

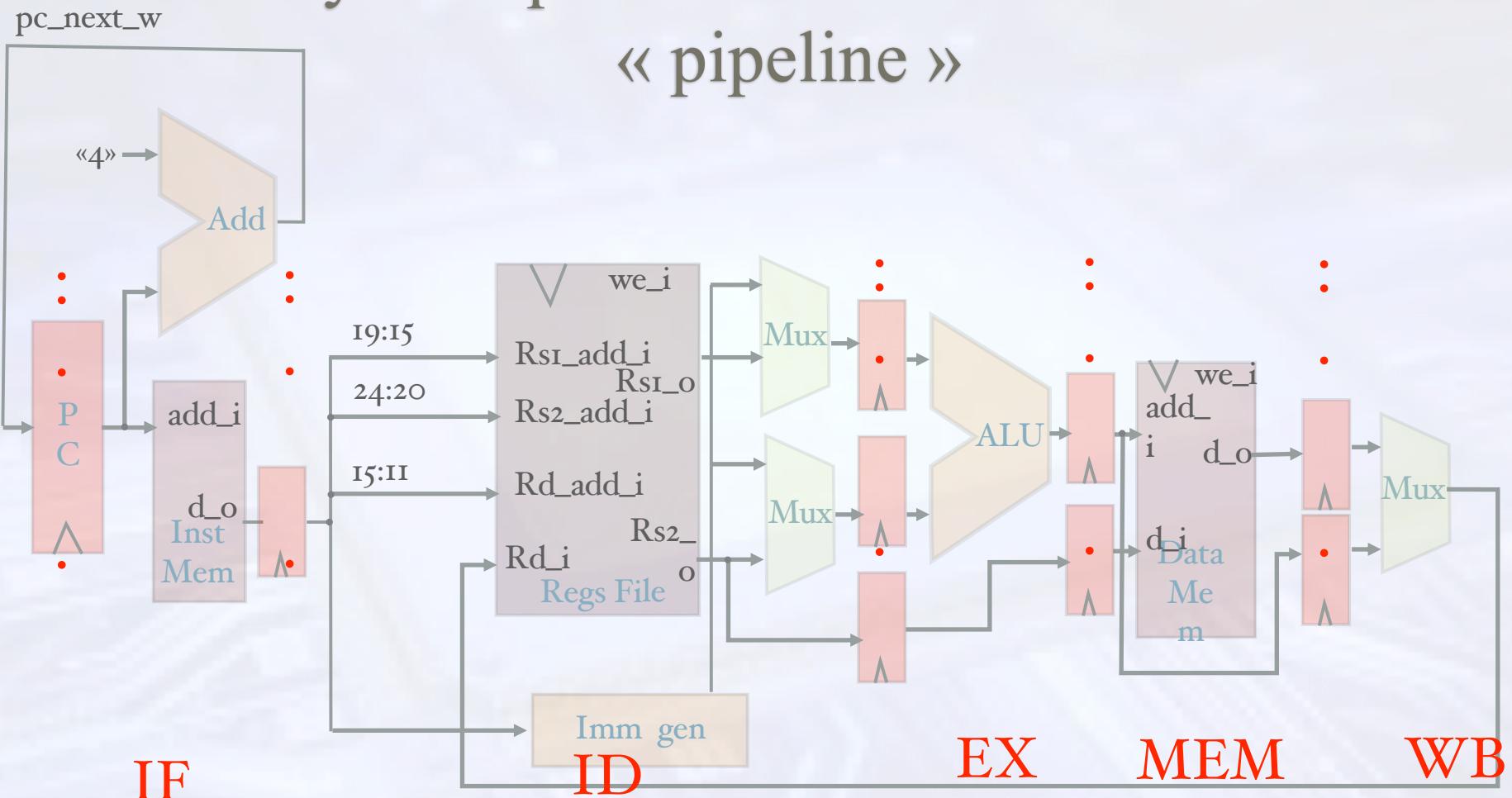
Architecture des microprocesseurs hiérarchie mémoire

Michel Agoyan
Marc Lacruche
Simon Pontie
Olivier Potin
Come Allart
Raphaël Comps

: michel.agoyan@st.com
: marc.lacruche@st.com
: simon.pontie@cea.fr
: olivier.potin@emse.fr
: come.allart@emse.fr
: rcomps@emse.fr

Architecture µP: Hiérarchie mémoire

Analyse de performance de l'architecture « pipeline »



$$* \quad T = \max(t_{IF}, t_{ID}, t_{EX}, t_{MEM}, t_{WB})$$

Architecture µP: Hiérarchie mémoire

Analyse de performance de l'architecture « pipeline »

$$T = \max(t_{IF}, t_{ID}, t_{EX}, t_{MEM}, t_{WB})$$

Les étages les plus critiques sont ceux pour lesquels un accès mémoire est effectué : t_{IF} et t_{MEM}

Exemple si : $\left. \begin{array}{l} t_{IF} = t_{MEM} = 2 * t_u \\ t_{ID} = t_{EX} = t_{WB} = t_u \end{array} \right\} \rightarrow T = 2 * t_u$

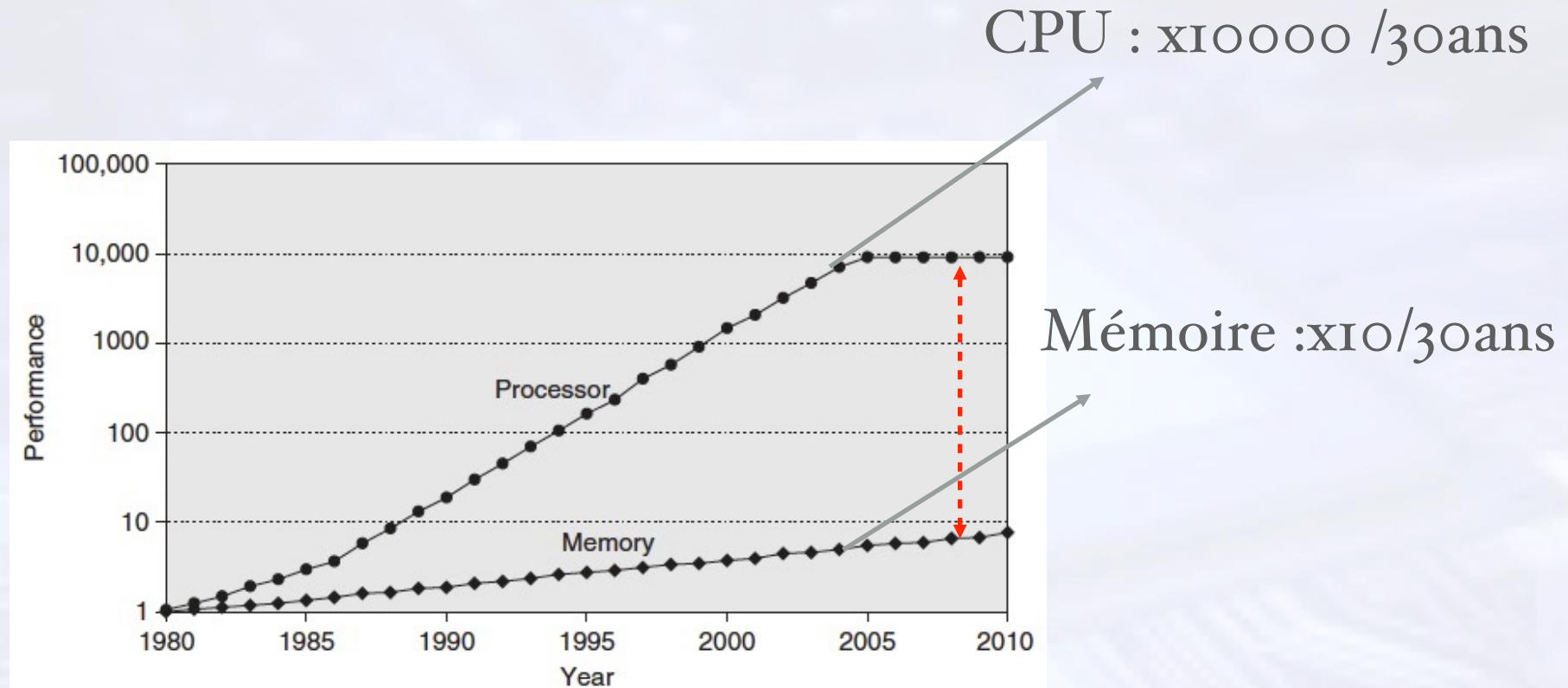
Pour l'architecture monocycle : $T = 7 * t_u$

L'efficacité du pipeline devient maximale lorsque tous les étages ont une durée de traitement égale

Comment réduire le temps d'accès à la mémoire ?

Architecture µP: Hiérarchie mémoire

Performance rappel



Ref : David Patterson UC Berkeley

Architecture µP: Hiérarchie mémoire SRAM

Static Random Access Memory

élément mémoire

4 transistors(mémoire) +2 transistors
d'accès

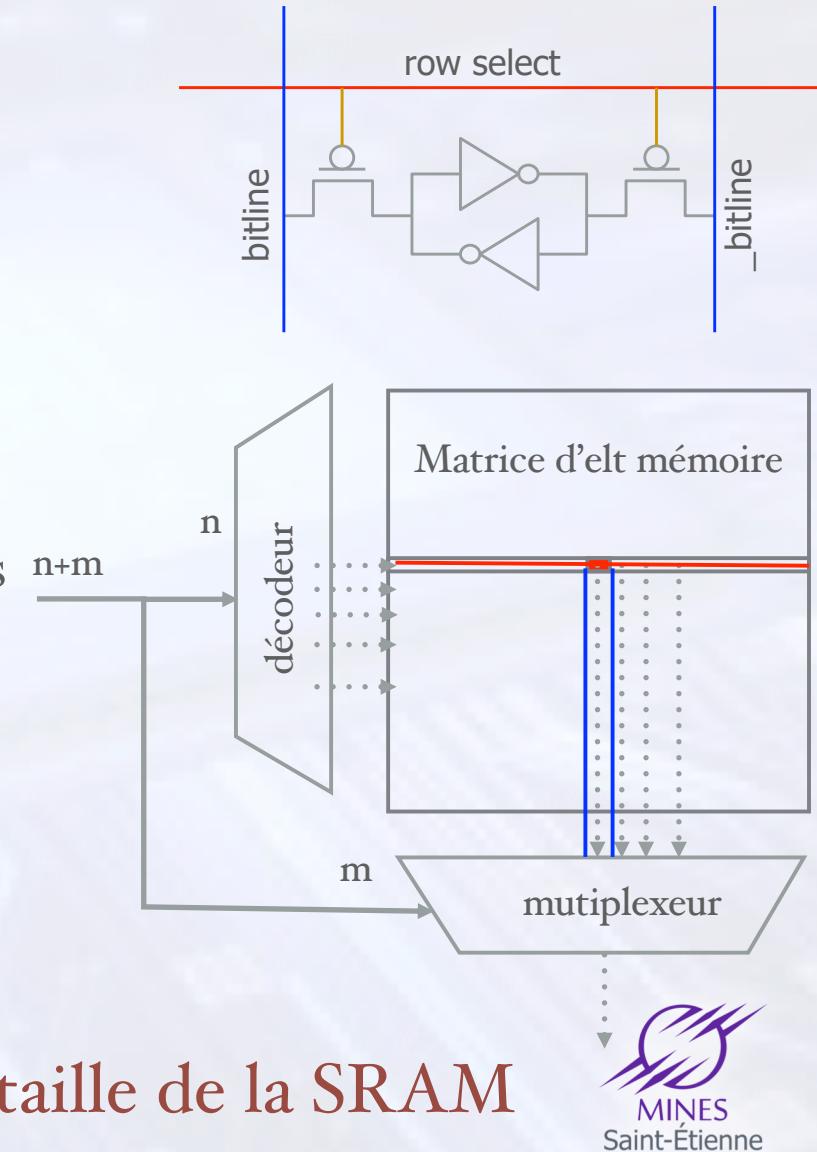
Organisation en matrice :

Compromis entre la taille mémoire et temps
d'accès :

dimension des lignes d'accès aux elts
mémoires ↗ avec la taille de matrice

Largeur m ↗ avec la dimension de la
matrice \Rightarrow ↗ délai du multiplexeur de sortie

temps d'accès ↗ avec la taille de la SRAM



Architecture µP: Hiérarchie mémoire DRAM

Dynamique Random Access Memory

élément mémoire

1 transistor d'accès + 1 capacité (mémoire)

lecture destructive

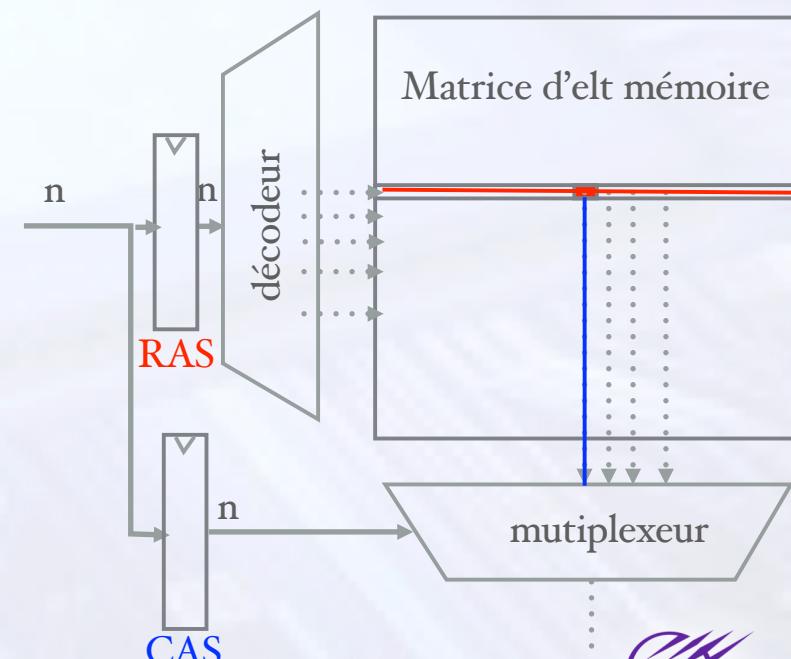
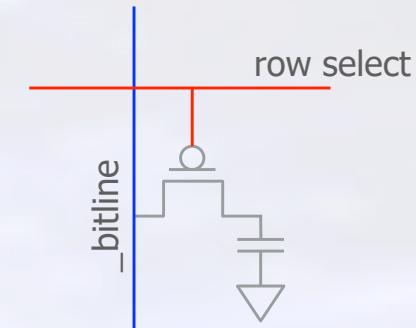
perte ohmique (limite temporelle)

⇒ nécessite un mécanisme de rafraîchissement (ms)

forte densité

Organisation en matrice :

Etant donné leur forte capacité (Gbit) : RAS puis CAS



Architecture µP: Hiérarchie mémoire SRAM/DRAM

SRAM		
64KB	256 KB	1MB
1 ns	10 ns	20 ns

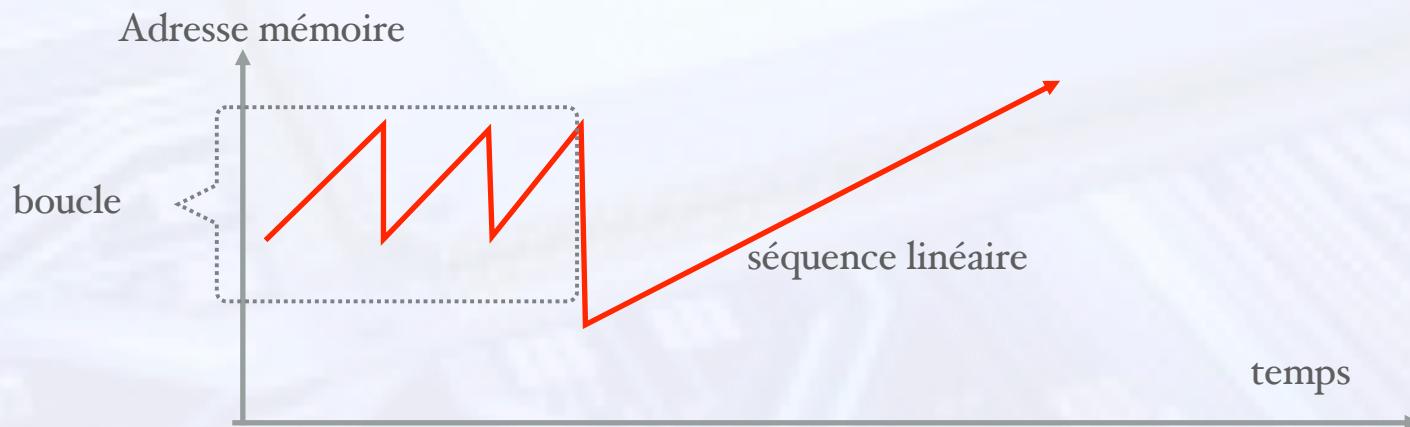
DRAM				
capacité	type de DRAM	freq	temps d'accès CAS (nb de périodes)	temps d'accès RAS (ns)
	DDR2-1066	533MHz	4	7.5ns
	DDR3-1066	533MHz	7	13ns
	DDR3-1600	800MHz	6	7.5 ns
	DDR4-1600	800MHz	10	12.5ns

temps d'accès ↗ avec la taille mémoire > (>>) temps de cycle processeur

Architecture µP: Hiérarchie mémoire

Propriétés de localité des programmes

- * Le flot d'exécution d'un programme est une succession de séquences linéaires, de boucles, manipulant les même structures de données :
- * Localité temporelle : si un emplacement est accédé à un instant il a de forte chance d'être accédé de nouveau dans **un futur proche**
- * Localité spatiale : si un emplacement est accédé à un instant il a de forte chance qu'un **emplacement voisin** soit accédé dans un futur proche



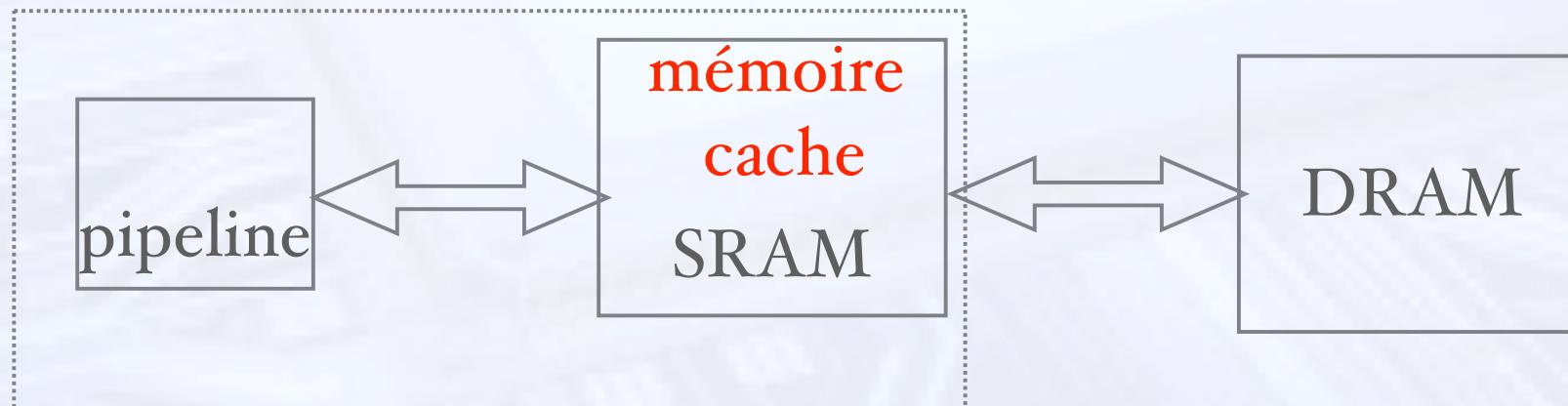
Architecture µP: Hiérarchie mémoire

Cache principe

Idée : pour diminuer l'impact du temps d'accès des mémoires de forte capacité :

Exploiter les propriétés de localité temporelle et spatiale :

Stocker les données récentes dans une mémoire de plus faible capacité mais plus rapide (sans impact sur CPI) :
mémoire cache



temps accès SRAM << temps accès DRAM
taille SRAM << taille DRAM

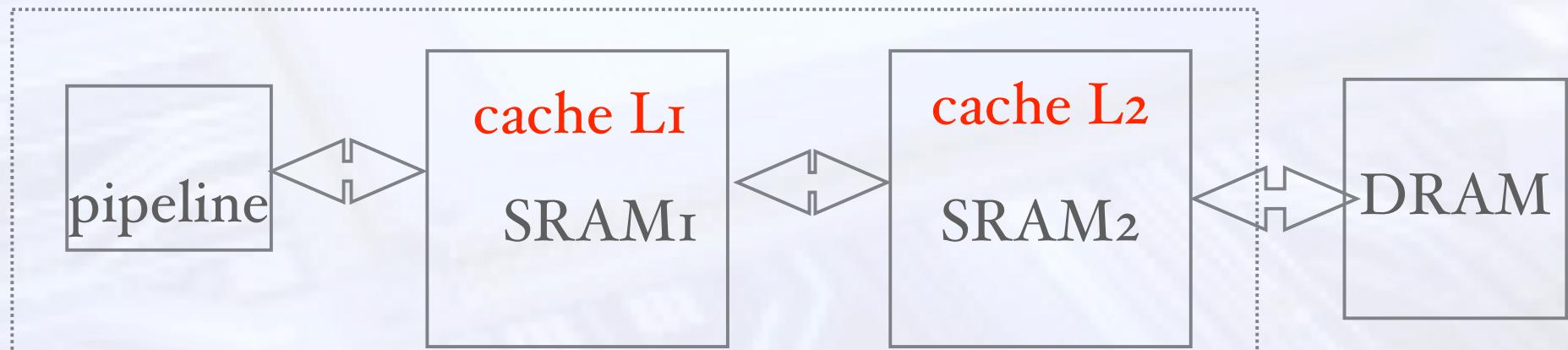
Architecture µP: Hiérarchie mémoire

Cache : Hiérarchie

le temps de cycle du pipeline donne la limite du temps d'accès de la mémoire cache et donc de sa taille

Si la taille du cache est trop petite on réduit son efficacité

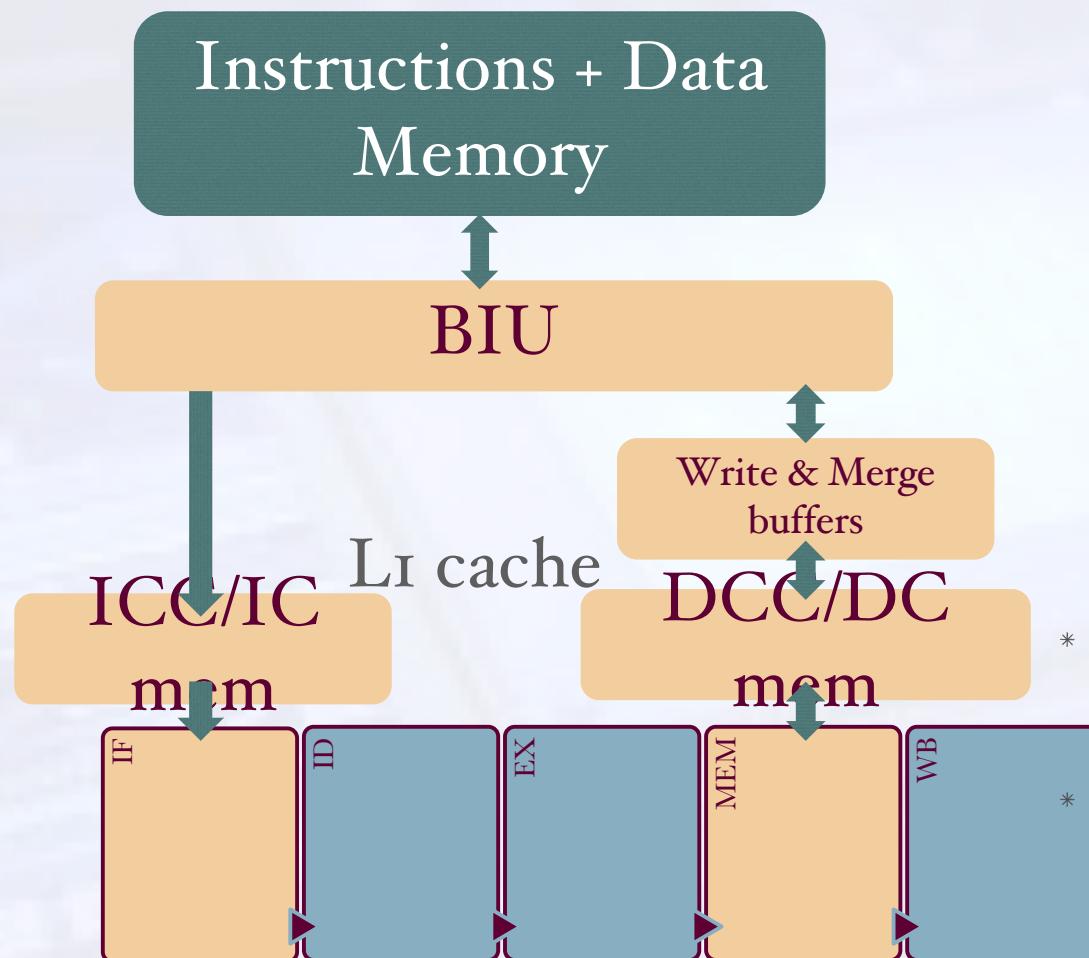
Pour compenser, on crée une hiérarchie mémoire : L1,L2



Taille SRAM₁ < taille SRAM₂
temps d'accès SRAM₁ < temps d'accès SRAM₂

Architecture µP: Hiérarchie mémoire

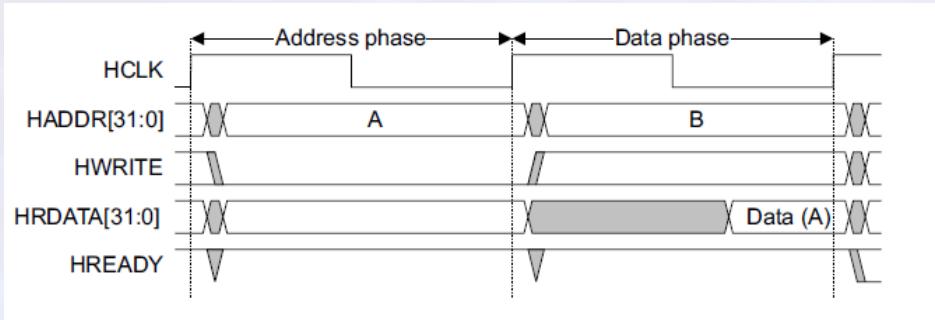
Cache : mémoire unifiée



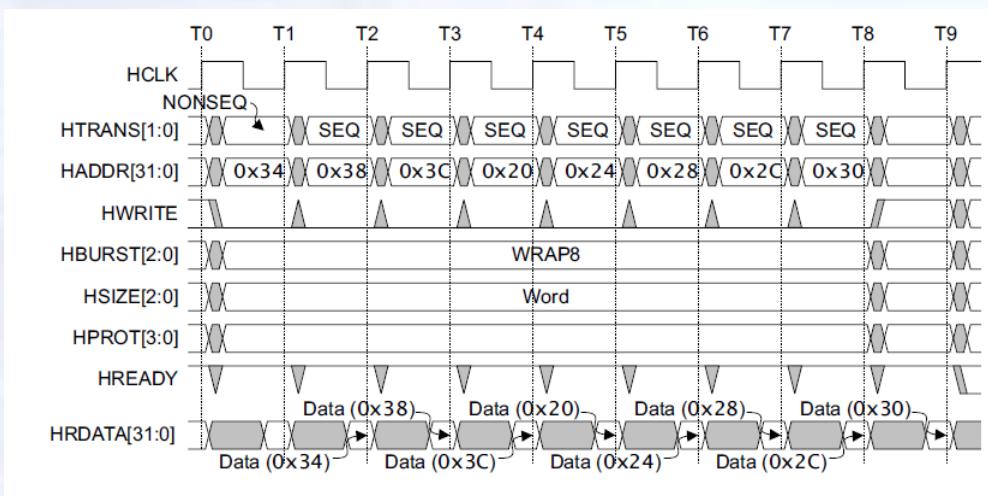
- * Deux séquenceurs permettent de remplir les mémoires cache :
 - * **ICC (Instruction Cache Controller)**
 - * **DCC (Data Cache Controller)**
- * **Write & Merge buffers** optimisent les opérations d'écritures : on favorise les transferts « larges »
- * Les requêtes ICC & DCC sont gérées par le **BIU (Bus Interface Unit)**
- * On retrouve une architecture à **mémoire unifiée** (Instructions+ données)

Architecture µP: Hiérarchie mémoire

Cache : BIU



Accès simple



Accès mode rafale

Exemple bus AMBA ARM

Accès simple en 2 cycles

Accès en mode rafale
("Burst") :

Adresses successives =>
temps d'accès de la
mémoire plus court

1 cycle

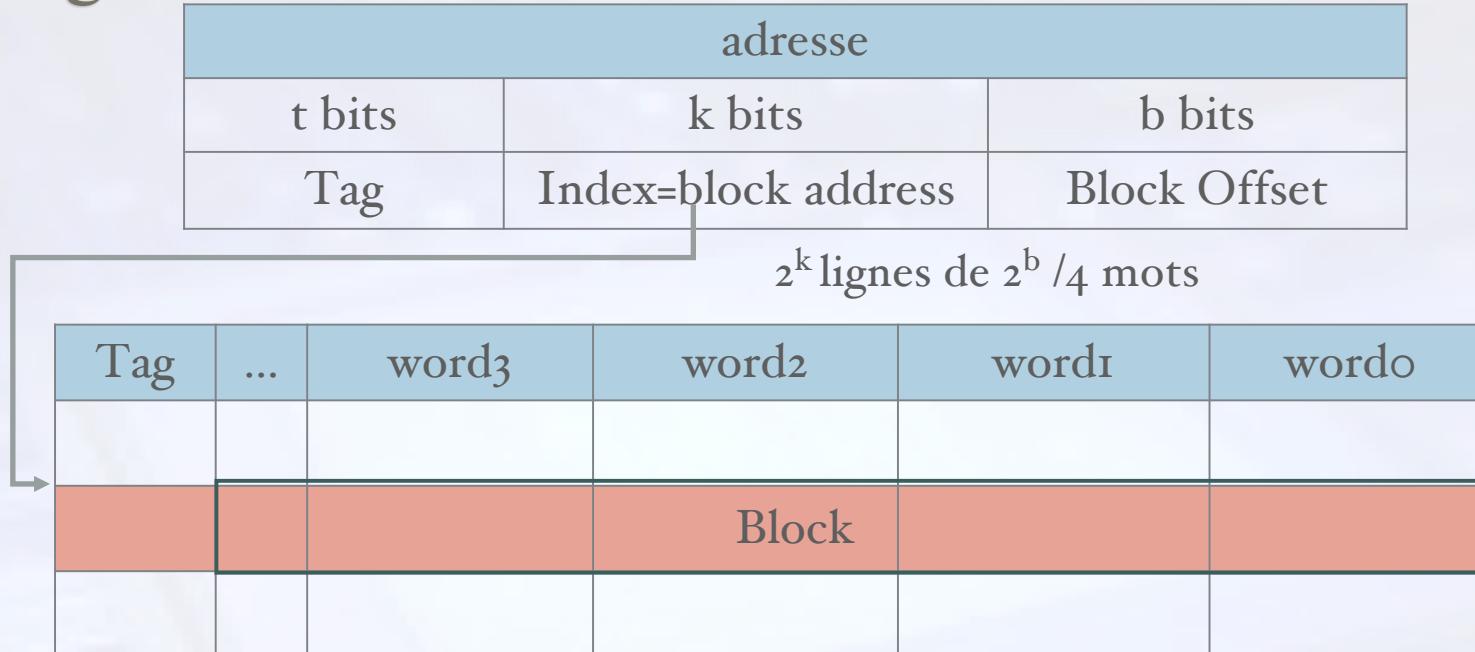
Architecture µP: Hiérarchie mémoire

Cache principe

- * Définitions :
- * HIT (donnée \in mémoire cache) \Rightarrow temps d'accès faible
- * MISS (donné \notin mémoire cache) \Rightarrow temps d'accès important
- * Hit rate = nb d'accès mémoire \in cache /nb d'accès mémoire total, hit time =temps d'accès à la mémoire cache
- * Miss rate = $1 - \text{Hit rate}$, pénalité =temps d'accès pour un cache miss
- * Temps d'accès moyen = hit time + (miss rate * pénalité)
- * CPI effectif = CPI + Miss rate * pénalité * Nb d'accès mémoire /instruction

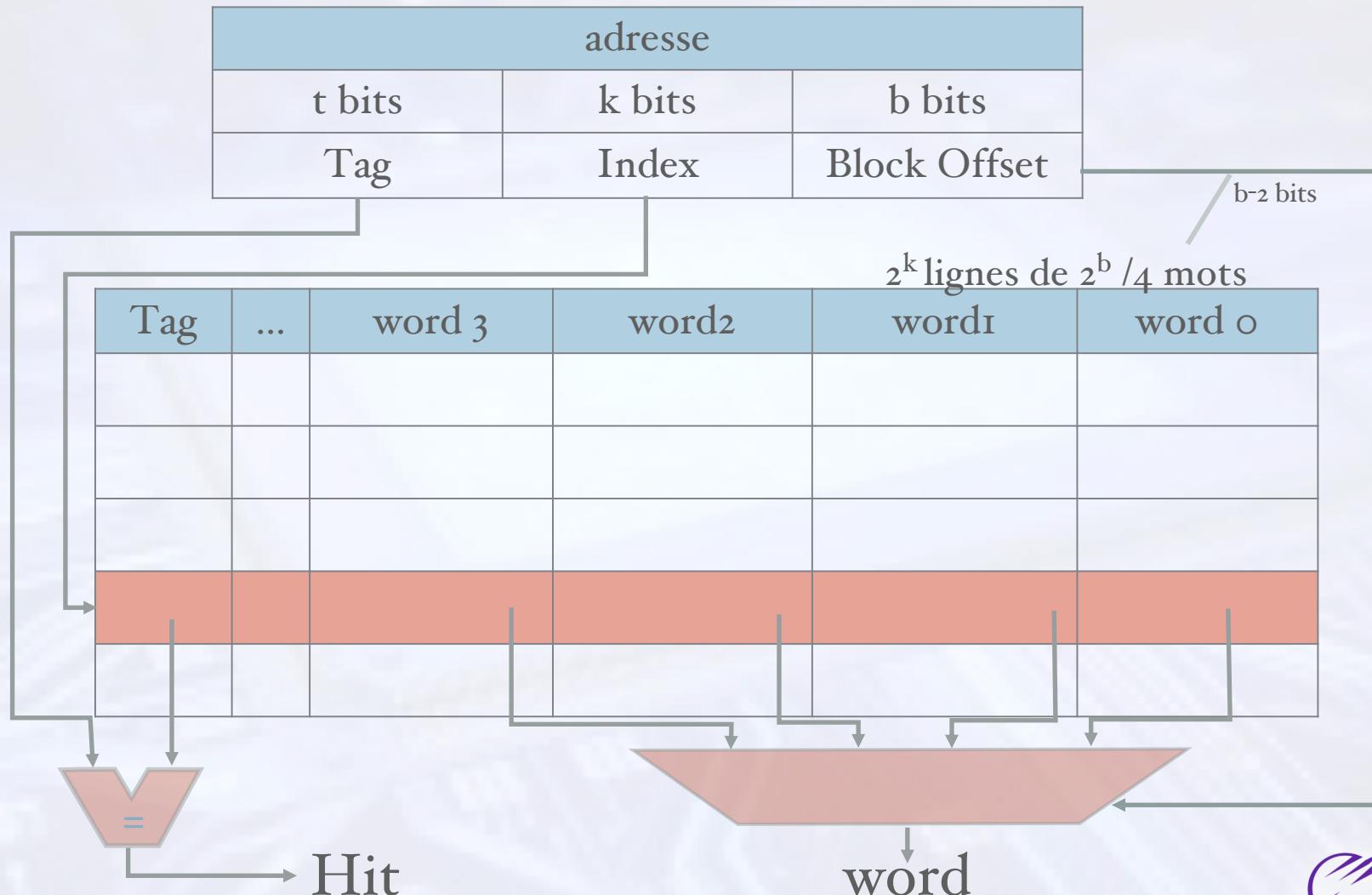
Architecture µP: Hiérarchie mémoire

Organisation de la mémoire cache/taille des blocs



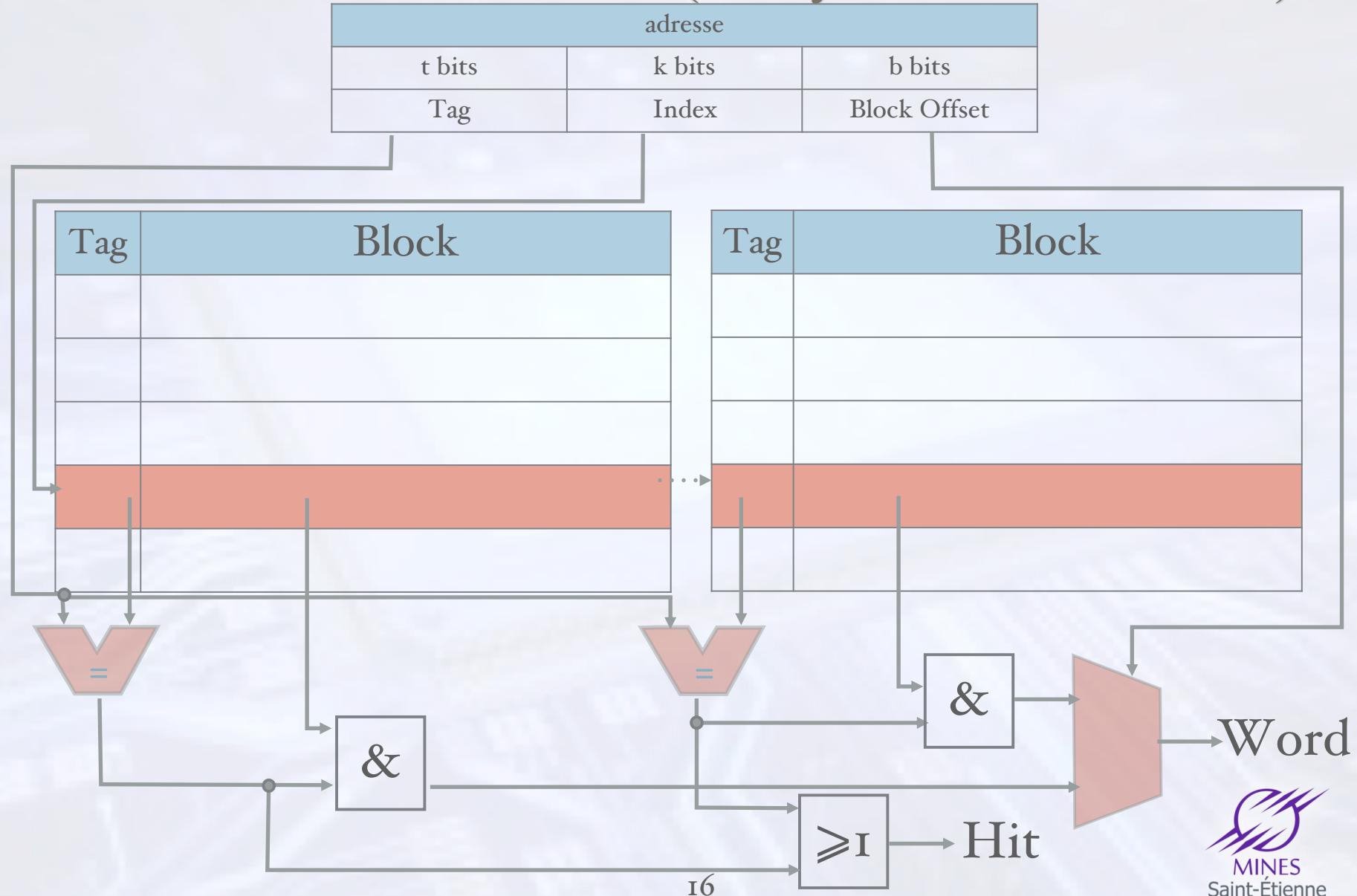
- * Exemple :
 - * $b=4 \Rightarrow$ taille des lignes = $2^4 / 4 = 4$ mots
 - * $k=6 \Rightarrow 2^6 = 64$ lignes de 16 octets = $1024 = 1KB$
 - * $32 - k - b = t = 22$
- * De grandes tailles de lignes peuvent tirer avantage du mode rafale (burst) de certains bus ou mémoire et réduire la collision de tag mais la pénalité de miss devient plus importante \Rightarrow à nouveau, des compromis à choisir

Architecture µP: Hiérarchie mémoire Cache direct(Direct-Mapped Cache)



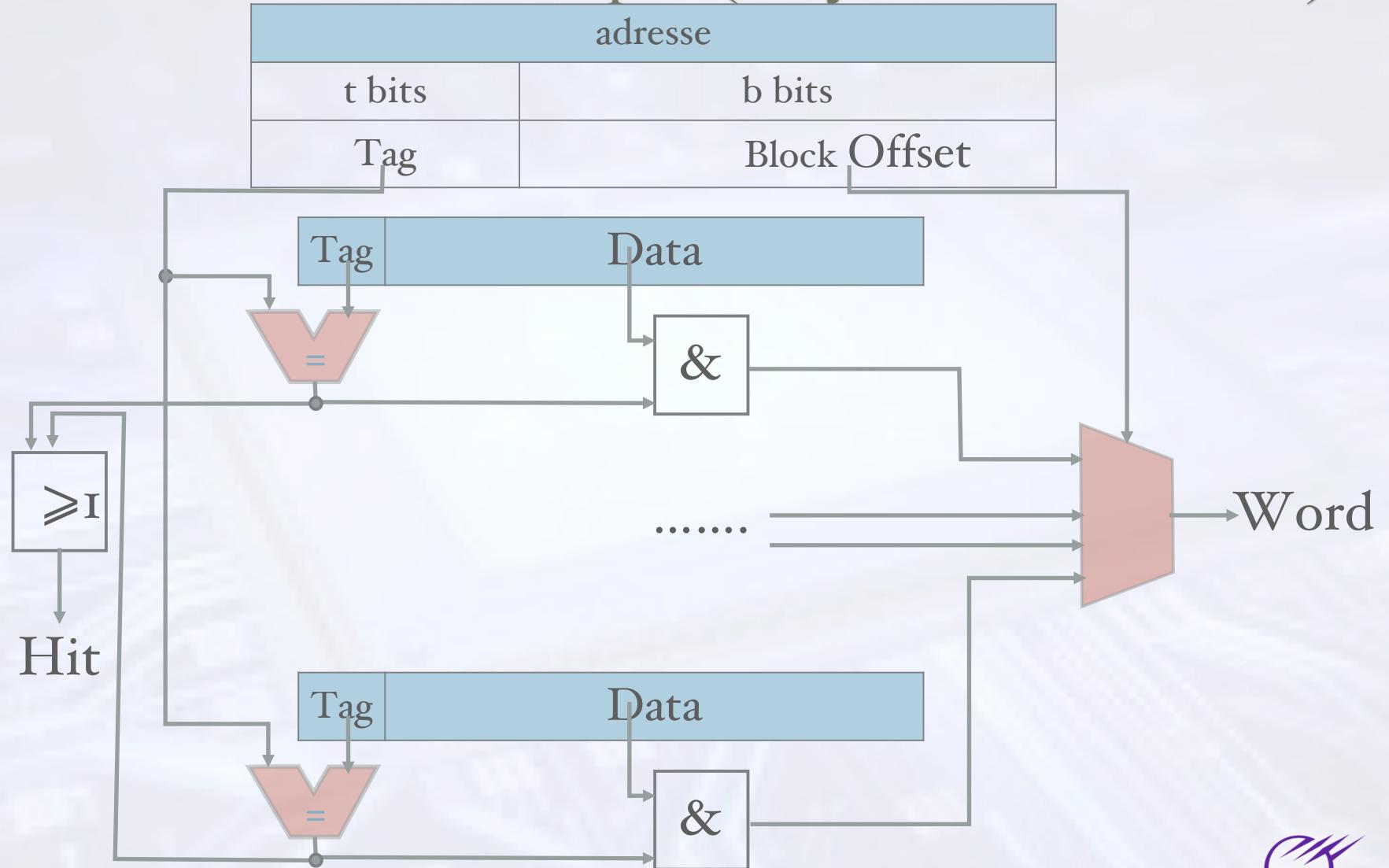
Architecture µP: Hiérarchie mémoire

Cache associatif 2 voies (2 ways associative cache)



Architecture µP: Hiérarchie mémoire

Cache associatif complet (fully associative cache)

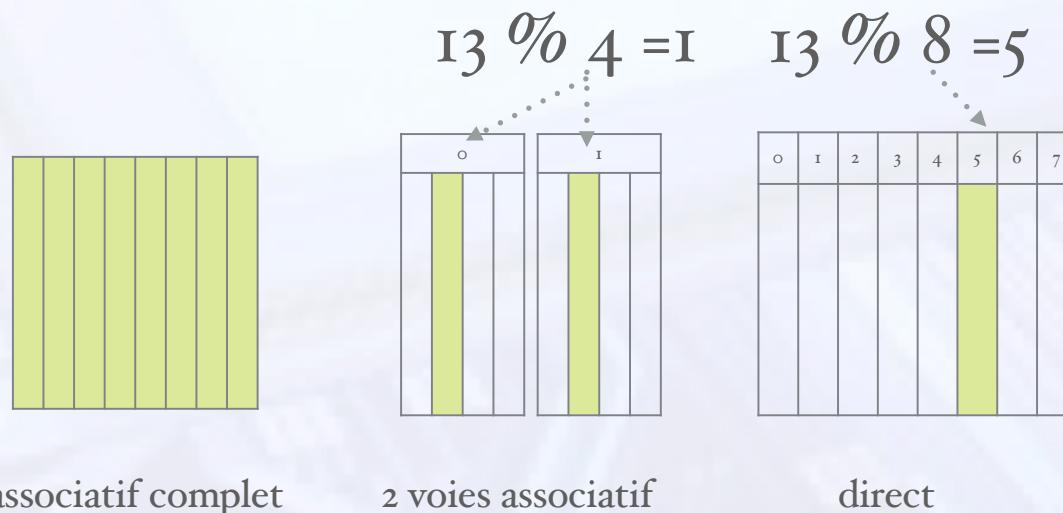


Architecture µP: Hiérarchie mémoire

Cache : organisation en blocs

- * La mémoire cache est divisée en blocs (taille = 2^n)
- * On cache des blocs de la même taille de la mémoire principale

numéro de bloc	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	..
mémoire																	



Exercice : et pour le bloc le bloc 29 ?

Architecture µP: Hiérarchie mémoire remplacement des lignes de cache

- * Que faire quand la ligne de cache « victime » est occupée ?
- * Plusieurs stratégies existent :
 - * Least Recently Used(**LRU**)
 - * Facile à implémenter sur des caches à 2 voies
 - * **FIFO**
 - * utilisé pour les caches associatifs à plusieurs voies
 - * Not Least Recently Used (**NLRU**)
 - * utilisé pour améliorer le mode FIFO
 - * MFU (Most Frequently Used)
 - * **Aléatoire**

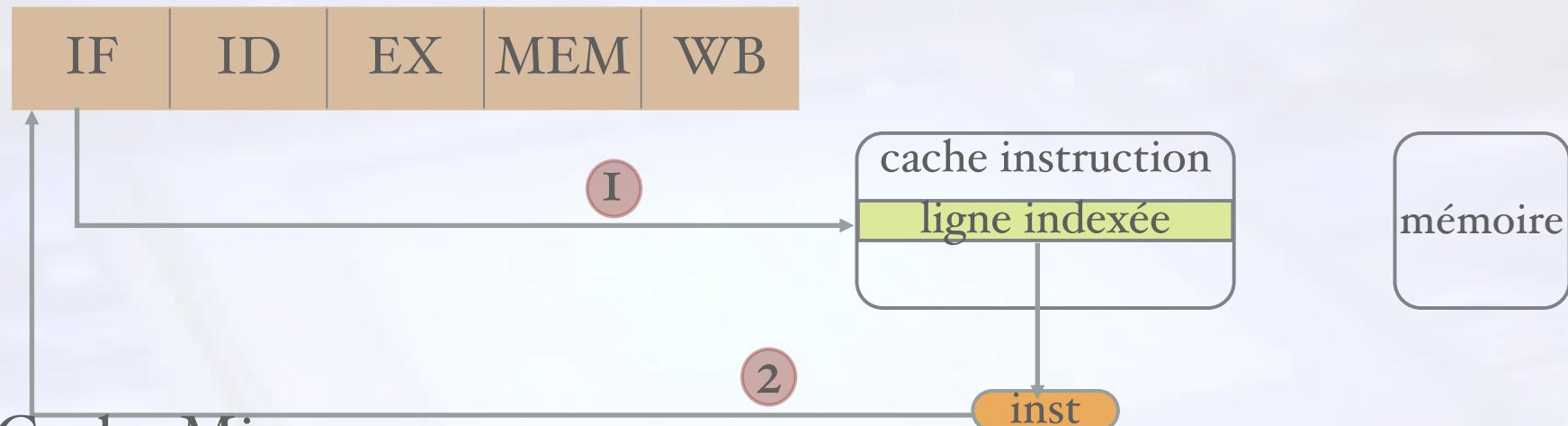
Architecture µP: Hiérarchie mémoire

Cache différent mode de gestion de l'écriture

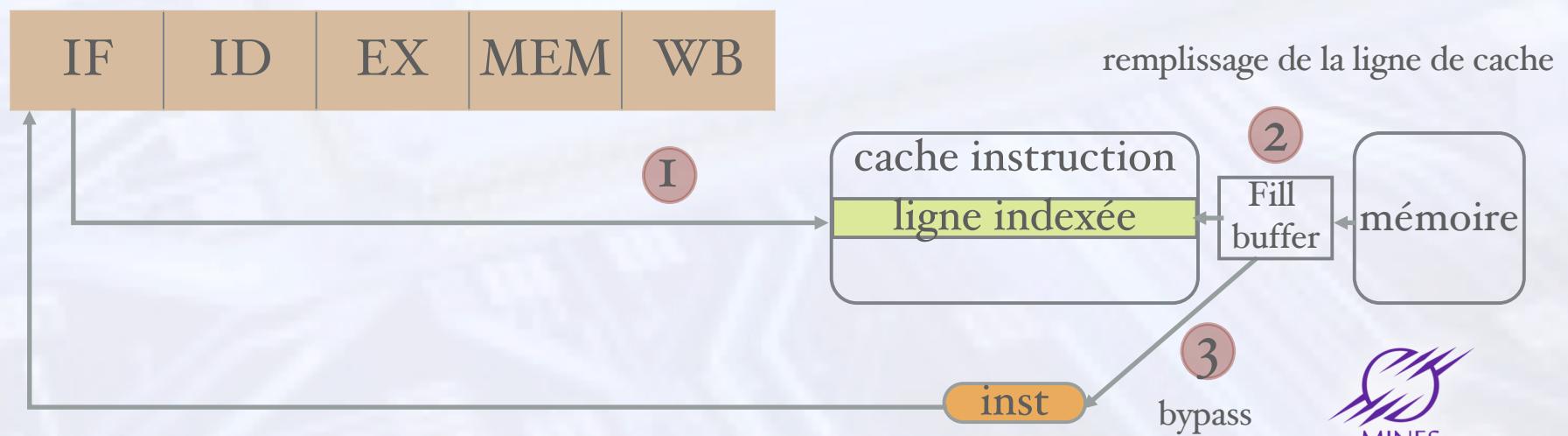
- * Cache Hit :
 - * « Write Through » :
 - * on met à jour à la fois la ligne de cache et la mémoire
 - * Simplifie la gestion de cohérence du cache mais impacte les performances
 - * limitation de l'impact par l'ajout d'un buffer d'écriture « Write buffer »
 - * « Write back » :
 - * On met seulement à jour la ligne de la mémoire cache
 - * Les données seront écrites en mémoire lorsque la ligne est invalidée (utilisation d'un bit indicateur : « dirty bit »)
- * Cache Miss :
 - * sans allocation (no write allocate) on ne met pas jour la mémoire cache
 - * avec allocation

Architecture µP: Hiérarchie mémoire lecture du cache instruction

- * Cache Hit :

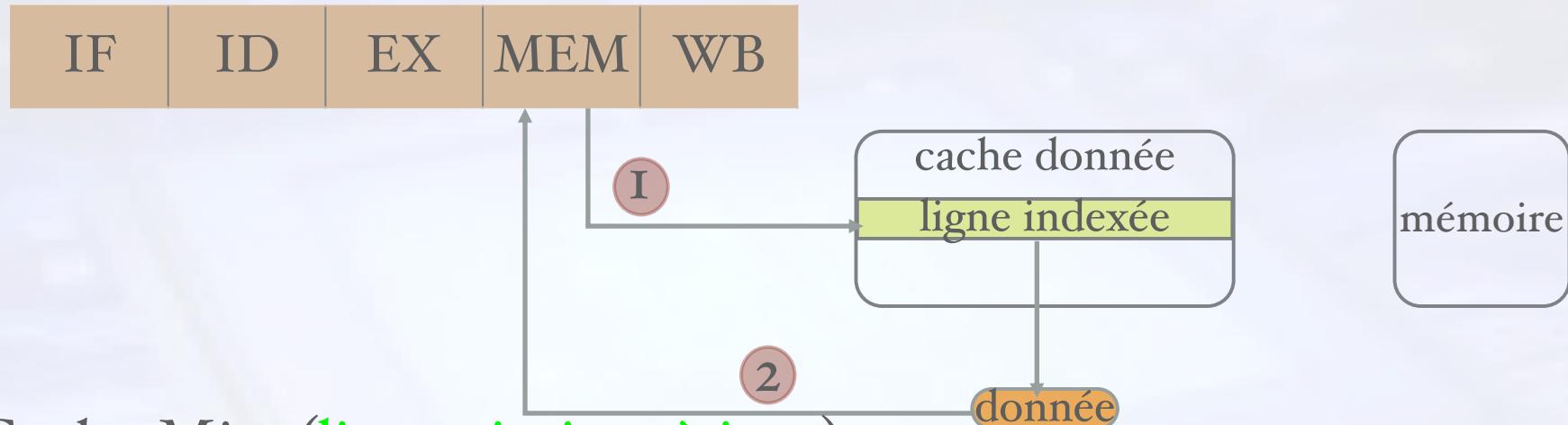


- * Cache Miss:

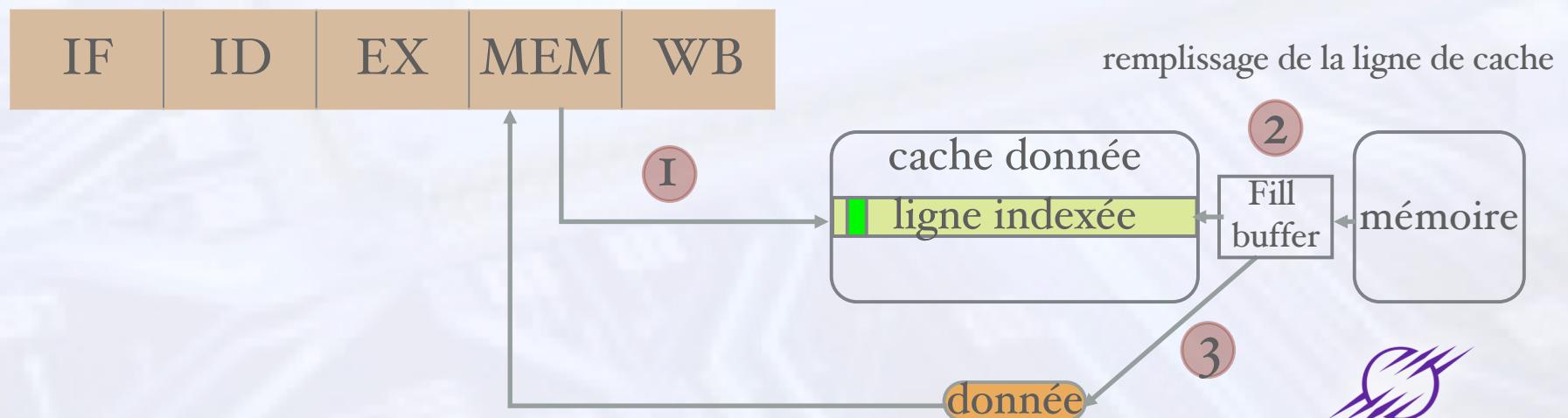


Architecture µP: Hiérarchie mémoire lecture du cache donnée

- * Cache Hit :

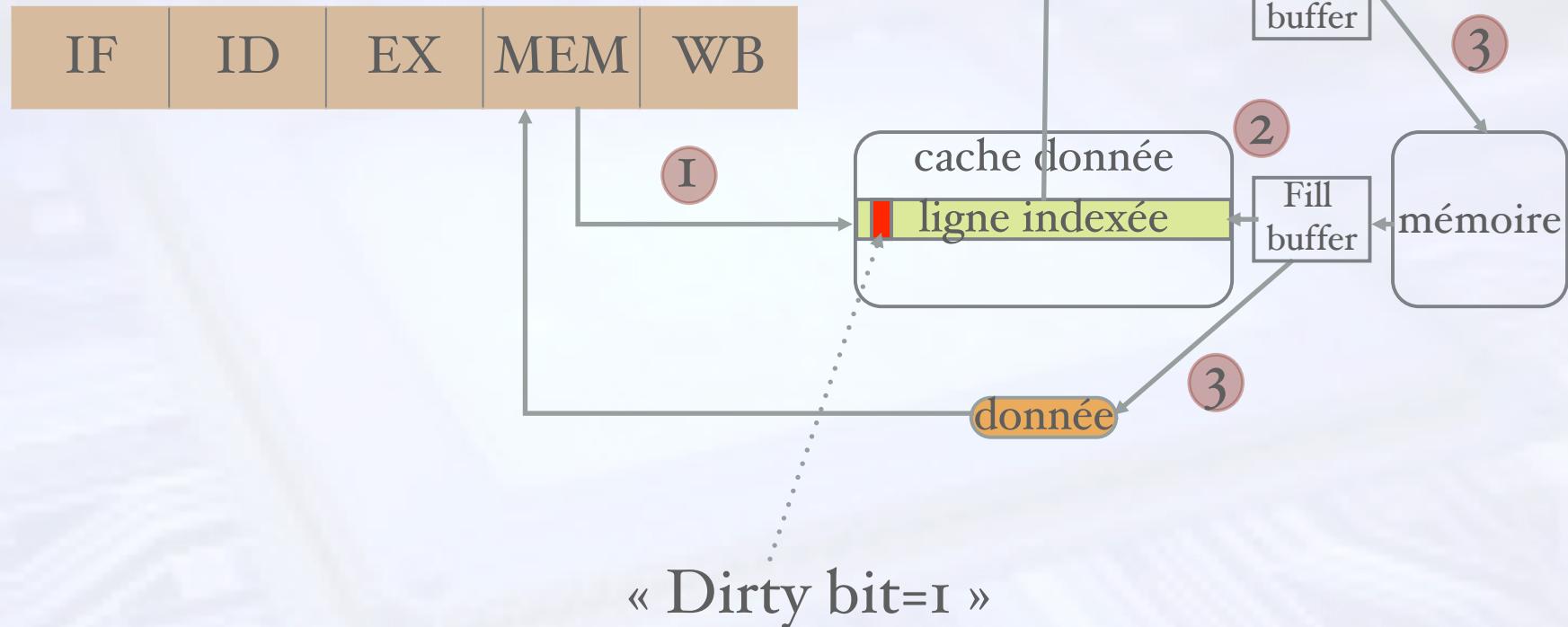


- * Cache Miss (ligne victime à jour):



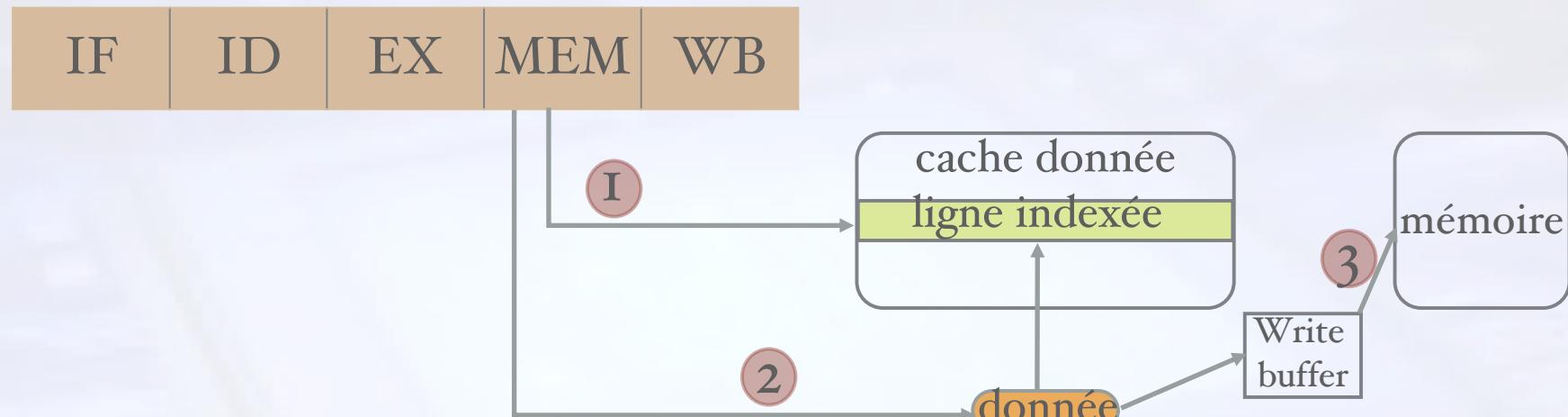
Architecture µP: Hiérarchie mémoire lecture du cache donnée

- * Cache Miss (ligne victime non à jour):

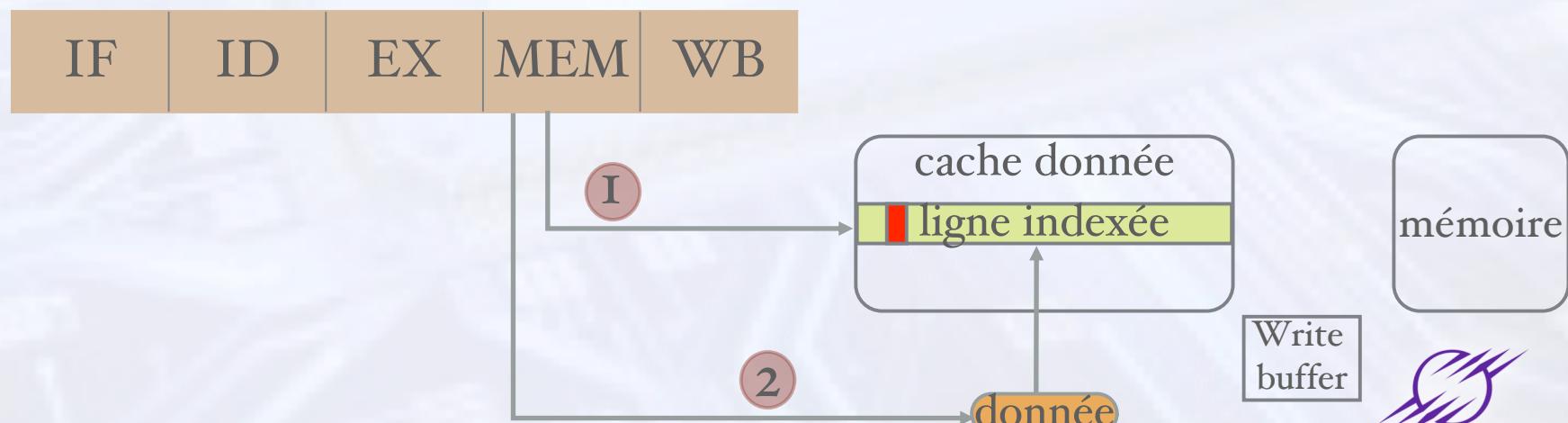


Architecture µP: Hiérarchie mémoire écriture du cache donnée

- * Cache Hit (Write-through):

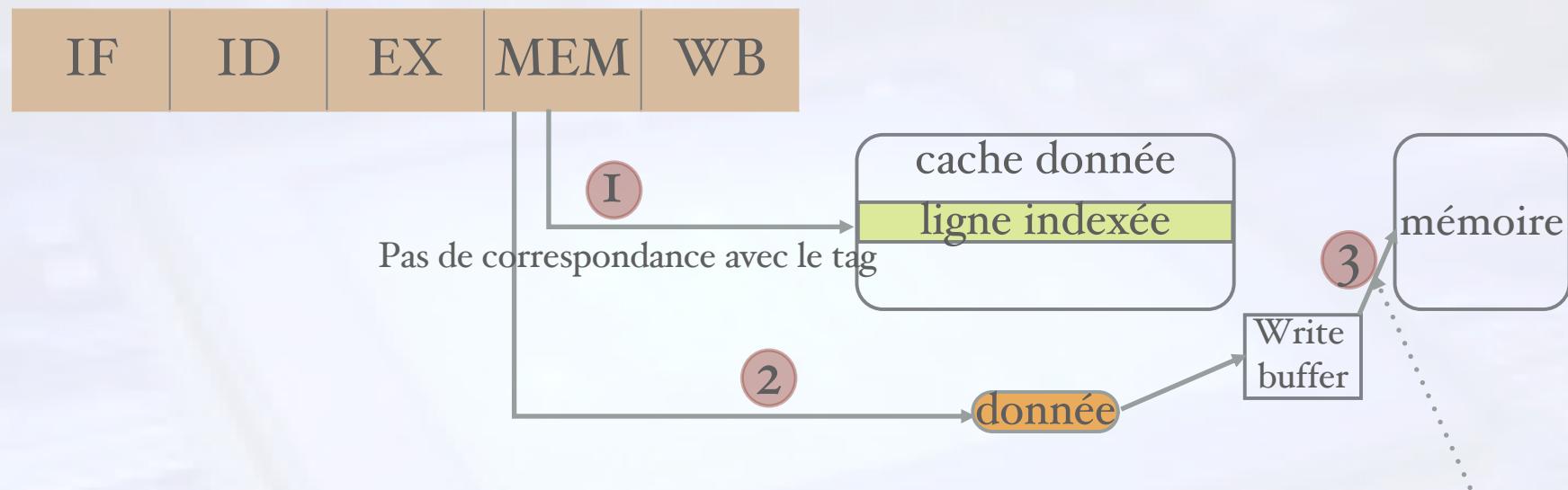


- * Cache Hit (Write-back):



Architecture µP: Hiérarchie mémoire écriture du cache donnée

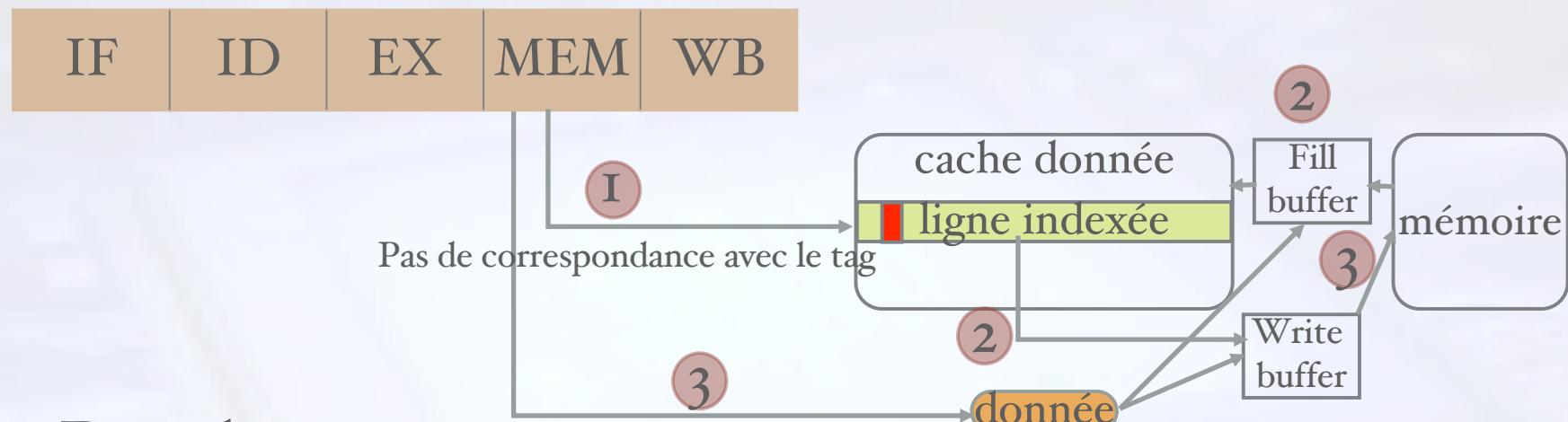
- * Cache Miss (Write-through , sans allocation):



Si l'écriture demi-mot / octet est supportée
le buffer d'écriture effectue une opération de fusion

Architecture µP: Hiérarchie mémoire écriture du cache donnée

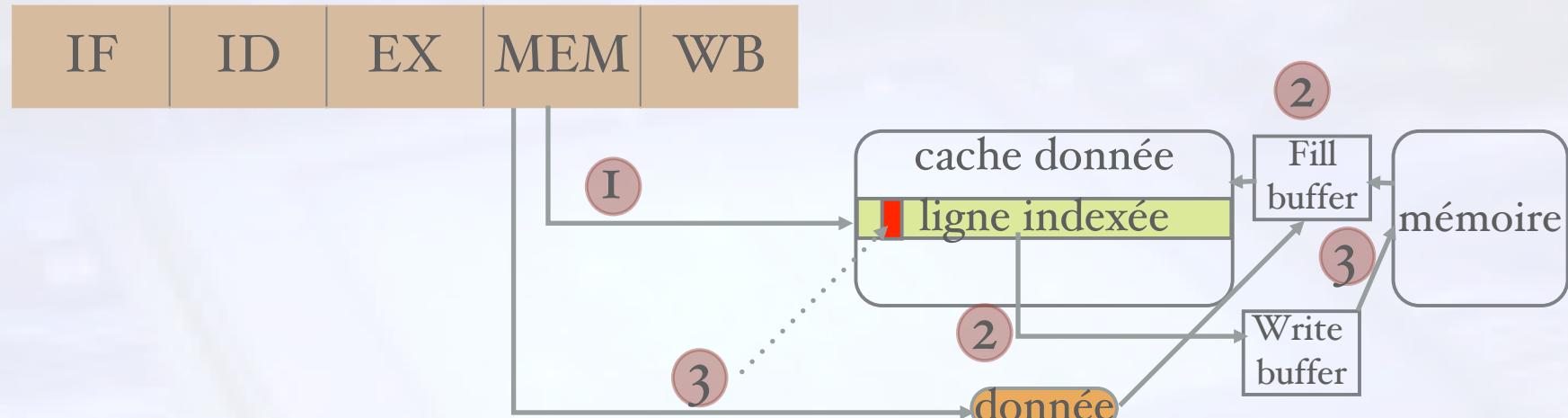
- * Cache Miss (Write-through , avec allocation):



- * Dirty bit =I :
 - * écriture des données de la ligne de cache occupée en mémoire via le «write buffer»
 - * remplissage de la ligne de cache via le « fill buffer »
- * Dirty bit =O : pas d'écriture de la ligne de cache en mémoire

Architecture µP: Hiérarchie mémoire écriture du cache donnée

- * Cache Miss (W^{rite}-back):



- * Dirty bit = 1 :
 - * écriture des données de la ligne de cache occupée en mémoire via le «write buffer»
 - * remplissage de la ligne de cache
- * Dirty bit = 0 pas d'écriture de la ligne de cache en mémoire
- * mise à 1 du dirty bit

Architecture des microprocesseurs

Unité de Gestion Mémoire

MMU

(Memory Management Unit)

Architecture µP: Unité de gestion Mémoire

Les problèmes à résoudre sur les OS modernes (multi utilisateurs , multitâches)

Fragmentation de la mémoire :

Plusieurs utilisateurs => Allocation dynamique de la mémoire

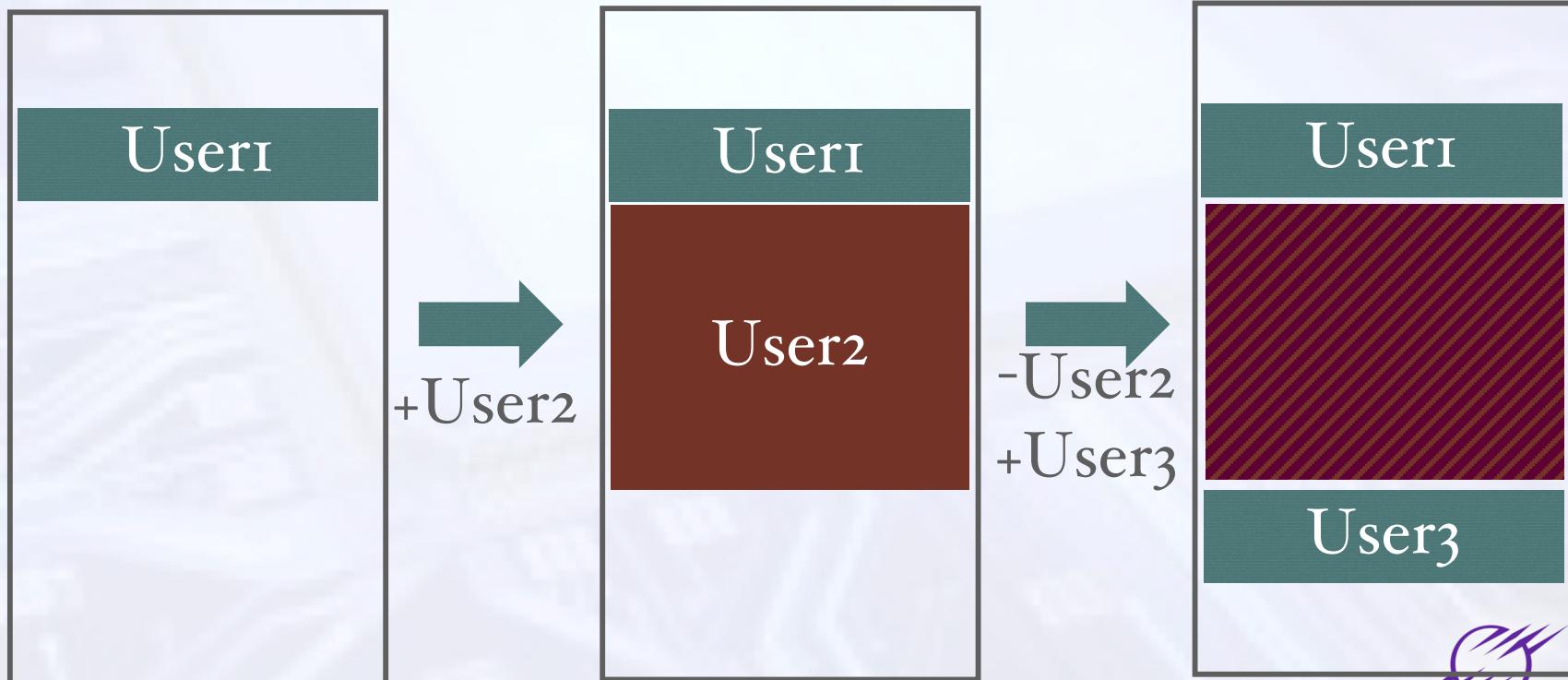
Espace d'adressage privé par Utilisateur

Espace mémoire limité :

Plusieurs utilisateurs => ↗ taille mémoire système requise

Architecture µP: Unité de gestion Mémoire

- * Fragmentation :



Architecture µP: Unité de gestion Mémoire

Espace d'adressage privé



Architecture µP: Unité de gestion Mémoire système à mémoire paginée

