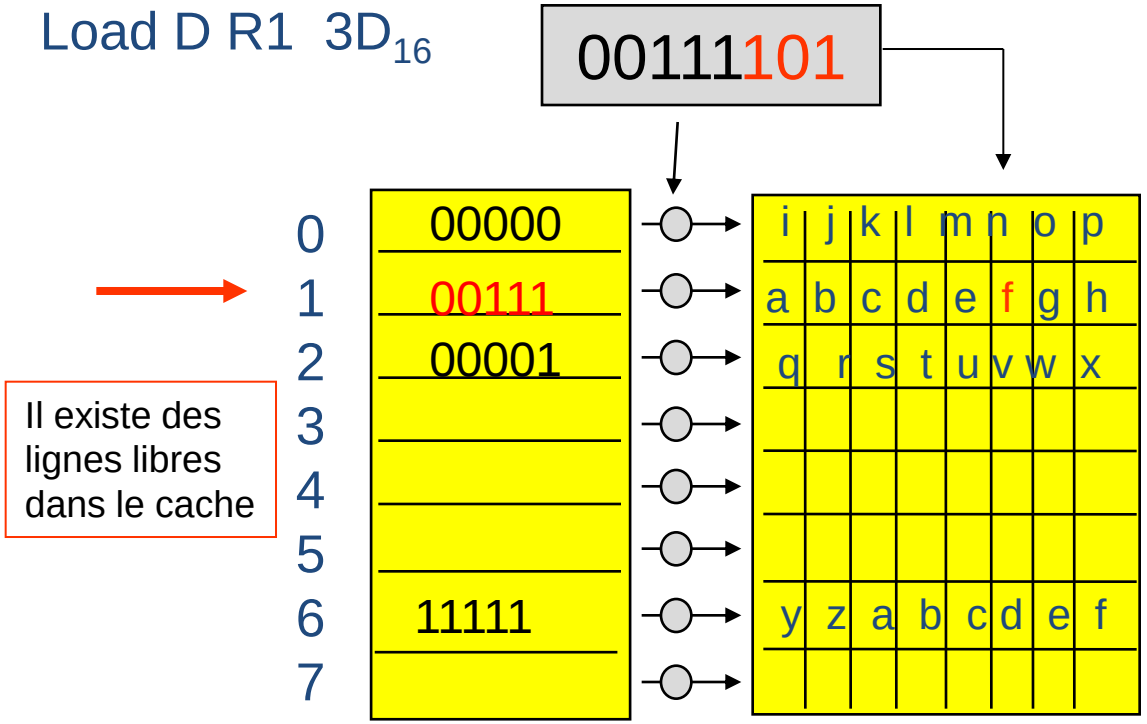



Aucune égalité entre l'étiquette de l'adresse et les étiquettes présentes
Dans le répertoire : défaut

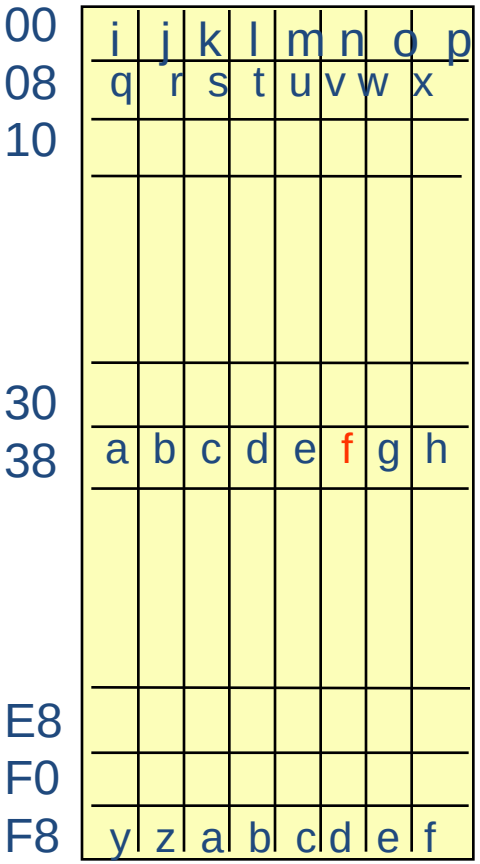
Si répertoire contient 00111 Alors Charger f dans processeur
Sinon Si répertoire plein Alors Remplacement ligne
Remplir ligne
Charger f dans processeur
Sinon Remplir une ligne
Charger f dans processeur

*Cache purement associatif : Défaut et ligne disponible*

Load D R1 $3D_{16}$




Mémoire centrale



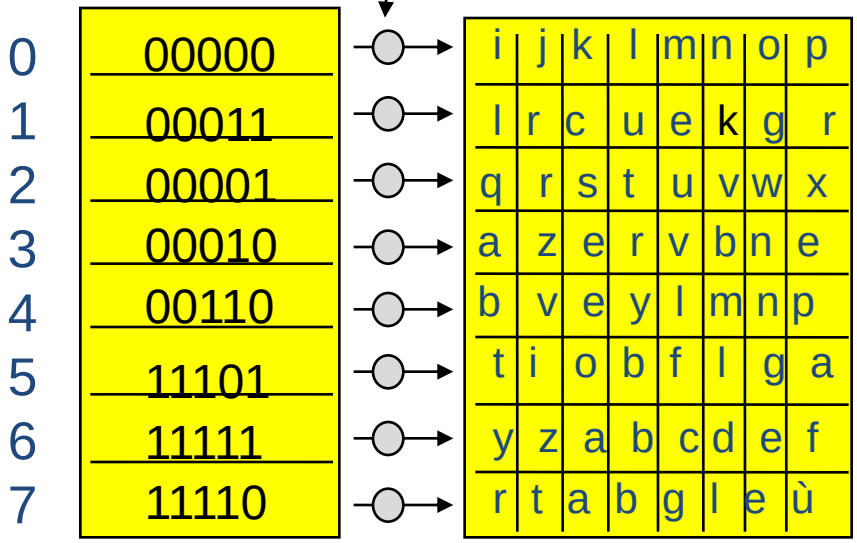

1/ Des lignes sont disponibles; on charge par exemple la ligne 1

Si répertoire contient 00111 Alors Charger f dans processeur
Sinon Si répertoire plein Alors Remplacement ligne
Remplir ligne
Charger f dans processeur
Sinon Remplir une ligne
Charger f dans processeur

*Cache purement associatif : Défaut et ligne disponible*

Load D R1 $3D_{16}$

00111101




Il n'existe pas de lignes libres dans le cache

Mémoire centrale

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

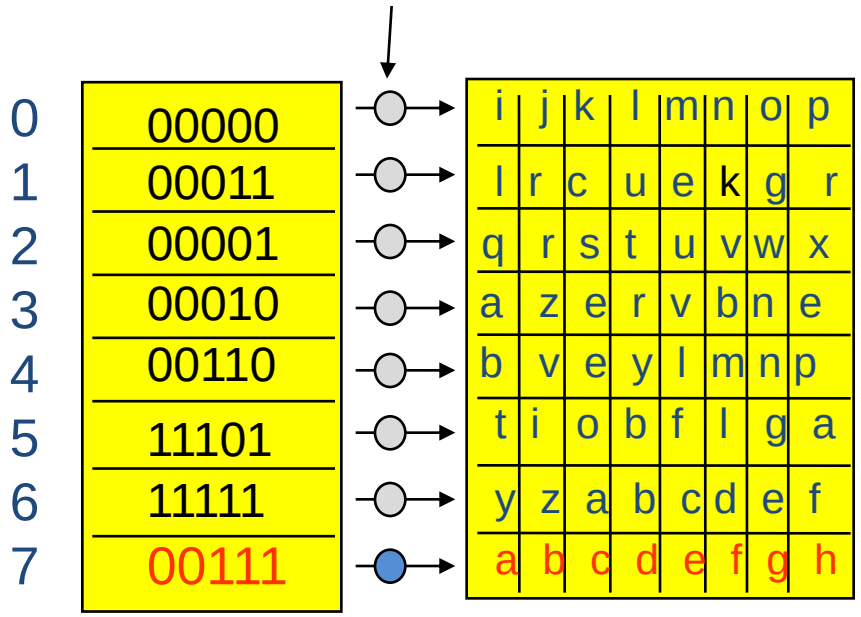
2/ aucune ligne n'est disponible . Il faut en libérer une. On fait appel à un algorithme de remplacement de ligne

Si répertoire contient 00111 Alors Charger f dans processeur
Sinon Si répertoire plein Alors Remplacement ligne
Remplir ligne
Charger f dans processeur
Sinon Remplir une ligne
Charger f dans processeur

*Cache purement associatif : Défaut et aucune ligne disponible*

Load D R1 $3D_{16}$

00111101




Il n'existe pas de lignes libres dans le cache
On choisit une ligne à remplacer

Mémoire centrale

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

On fait tourner cet algorithme (FIFO, LRU) : le ligne 7 est victime. on Remplace son contenu

Si répertoire contient 00111 Alors Charger f dans processeur
Sinon Si répertoire plein Alors Remplacement ligne
Remplir ligne
Charger f dans processeur
Sinon Remplir une ligne
Charger f dans processeur

*Cache purement associatif : Défaut et aucune ligne disponible*

# Cache associatif

– Un bloc de mots de la mémoire centrale est placé dans n'importe quelle entrée libre du cache

  - si le cache est plein, il faut libérer une entrée

  Algorithme de remplacement de ligne

  - Aléatoire : une ligne au hasard
  - FIFO : *First In First Out* : la ligne remplacée est la plus ancienne dans le cache
  - LRU : *Least recently Used* : la ligne remplacée est la moins récemment accédée
  - NMRU : *Not most recently Used* : la ligne remplacée n'est pas la plus récemment utilisée

# Cache associatif

## Algorithme de remplacement de ligne

FIFO : *First In First Out* : la ligne remplacée est la plus ancienne dans le cache.

Simple mais pas forcément pertinent.

| Accès (étiquette bloc) | 00000 | 00001 | 00010 | 00100 | 00000 | 10000 | 00010 | 11000 |
|---|---|---|---|---|---|---|---|---|
| Ligne 0 | 00000 | 00000 | 00000 | 00000 | 00000 | 10000 | 10000 | 10000 |
| Ligne 1 | | 00001 | 00001 | 00001 | 00001 | 00001 | 00001 | 11000 |
| Ligne 2 | | | 00010 | 00010 | 00010 | 00010 | 00010 | 00010 |
| Ligne 3 | | | | 00100 | 00100 | 00100 | 00100 | 00100 |
| | D | D | D | D | S | D | S | D |

# Cache associatif

## Algorithme de remplacement de ligne

LRU : *Least recently Used* : la ligne remplacée est la moins récemment accédée

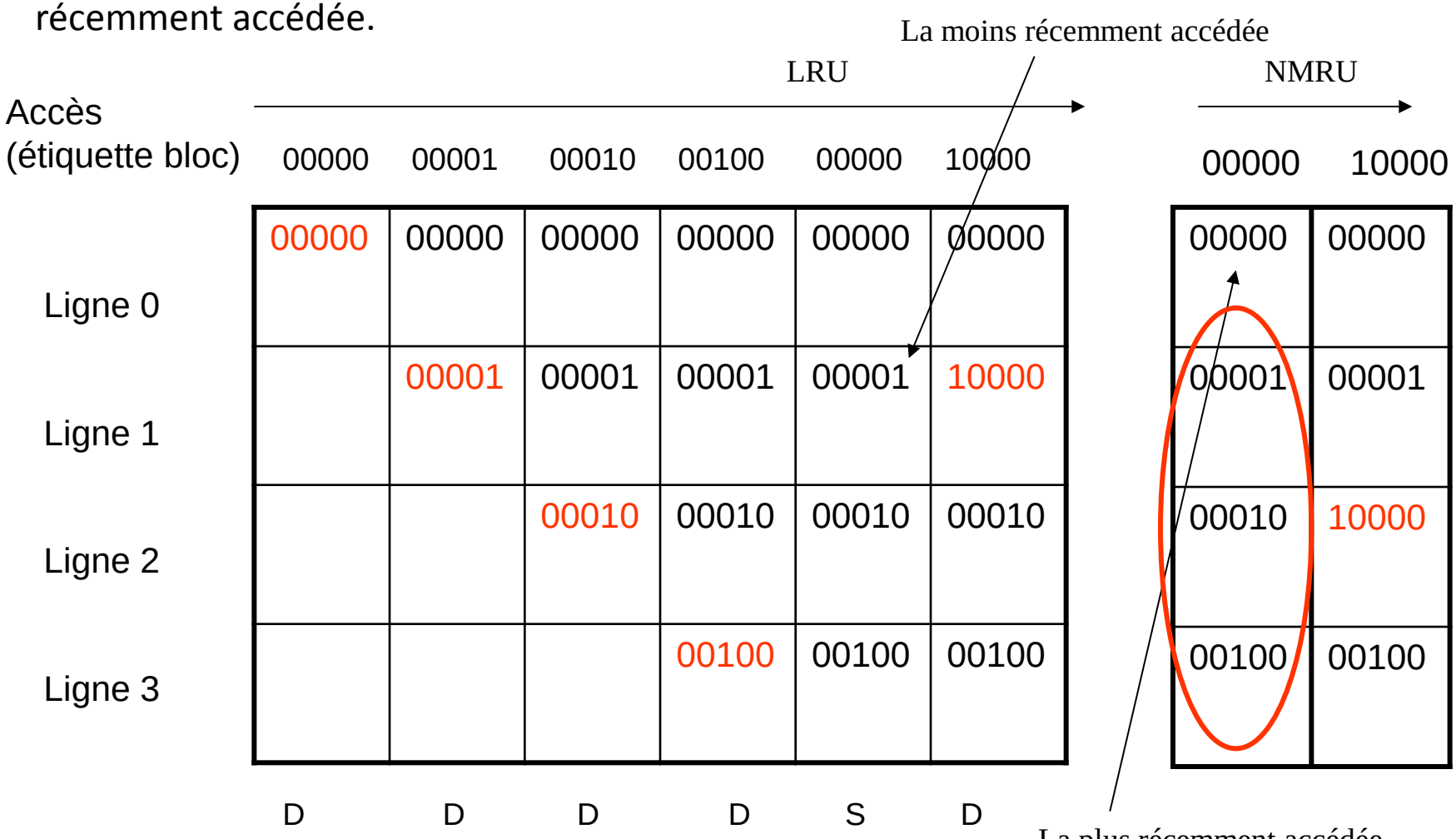
Complexe à mettre en œuvre car nécessite de maintenir l'ordre des accès.

| Accès (étiquette bloc) | 00000 | 00001 | 00010 | 00100 | 00000 | 10000 | 00010 | 11000 |
|---|---|---|---|---|---|---|---|---|
| Ligne 0 | 00000 | 00000 | 00000 | 00000 | 00000 | 00000 | 00000 | 00000 |
| Ligne 1 | | 00001 | 00001 | 00001 | 00001 | 10000 | 10000 | 10000 |
| Ligne 2 | | | 00010 | 00010 | 00010 | 00010 | 00010 | 00010 |
| Ligne 3 | | | | 00100 | 00100 | 00100 | 00100 | 11000 |
| | D | D | D | D | S | D | S | D |

# Cache associatif

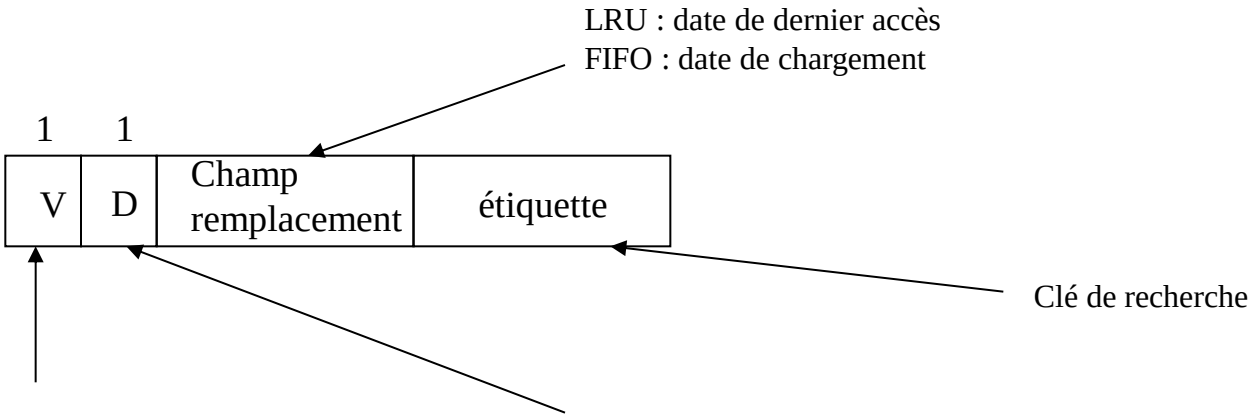
## Algorithme de remplacement de ligne

NMRU : *Not most recently Used* : la ligne remplacée n'est pas la plus récemment utilisée. La ligne remplacée est choisie aléatoirement parmi celles autres que la ligne la plus récemment accédée.



# Cache associatif

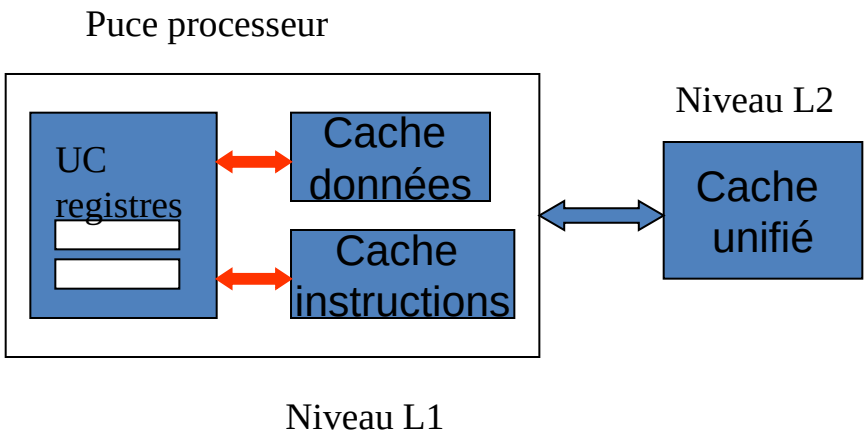
- Coûteux et « encombrants » : 1 comparateur par ligne.
- Complexe : politique de remplacement de ligne.
- Format d'une entrée de cache (répertoire)



# Les mémoires de l'ordinateur
# Le principe de hiérarchie mémoire : les caches

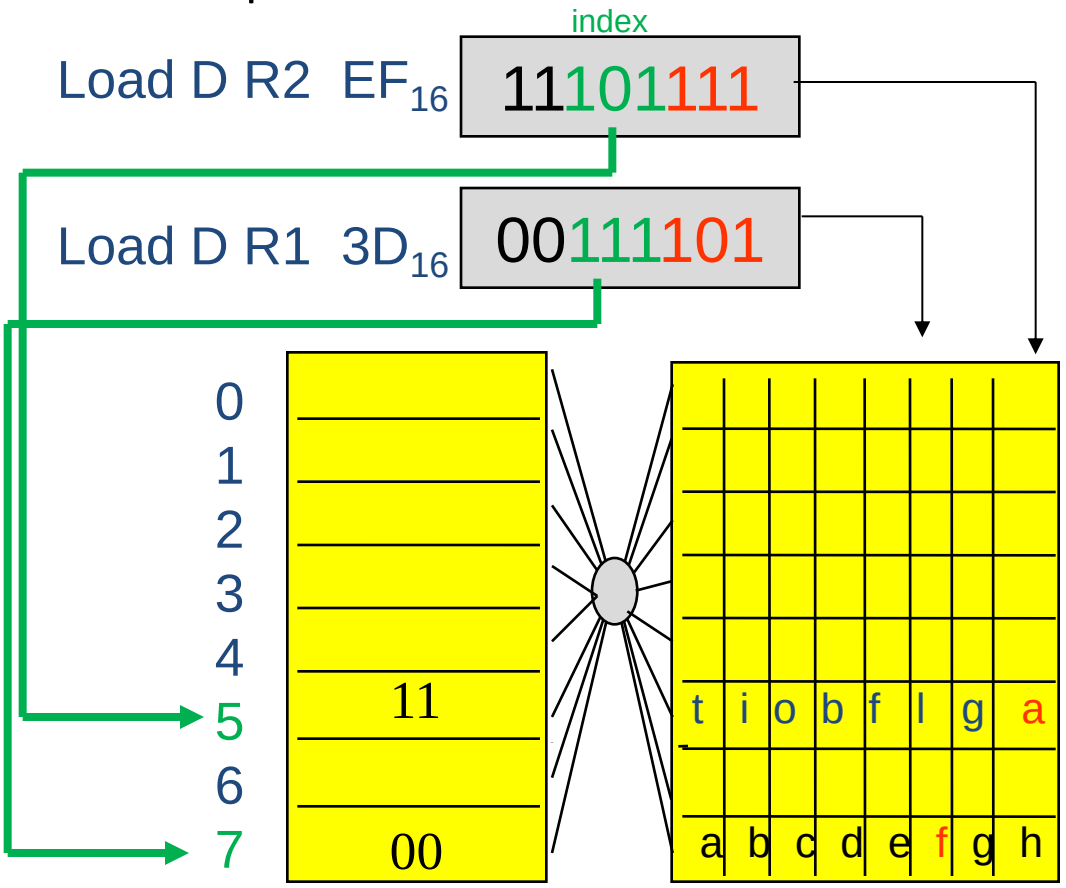
## Les structures de caches

## Cache à correspondance directe



# Cache à correspondance directe

– Un bloc de mots de la mémoire centrale est placé dans une entrée du cache qui est fonction de son adresse en mémoire centrale.

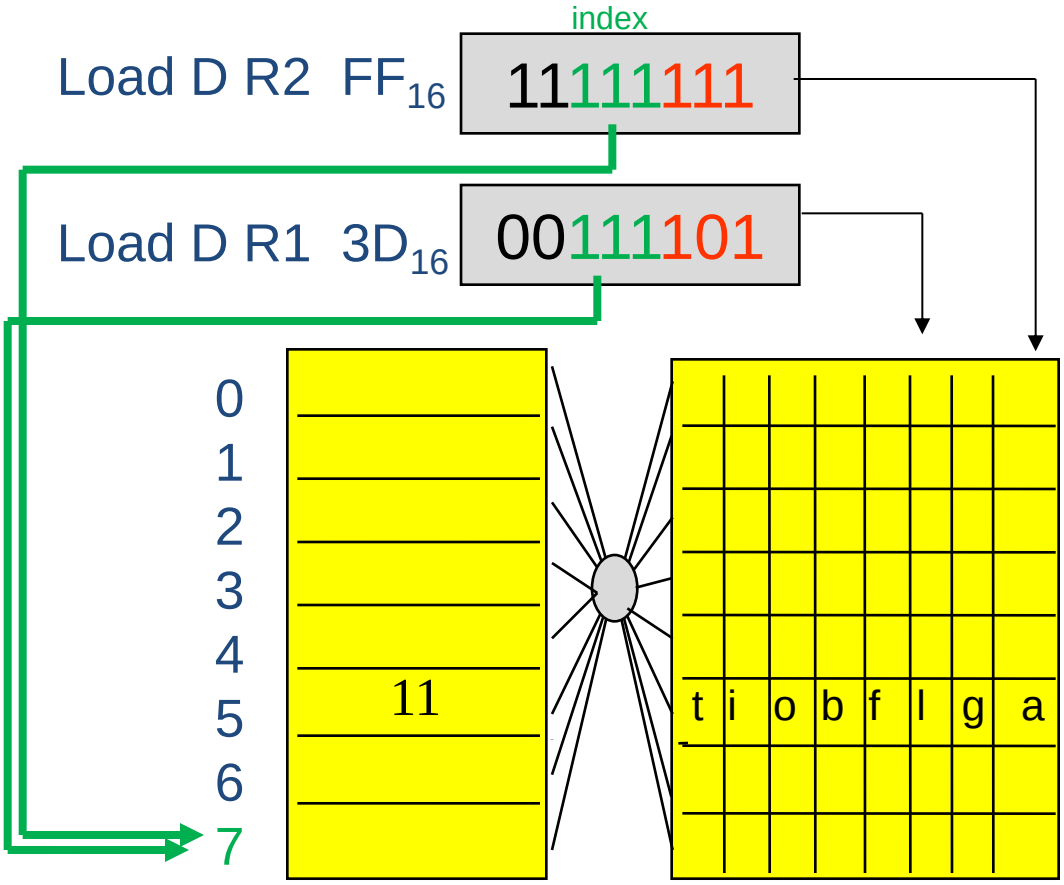



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

Chaque bloc contient 8 octets → 3 bits de poids faible pour les désigner
Le cache contient 8 entrées → 3 bits pour les désigner
L'étiquette est formée des 2 bits de poids fort restant

# Cache à correspondance directe

– Un bloc de mots de la mémoire centrale est placé dans une entrée du cache qui est fonction de son adresse en mémoire centrale.

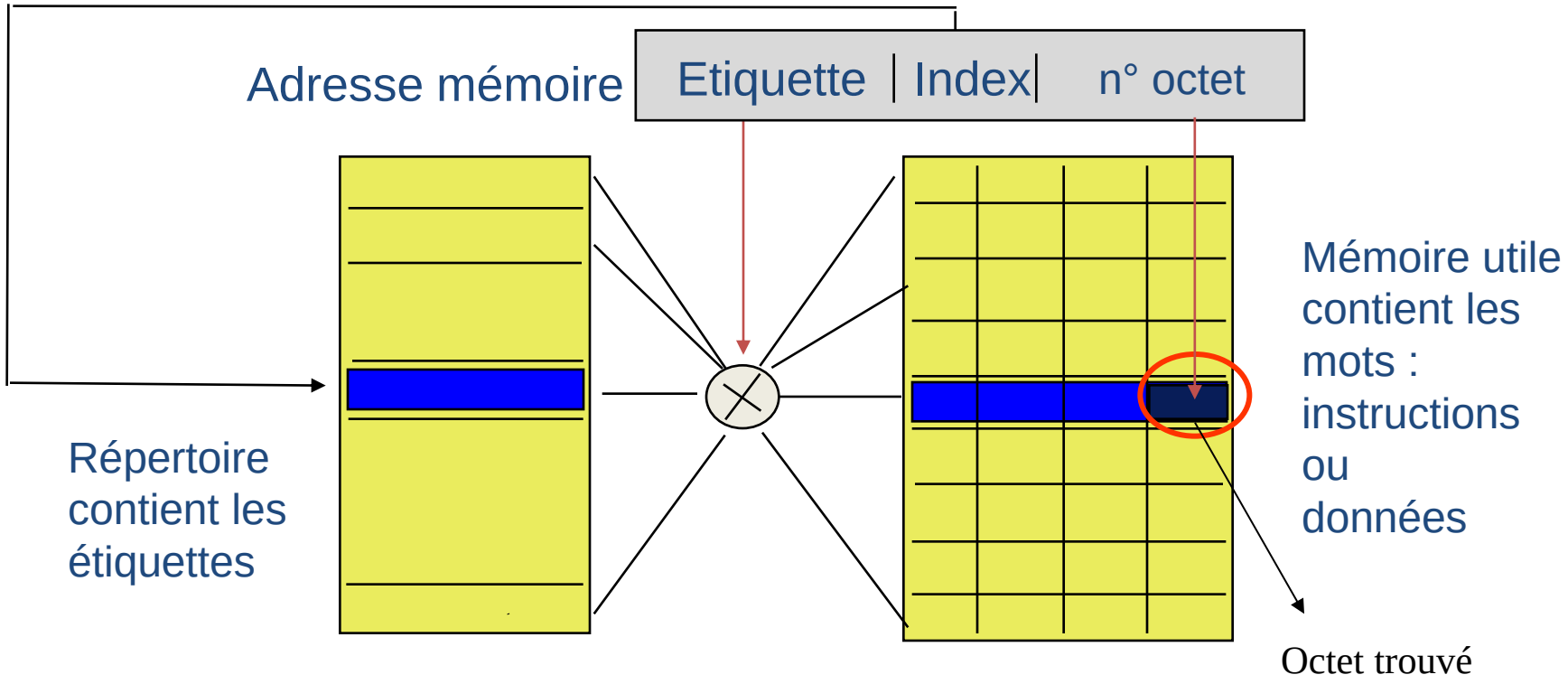



Deux (n) blocs de la mémoire centrale entrent dans la même entrée du cache (tous ceux ayant la même valeur d'index)

La valeur d'étiquette stockée dans la ligne permet de connaître quel bloc occupe la ligne à un instant donné

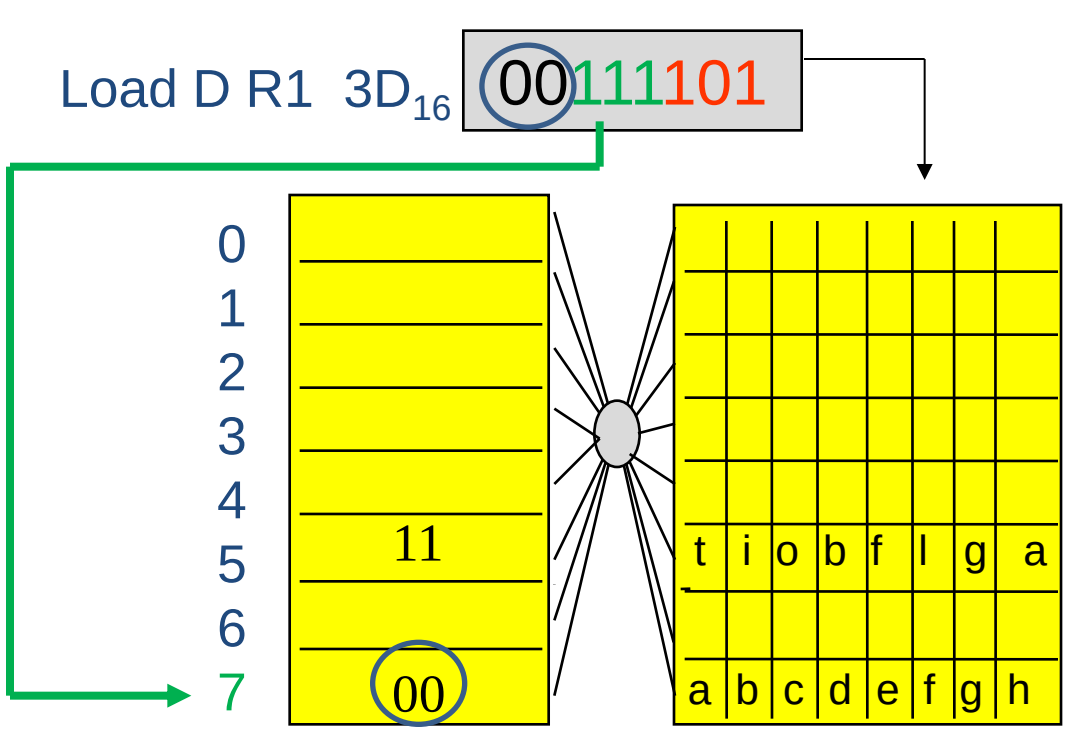
Chaque bloc contient 8 octets → 3 bits de poids faible pour les désigner
Le cache contient 8 entrées → 3 bits pour les désigner
L'étiquette est formée des 2 bits de poids fort restant

# Cache à correspondance directe




Si Répertoire [Index] = Etiquette Alors Bloc de mots trouvé
Charger processeur avec MemoireUtile[Index,n°octet]
Sinon Répertoire[Index] = Etiquette
Charger Ligne[Index] à partir de la mémoire centrale
Charger processeur avec MémoireUtile[Index,n°octet]
FinSi

# Cache à correspondance directe : succès




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

La ligne est occupée, les étiquettes sont égales : succès

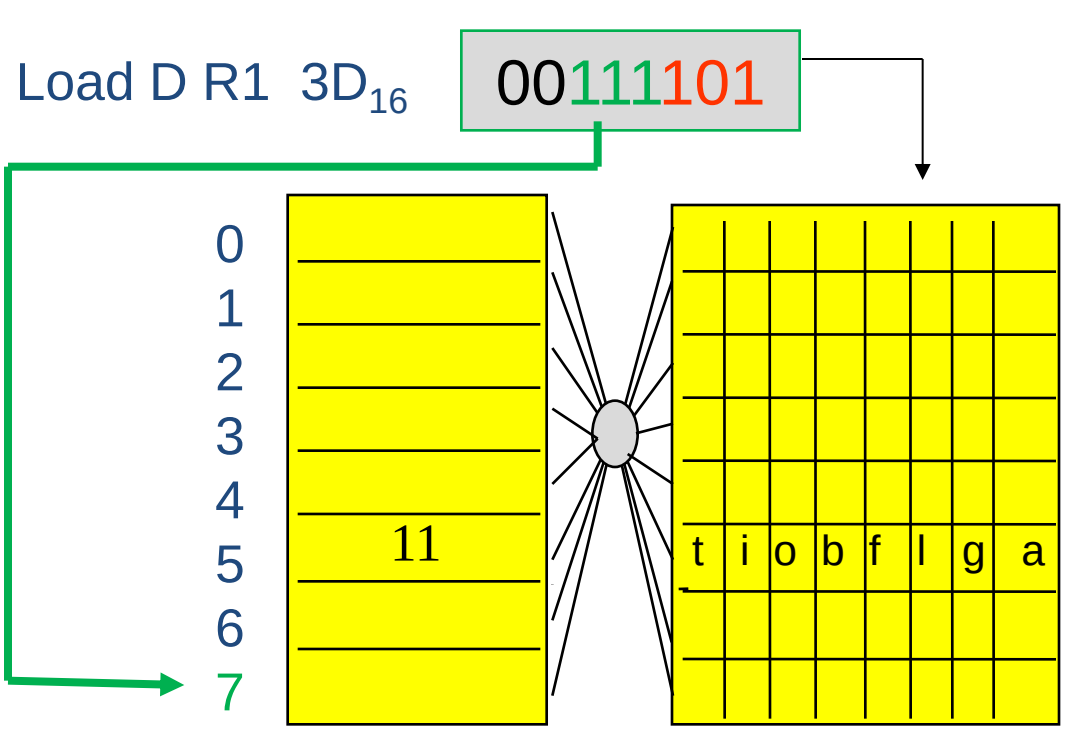
Si Répertoire [111] = 00

Alors Bloc de mots trouvé
Charger processeur avec MemoireUtile[111,101] (f)

Sinon Répertoire[Index] = Etiquette
Charger Ligne[Index] à partir de la mémoire centrale
Charger processeur avec MémoireUtile[Index,n°octet]

FinSi

# Cache à correspondance directe : défaut




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| | | | | | | | | |
| | | | | | | | | |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| | | | | | | | | |
| | | | | | | | | |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

Si Répertoire [111] = 00

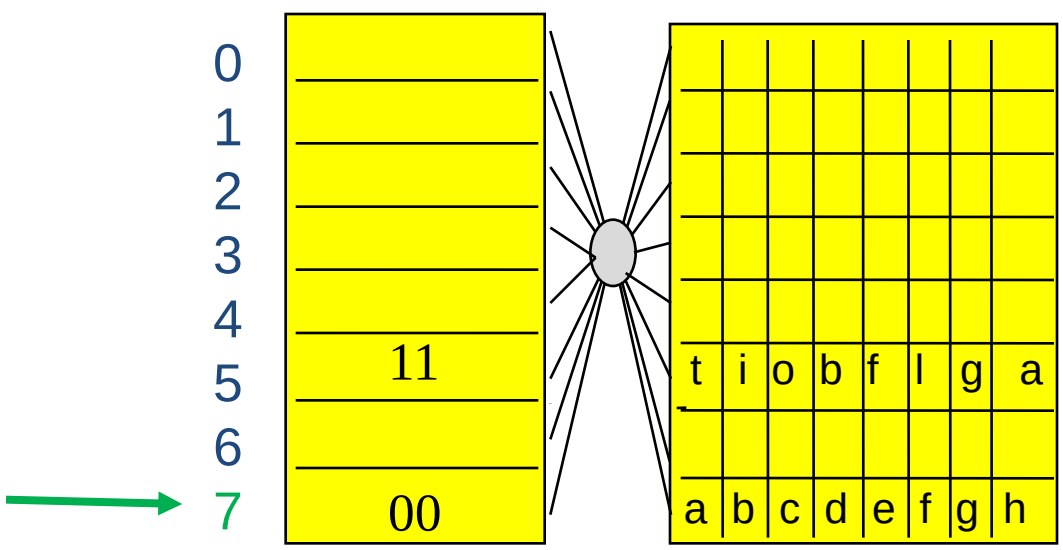
Alors Bloc de mots trouvé
Charger processeur avec MemoireUtile[111,101] (f)

Sinon Répertoire[Index] = Etiquette
Charger Ligne[Index] à partir de la mémoire centrale
Charger processeur avec MémoireUtile[Index,n°octet]

FinSi

# Cache à correspondance directe : défaut

Load D R1 $3D_{16}$ 00111101




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

La ligne est vide : défaut; on remplit la ligne.

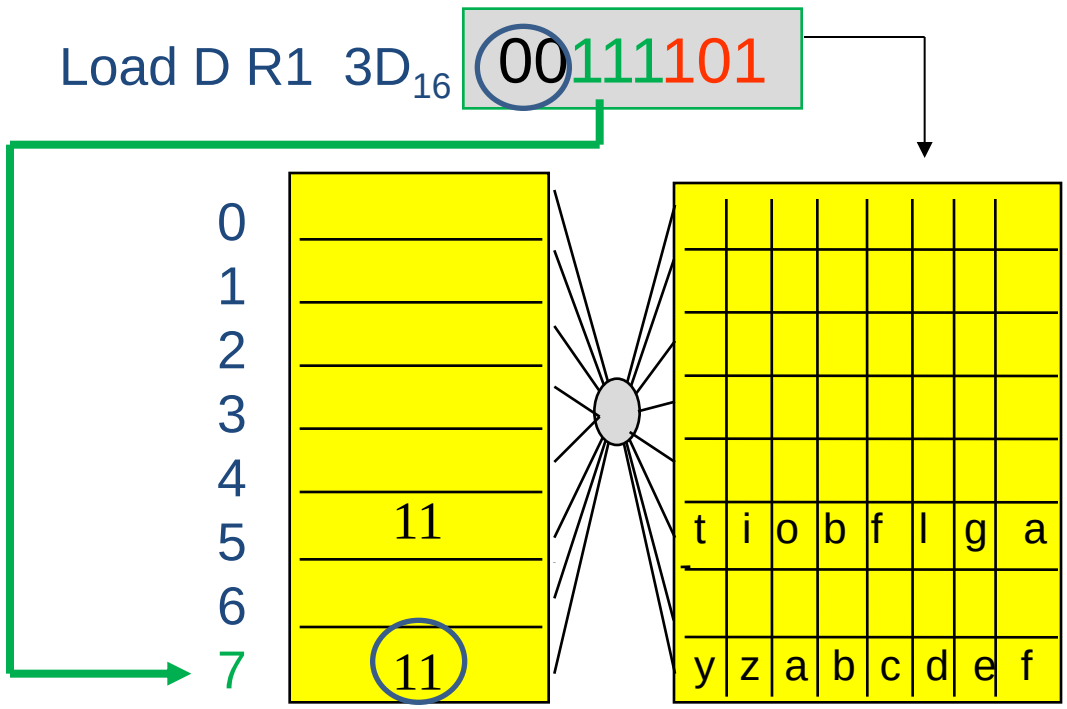
Si Répertoire [111] = 00

Alors Bloc de mots trouvé
Charger processeur avec MemoireUtile[111,101] (f)

Sinon Répertoire[Index] = Etiquette
Charger Ligne[Index] à partir de la mémoire centrale
Charger processeur avec MémoireUtile[Index,n°octet]

FinSi

# Cache à correspondance directe : défaut par collision




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

La ligne est occupée, mais ce n'est pas le bloc cherché car
Les étiquettes sont différentes : Défaut Par collision

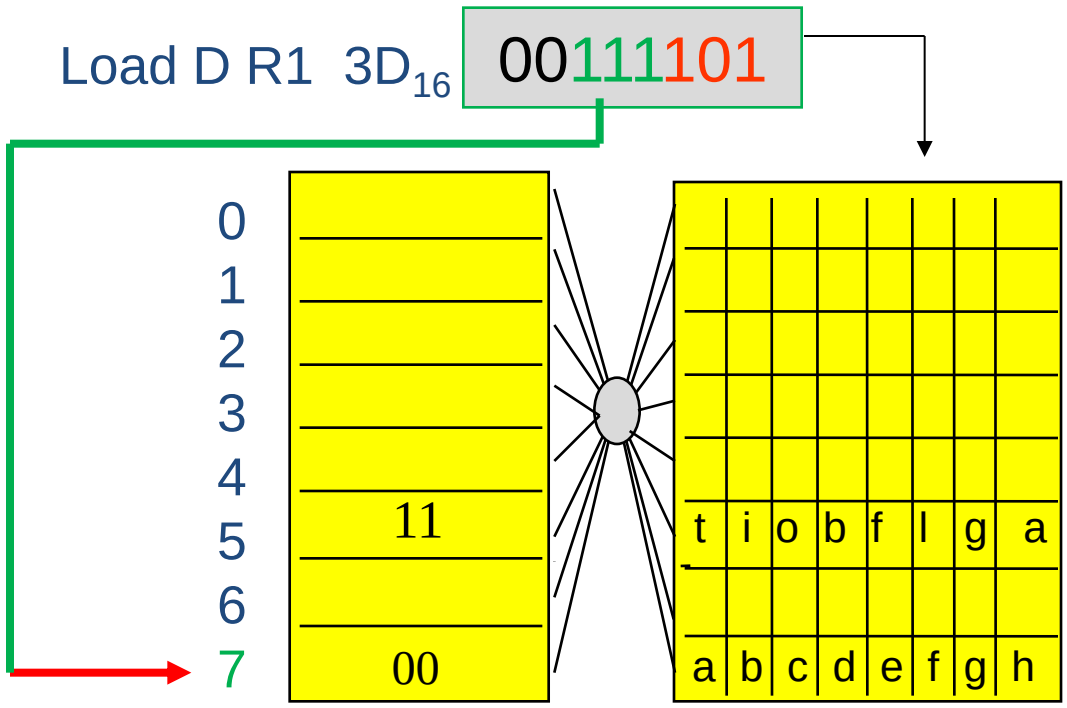
Si Répertoire [111] = 00

Alors Bloc de mots trouvé
Charger processeur avec MemoireUtile[111,101] (f)

Sinon Répertoire[Index] = Etiquette
Charger Ligne[Index] à partir de la mémoire centrale
Charger processeur avec MémoireUtile[Index,n°octet]

FinSi

# Cache à correspondance directe : défaut par collision




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 00 | i | j | k | l | m | n | o | p |
| 08 | q | r | s | t | u | v | w | x |
| 10 | a | z | e | r | v | b | n | e |
| 18 | l | r | c | u | e | k | g | r |
| 30 | b | v | e | y | l | m | n | p |
| 38 | a | b | c | d | e | f | g | h |
| E8 | t | i | o | b | f | l | g | a |
| F0 | r | t | a | b | g | l | e | ù |
| F8 | y | z | a | b | c | d | e | f |

On charge la ligne avec le bloc référencé

Si Répertoire [111] = 00

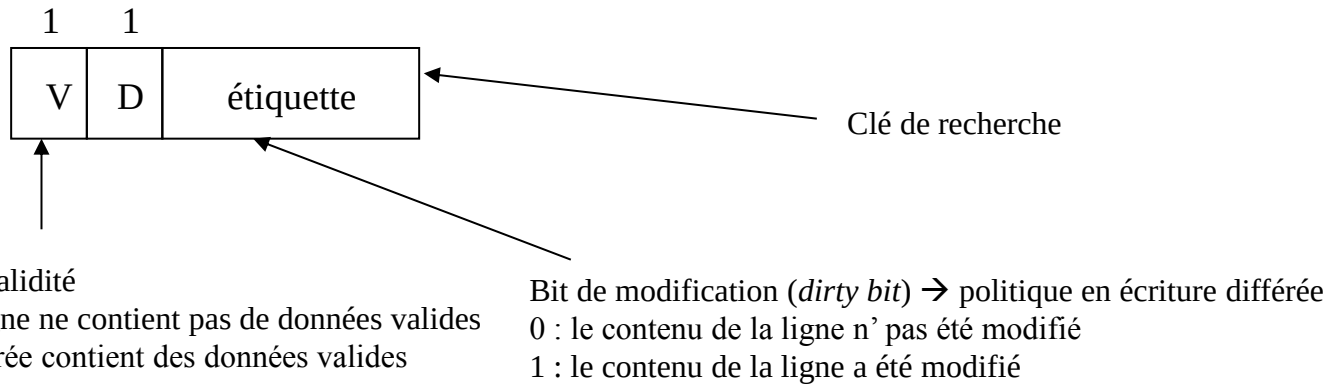
Alors Bloc de mots trouvé
Charger processeur avec MemoireUtile[111,101] (f)

Sinon Répertoire[Index] = Etiquette
Charger Ligne[Index] à partir de la mémoire centrale
Charger processeur avec MémoireUtile[Index,n°octet]

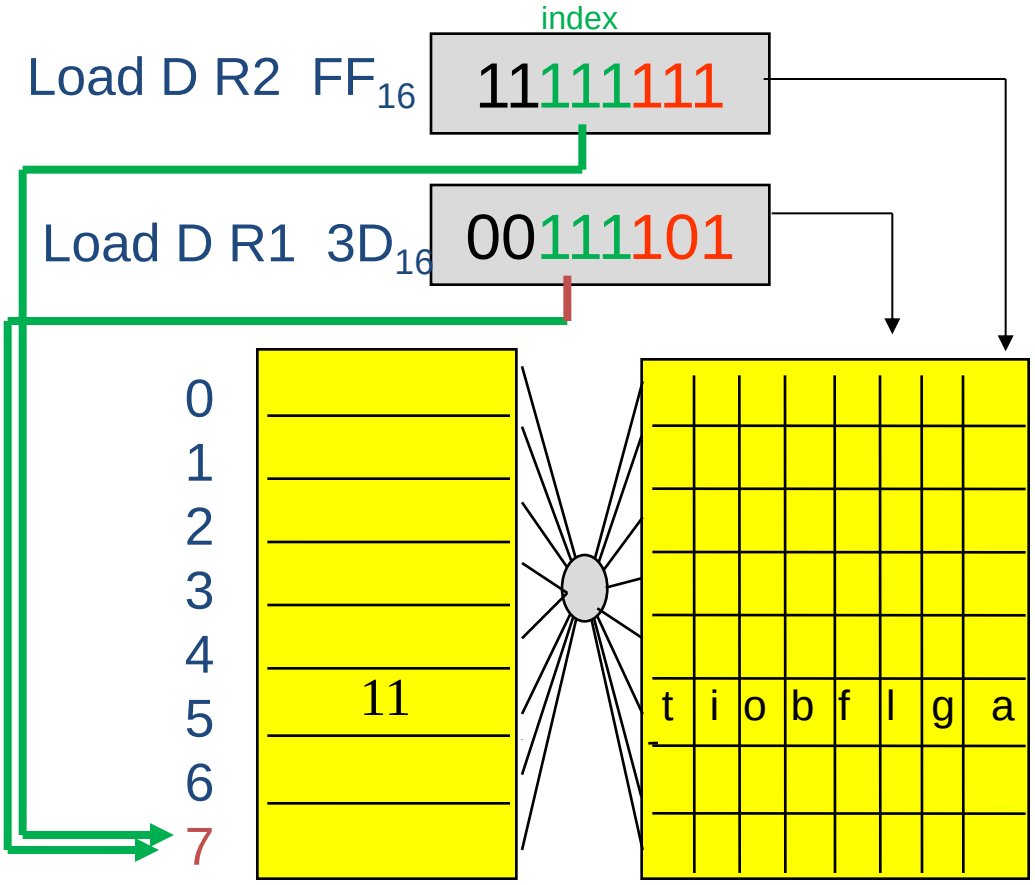
FinSi

# Cache à correspondance directe

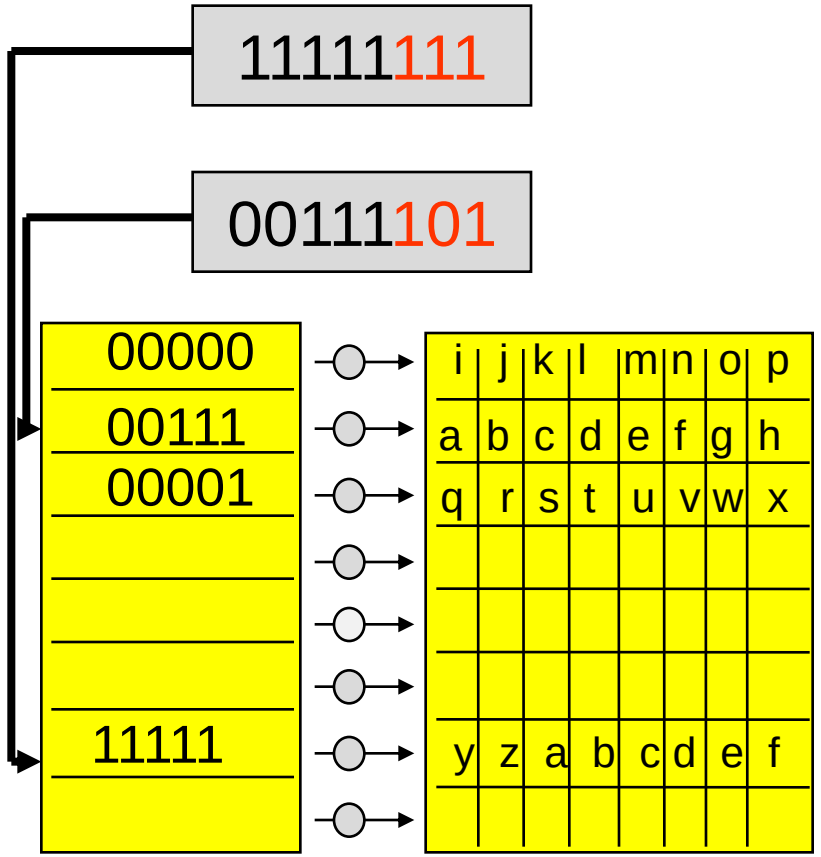
- Moins coûteux et « encombrants » : 1 comparateur par cache
- Complexité : pas de politique de remplacement de ligne.
- Performance moindre due aux conflits de ligne : plusieurs blocs de mots (ceux de même index) se partagent une même entrée.
- Format d'une entrée de cache (répertoire)



# Cache à correspondance directe (échecs par collision)




Cache à correspondance directe : les deux blocs de mots occupent la même ligne.
Des référencements successifs à ces deux blocs causent des défauts même si la cache comporte encore des entrées libres

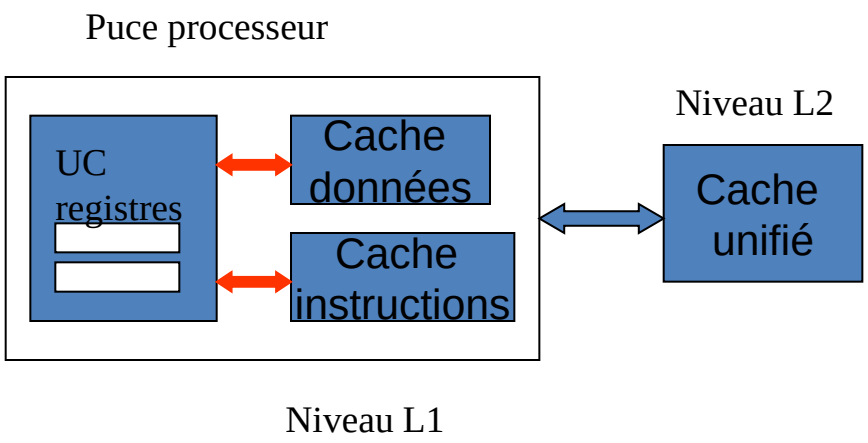



Cache associatif : les deux blocs de mots occupent des lignes différentes

# Les mémoires de l'ordinateur
# Le principe de hiérarchie mémoire : les caches

## Les structures de caches

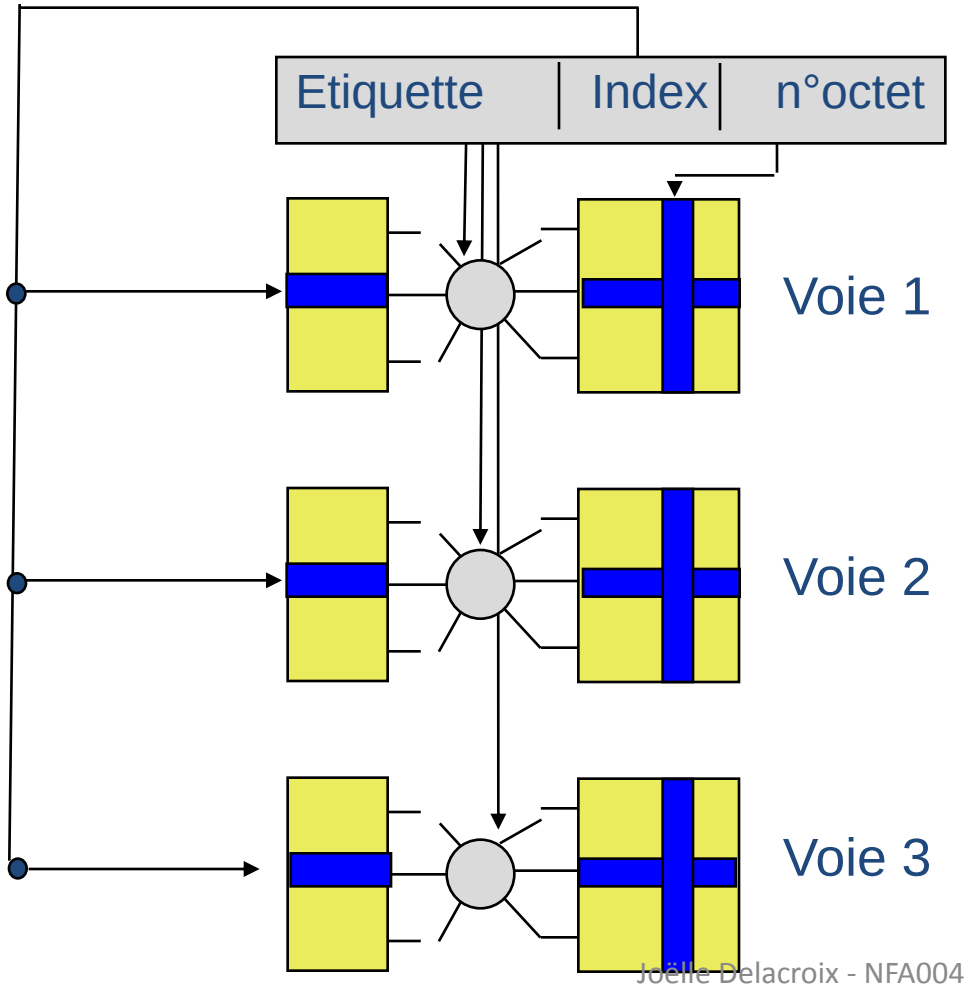
## Cache mixte



# Cache mixte ou associatif par blocs

– Solution intermédiaire entre le cache purement associatif et le cache à correspondance directe

- Le cache est divisé en sous-ensembles appelés voies.

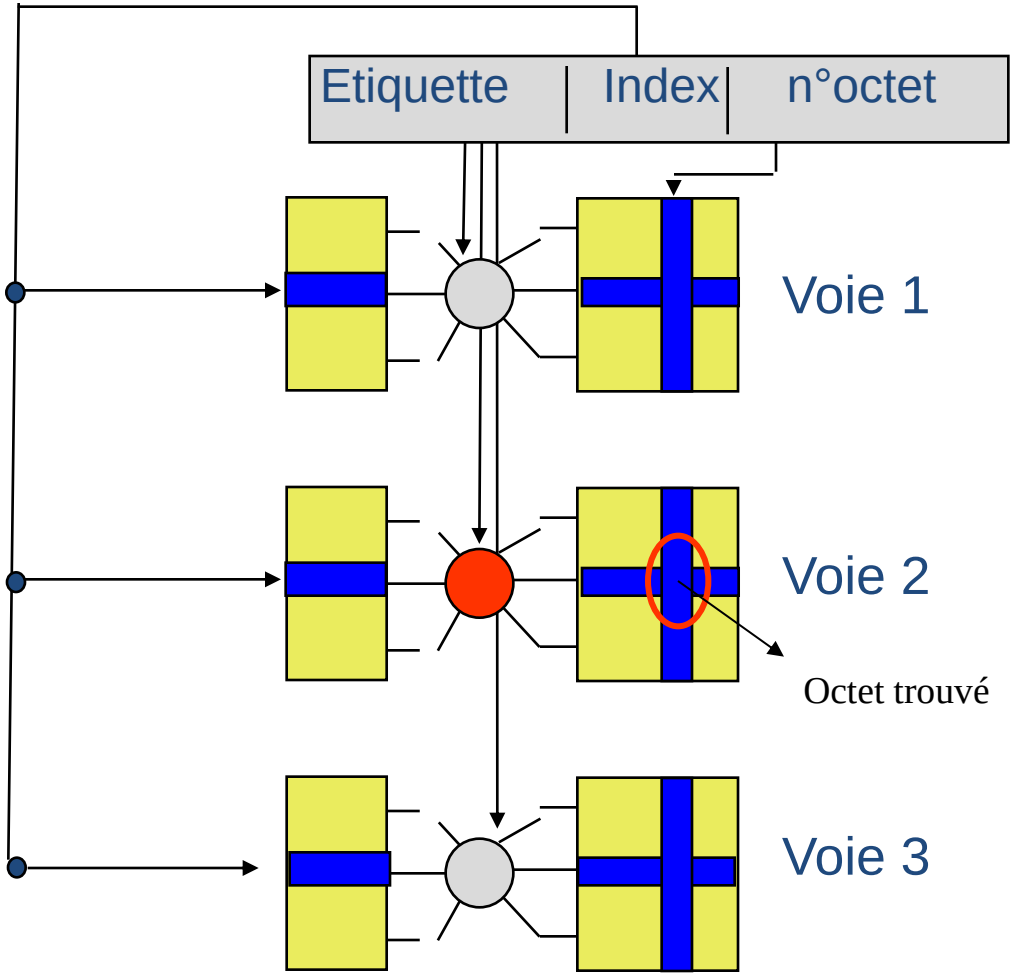



➢ L'adresse de l'octet est utilisée comme pour le cache à correspondance directe : l'index sélectionne une ligne dans toutes les voies.

➢ Le contenu de chaque entrée est comparé de façon associative avec l'étiquette de l'adresse.

➢ Le nombre de voies du cache constitue le degré d'associativité du cache (3)

# Cache Mixte



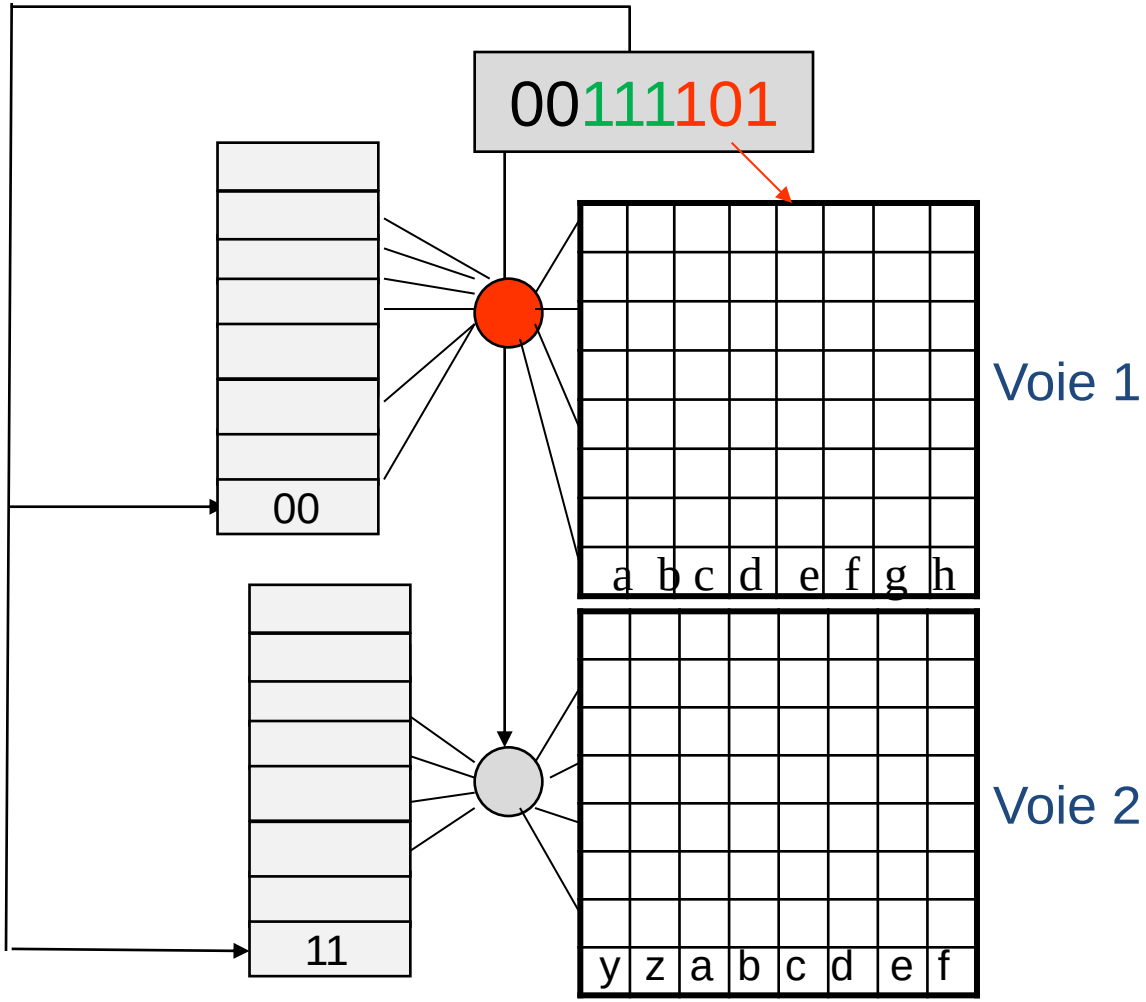

Si Répertoire[Index] Contient Etiquette Alors Charger MemoireUtile[Voie,Index,n°octet]
Sinon Choisir Voie pour remplacer Ligne
Remplacer Ligne dans Voie choisie
Charger MémoireUtile[Voie choisie ,Index,n°octet]

FinSi

# Cache Mixte

Load D R1 $3D_{16}$

SUCCES sur la voie 1




Si Répertoire[111] Contient 00 Alors Charger MemoireUtile[voie 1, entrée 111, octet 101]
Sinon Choisir Voie pour remplacer Ligne
Remplacer Ligne dans Voie choisie
Charger MémoireUtile[Voie choisie,Index,n°octet]

FinSi

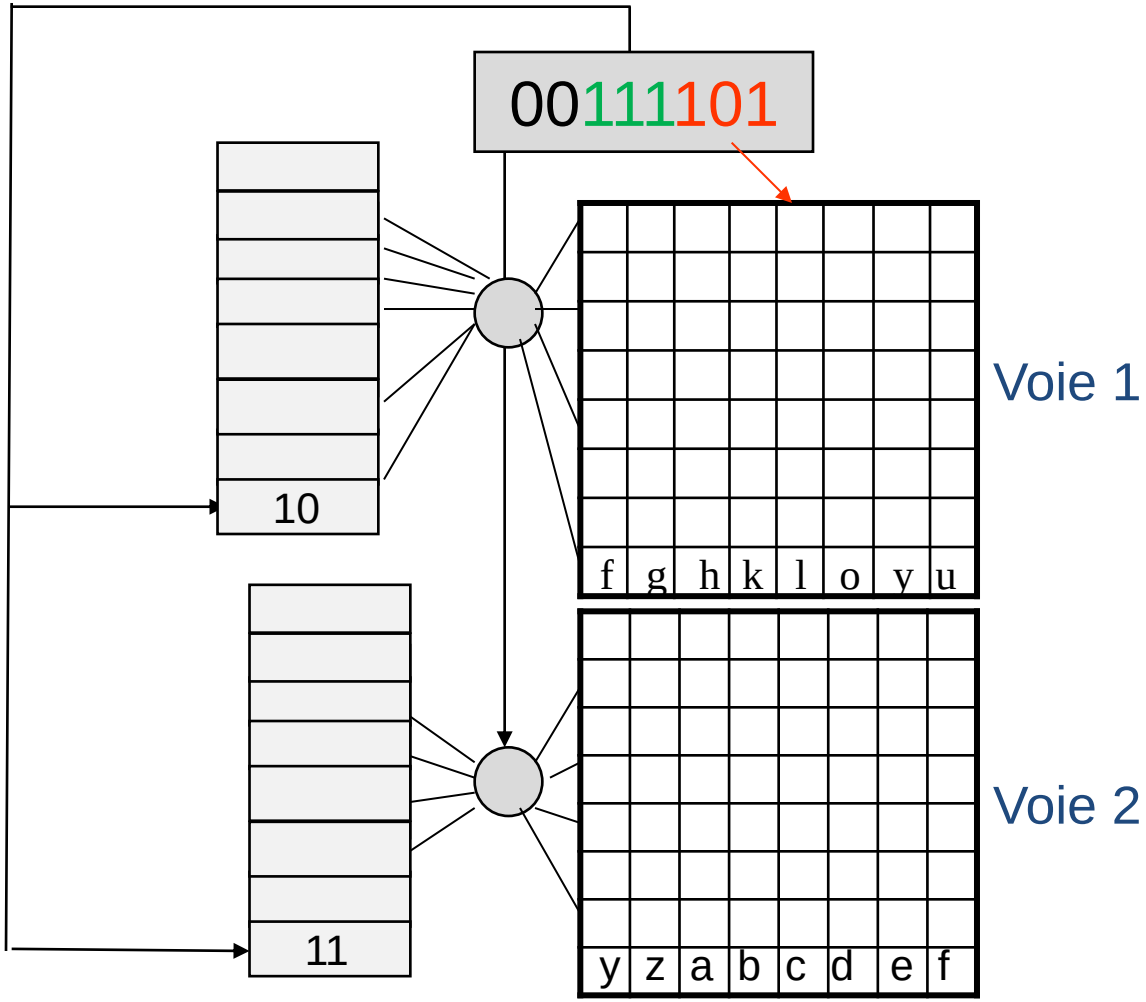
# Cache Mixte

Load D R1 $3D_{16}$

DEFAUT; aucune des deux Voies ne contient la bonne Étiquettes.

- Si une voie est libre, on la remplit
- Sinon Le choix de la voie à remplacer s'effectue à l'aide d'un algorithme

de remplacement de lignes (FIFO, LRU, NMRU)




Si Répertoire[111] Contient 00 Alors Charger MemoireUtile[voie 1, entrée 111, octet 101]

Sinon Choisir Voie pour remplacer Ligne

Remplacer Ligne dans Voie choisie

Charger MémoireUtile[Voie choisie,Index,n°octet]

FinSi

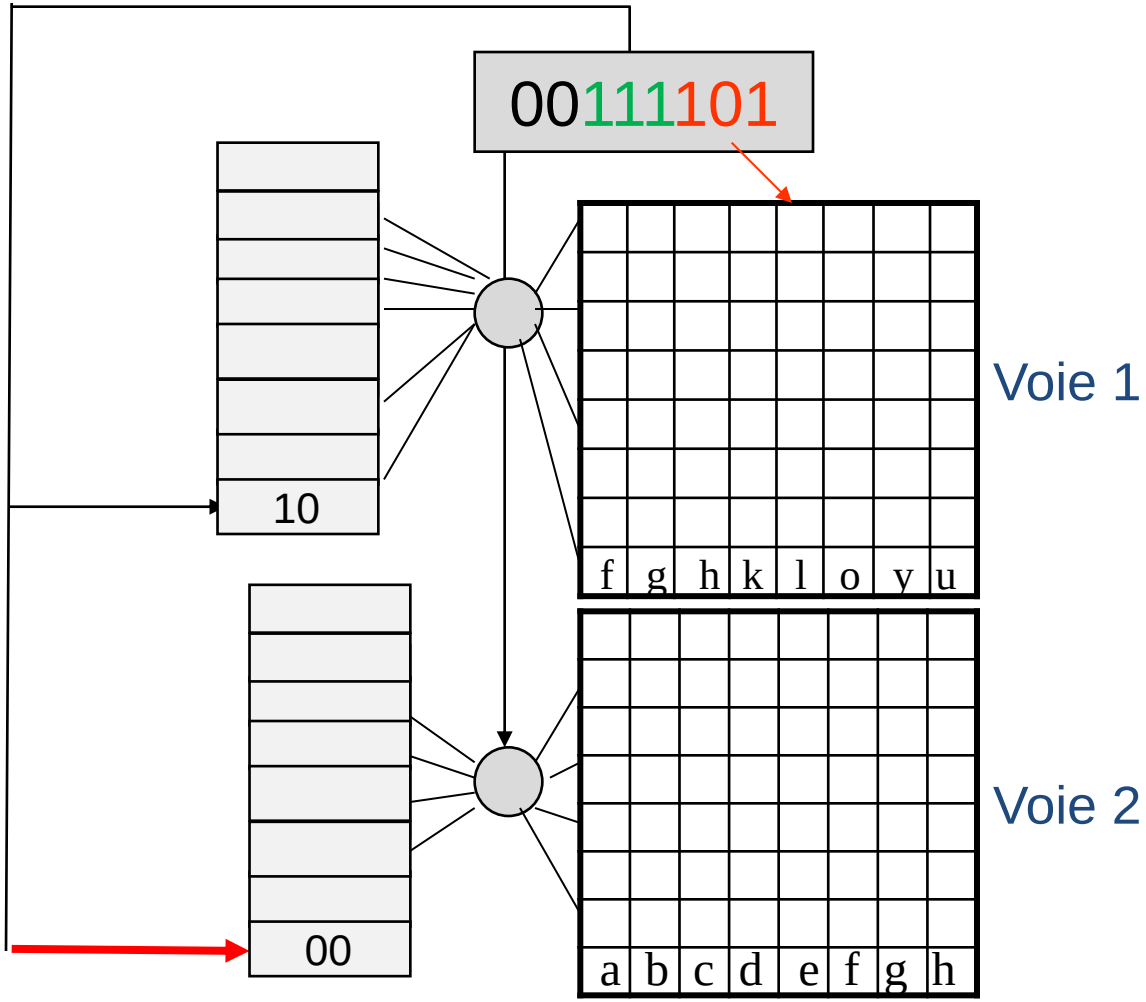
# Cache Mixte

Load D R1 $3D_{16}$

DEFAUT; aucune des deux Voies ne contient la bonne Étiquettes.

- Si une voie est libre, on la remplit
- Sinon Le choix de la voie à remplacer s'effectue à l'aide d'un algorithme

de remplacement de lignes (FIFO, LRU, NMRU)




Si Répertoire[111] Contient 00 Alors Charger MemoireUtile[voie 1, entrée 111, octet 101]

Sinon Choisir Voie pour remplacer Ligne

Remplacer Ligne dans Voie choisie

Charger MémoireUtile[Voie choisie,Index,n°octet]

FinSi

# Cache mixte ou associatif par blocs

- Solution intermédiaire en terme de coût et d'encombrement
- Complexité : politique de remplacement de ligne.
- Performance intermédiaire : les différentes voies du cache permettent de réduire le nombre d'échecs par collision.
- Format d'une entrée de cache (répertoire) similaire à celui du cache associatif

# Quelques exemples de caches

- Processeur Intel 486
  - Cache L1 unifié données et instructions
  - Capacité 8 Ko
  - Cache mixte à 4 voies  de 128 lignes de 16 octets

- Processeur pentium
  - Cache L1 séparé données et instructions
  - Chaque cache a une capacité de 8 Ko
  - Chaque cache est un cache mixte à 2 voies de 128 lignes de 32 octets.