Mémoires CACHE

Henri-Pierre Charles & Frédéric Rousseau
Henri-Pierre.Charles@cea.fr & Frederic.Rousseau@imag.fr

Motivations et principes de base

Motivation

- Les performances d'un microprocesseur simple ne sont pas très bonnes, en particulier les temps d'accès à des mémoires de grande taille. Par exemple, la lecture d'une information en RAM peut prendre une centaine de cycles processeurs, laissant ce dernier en attente

idée

- Utiliser une mémoire petite mais rapide pour que le processeur attende le moins possible

Difficultés

- La mémoire cache est une solution générale qui procure en *moyenne* un gain de performance significatif. Mais elle requiert une partie matérielle spécifique (et logiciel avec un compilateur adapté).

Principe de localité

Localité temporelle

- Les données ou instructions utilisées récemment le seront encore dans un futur proche (boucles, tableaux)
- Dans un programme, 90% du temps est passé dans seulement 10% des instructions

Localité spatiale

- Les données ou instructions proches en mémoire seront utilisées en même temps (tableaux, instructions séquentielles)

Par conséquent

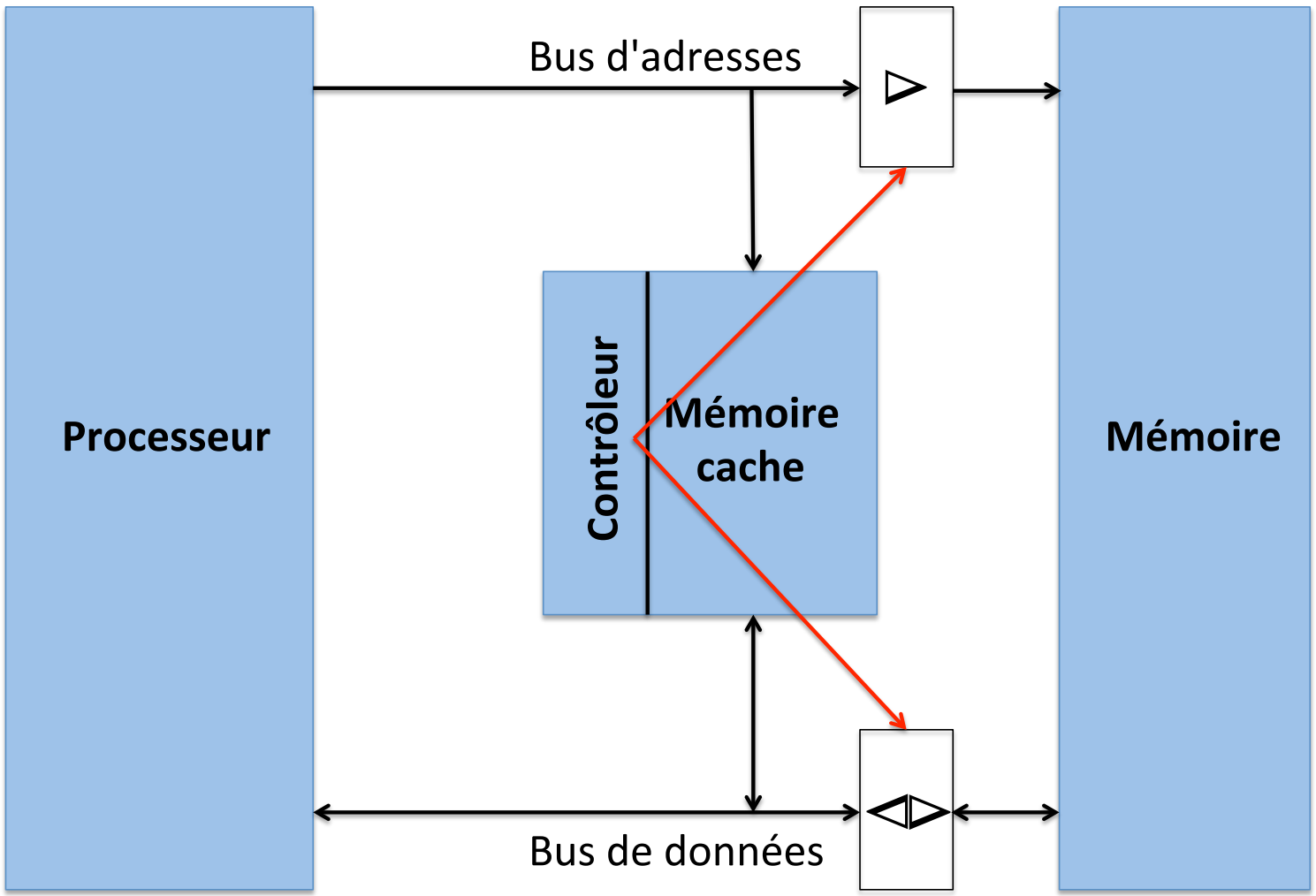
- Placer les données et les instructions en cours d'utilisation proche du processeur
- Les données ou instructions accédées le plus souvent doivent être proche du processeur

Intérêt et vocabulaire

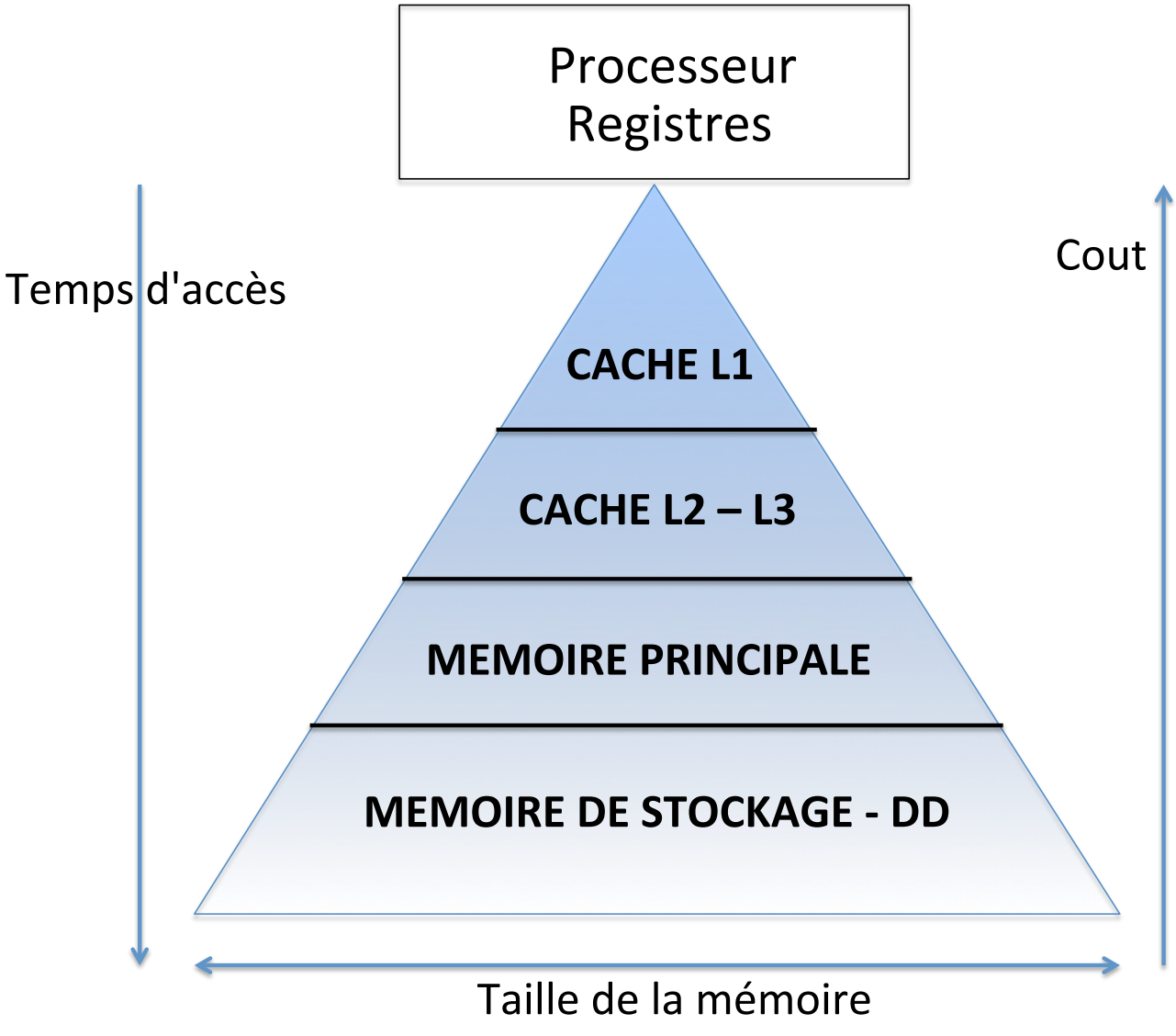
- Les caches sont utilisés pour exploiter la localités spatiale et temporelle des données (et/ou instructions)

- Succès de cache - *cache hit*
- Défaut de cache - *cache miss*
- Pénalité d'un défaut de cache - *miss penalty*
- Taux de défauts de cache - *miss rate*
- Cache d'instructions et de données - *instruction and data cache*

Principes de fonctionnement (lecture et écriture)



Hiérarchie mémoire



Performances

- Hypothèses : 1 instruction d'accès à une données nécessite 2 cycles processeur en cas de succès, 10 cycles en cas d'échec.
- En fonctionnement normal, on atteint 80% de succès.
  - Combien de cycles sont nécessaires pour exécuter 100 accès à une donnée ?
  - Réponse : (100 x 2 x 0,8) + (100 x 10 x 0,2) = 360 cycles
  - Combien de cycles le processeur attend ?
  - Réponse : 100 x (10 - 2) x 0,2 = 160 cycles (environ 44% du temps)
- Les instructions d'accès à une donnée représente 50% des instructions d'un programme.
  - Quel est le gain en terme de performance si toutes les données étaient en cache (1 instruction par cycle pour les instructions autres que load et store) ?
  - Réponse : (50 + 50 x ((2 x 0,8) + (10 x 0,2))) / (50 + 50 x 2) = 230 / 150 = 1,53

Les questions de base

4 questions sur les caches

- Ou placer une donnée (ou un bloc) dans un cache ?
- Si une donnée (ou un bloc) est présente dans un cache, comment la trouver ?
- En cas de défaut de cache, quelle donnée (bloc) doit être remplacée ?
- Que se passe t-il lors d'une écriture ?

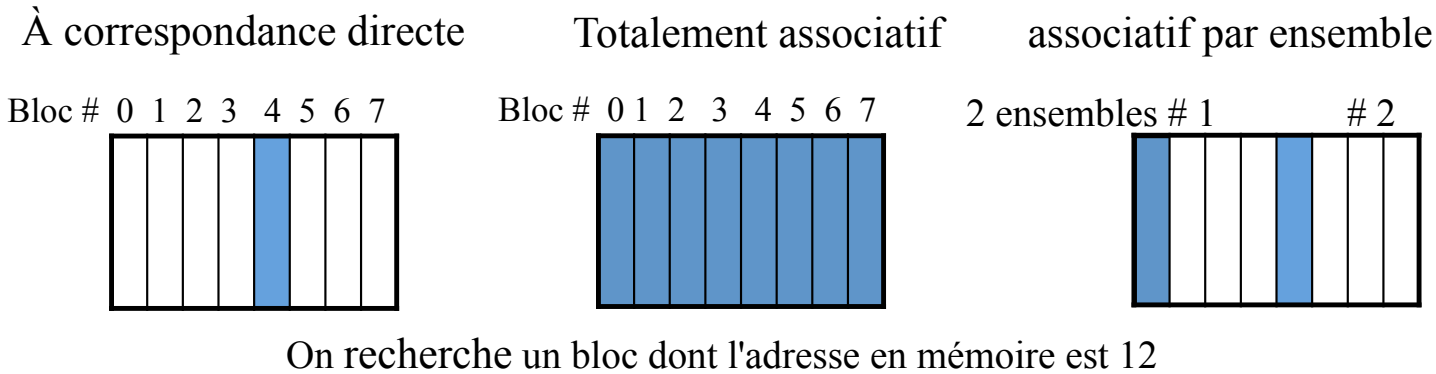
2 réponses

- Architectures des caches
- Politique d'écriture

Ou placer un nouveau bloc (donnée) dans la cache ?

Dépend de l'organisation

- Cache à correspondance directe : chaque bloc a une place unique dans le cache
- Cache totalement associatif : un bloc peut prendre n'importe quelle place
- Cache associatif par ensemble : un bloc peut prendre un ensemble restreint de places



Si un bloc est présent, comment le trouver ?

- Il existe une étiquette (*tag*) pour chaque bloc qui contient le numéro du bloc
- Toutes les étiquettes sont examinées en parallèle pour voir si l'une d'elles correspond au numéro recherché
- Enfin, pour savoir si un bloc est valide, on ajoute 1 bit de validité

Quel bloc doit être remplacé en cas de défaut ?

Dépend de l'architecture

- Cache à correspondance directe
  - Pas de choix possible
- Cache associatif par ensemble
  - Plusieurs solutions sont possibles
    - Au hasard (RR pour *Ramdom Replacement*)
    - Le moins récémment utilisé (LRU pour *Least Recently Used*)
    - Le plus vieux (LRR pour *Least Recently Replaced* ou FIFO)
    - Le moins utilisé (LFU pour *Least Frequently Used*)

Architecture des caches

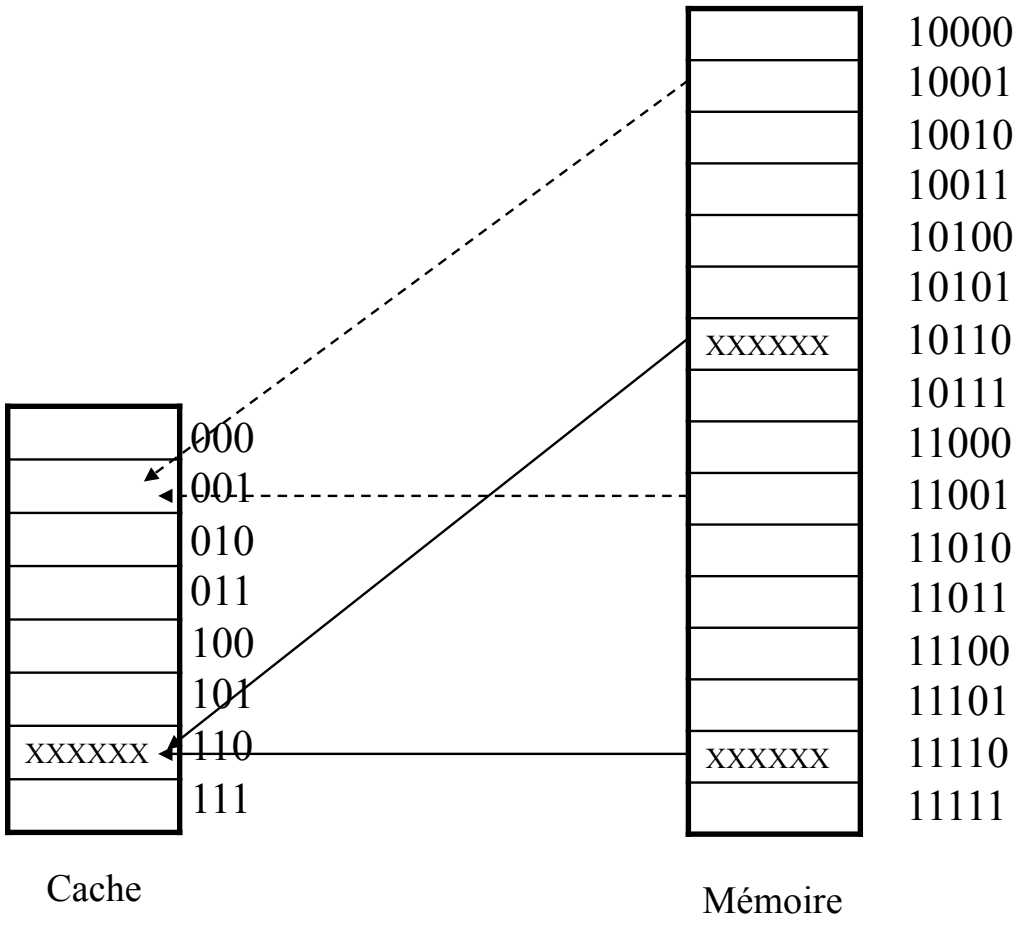
Caches à correspondance directe

- Architecture
- Exemple DECSTATION 3100

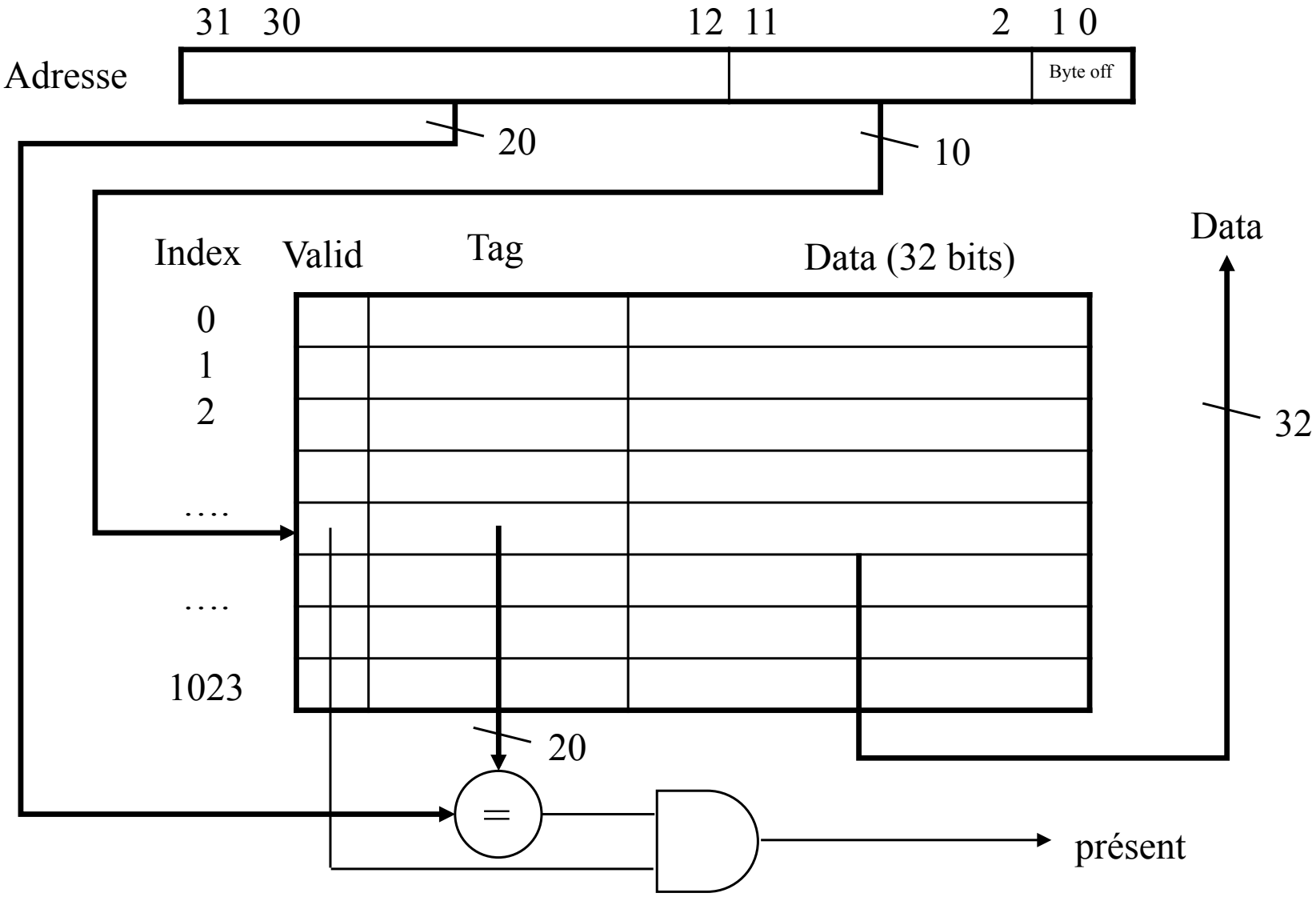
Relation adresses mémoire - adresse dans le cache

- AdresseDonneeCache = AdresseDonneeMemoire % TailleCache
- AdresseBlocCache = AdresseBlocMemoire % NbBlocCache
- Simple si la taille du cache est une puissance de 2 (modulo nb de bits)

Exemple de correspondance mémoire - cache



Architecture générale d'un cache à correspondance directe



Cache à correspondance directe : exercices

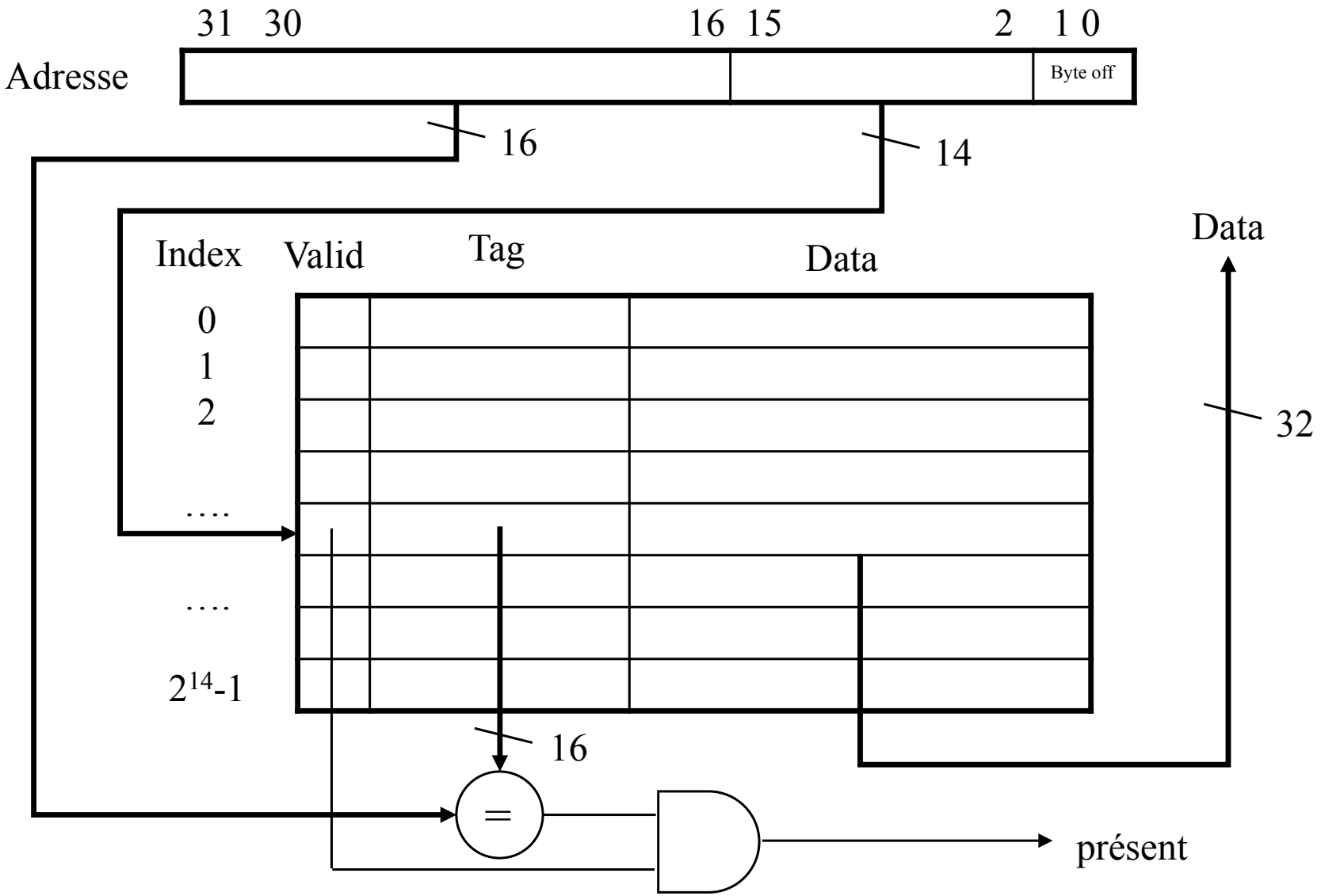
Exercice 1 : valeurs spécifiques

- Quelle est la taille (en nombre de bits) d'un cache de données (32 bits) de 64 KO à correspondance directe avec un espace d'adressage de 32 bits ?

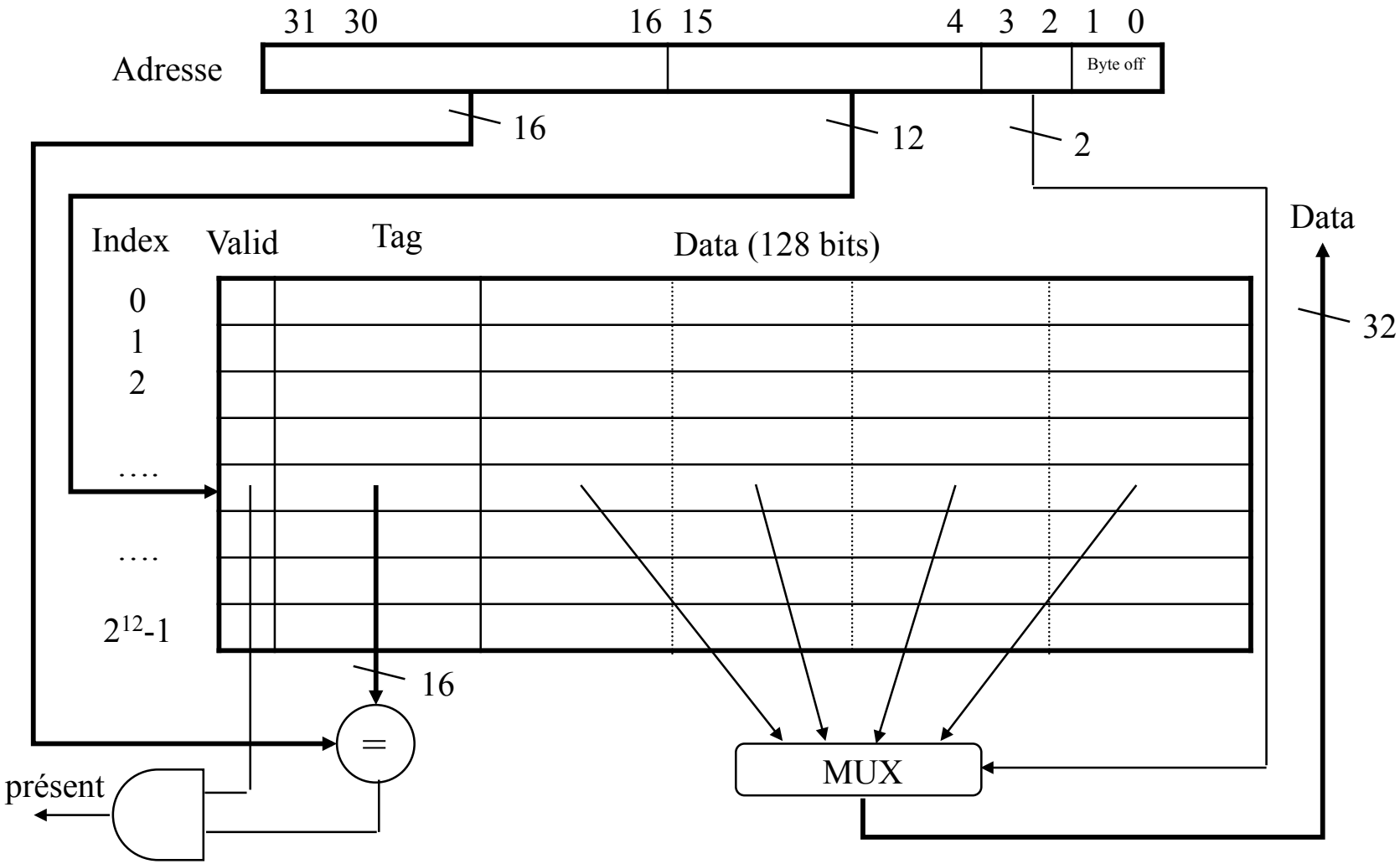
Exercice 2 : généralisation

- Même calcul dans le cas général pour un cache de taille 2^n toujours avec un espace d'adressage de 32 bits et des données de 32 bits ?

Exemple DEC Station 3100 : cache 64 KO



Cache par blocs



Caches associatifs

Objectif

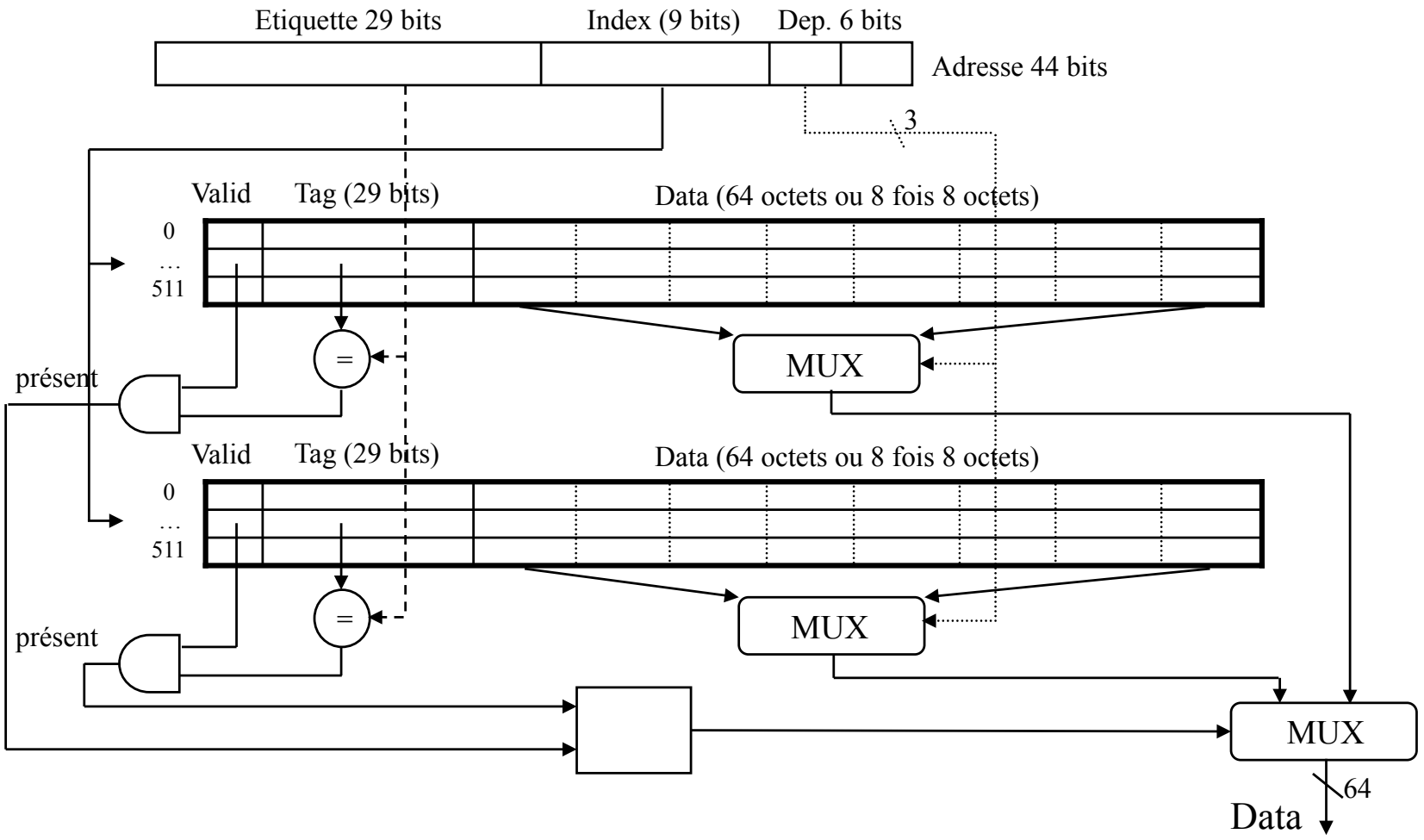
- Réduire les défauts de cache

Principe de la solution

- Un bloc mémoire peut être associé à plusieurs adresses de blocs du cache
  - A toutes les adresses de blocs du cache : "fully associatif"
  - A quelques adresses de blocs du cache : "set associatif" ou "cache associatif à n voies (n ensembles)"
  - A une seule adresse de blocs du cache : "équivalent à correspondence directe"

Exemple cache associatif - serveur COMPAQ

Exemple Processeur ALPHA 21264 : cache 64 KO associatif à 2 voies, bloc de 64 octets, mots de 64 bits, à réécriture



Que se passe t-il lors d'une écriture ?

2 politiques d'écriture

- Ecriture immédiate (ou simultannée) (*write though*) : l'information est écrite à la fois dans le cache et dans la mémoire
- Ecriture différée (*write back*) : l'information est écrite uniquement dans le cache. La mémoire ne sera mise à jour que lors d'un remplacement du bloc.

Avantages

- L'écriture immédiate est facile à implémenter. Les échecs en lecture ne provoquent jamais d'écriture. Meilleure cohérence de données.
- Avec l'écriture différée, plusieurs écritures ne nécessitent qu'une seule écriture en méoire. Solution qui consomme moins.

Performances des caches

Critère d'évaluation

- La pénalité d'échec correspond au temps pour remplacer un bloc dans le cache
- Le taux d'échec est la fraction des accès cache qui provoque un échec

Comment améliorer la performance des caches

- Réduire la pénalité d'échec : cache multi-niveaux, ...
- Réduire le temps des accès réussis : cache plus petit, pas de traduction d'adresses, accès pipeliné, ...
- *Réduire le taux d'échecs*

Réduire le taux d'échecs

3 catégories d'échecs

- Obligatoires : échec de première référence
- Capacités : Quand le cache ne peut pas contenir tous les blocs utilisés
- Conflit : échec de collision dans les caches à correspondance directe ou associatifs par ensemble.

Améliorations

- Taille de blocs plus grande
- Caches plus gros
- Associativité plus élevée
- Optimisation du compilateur

Amélioration des performances par le compilateur

Echange des boucles : améliorer la localité spatiale

- Hypothèses
  - Les données d'un tableau sont généralement placées à des adresses consécutives en mémoire
  - Un bloc correspond à plusieurs données

Avant

```
for (j = 0 ; j < 100 ; j++)
for (i = 0 ; i < 5000 ; i++)
x[i][j] = 3 * x[i][j] ;
```

Après

```
for (i = 0 ; i < 5000 ; i++)
for (j = 0 ; j < 100 ; j++)
x[i][j] = 3 * x[i][j] ;
```

Explication

- Si les valeurs x[a][n], x[a][n+1], x[a][n+2], ... sont à des adresses consécutives, un chargement par blocs permet de placer dans le cache toutes ces valeurs en une seule requête.

Comparaison Pentium Pro et PowerPC - années 2000

	Intel Pentium Pro	**Power PC 604**
Utilisation	PC – serveurs Dell, DEC, HP, …	Apple Macintosh IBM RS6000
Niveaux de cache	2	2
Cache L1	Cache d'instructions Cache de données	Cache d'instructions Cache de données
Taille du cache de données	8 KO	16 KO
Associativité	4 groupes	4 groupes
Remplacement	LRU approximatif	LRU
Taille du bloc	32 octets	32 octets
Ecriture	Réécriture	Réécriture & simultanée
Taille du cache L2	256 ou 512 KO	256 ou 512 KO

Cache configurable du processeur ARM

Aujourd'hui les caches sont configurables

- A l'aide de registres dédiés (ARM) : taille, taille ligne, politique d'écriture, validation, ...
- Ex : ARM Cortex A-57 : L1 32 KO, associatif à 4 ensembles, 64 octets par ligne (16 mots)

Table 8-9 Instruction cache data RAM sizes, no parity or ECC

Cache size	Data RAMs
4KB, 4 1KB ways	4 banks 64 bits 128 lines or 8 banks 32 bits 128 lines
8KB, 4 2KB ways	4 banks 64 bits 256 lines or 8 banks 32 bits 256 lines
16KB, 4 4KB ways	4 banks 64 bits 512 lines or 8 banks 32 bits 512 lines
32KB, 4 8KB ways	4 banks 64 bits 1024 lines or 8 banks 32 bits 1024 lines
64KB, 4 16KB ways	4 banks 64 bits 2048 lines or 8 banks 32 bits 2048 lines

Table 8-10 Data cache data RAM sizes, no parity or ECC

Cache size	Data RAMs
4KB, 4 1KB ways	8 banks 32 bits 128 lines
8KB, 4 2KB ways	8 banks 32 bits 256 lines

Mémoire cache (de données) du processeur UltraSPARC II

Le processeur UltraSPARC II de chez SUN Microsystems est un processeur RISC 64 bits (8 octets). Les données manipulées ont une taille de 64 bits. Les adresses sont sur 44 bits.

• Le cache de données est un cache de 16 KOctets à correspondance directe. Chaque bloc est composé de 32 octets.

Cache de données - Questions

- Indiquer la taille totale du cache de données
- Dessiner l'architecture du cache de données en précisant tous les paramètres de l'architecture (nombre de lignes, taille des bus, . . .).
- On charge dans ce cache le bloc de données (32 octets) dont l'adresse mémoire de la première donnée du bloc est 0x000AB240020. Indiquer la ligne du cache qui reçoit ce bloc et le contenu du champ index de la ligne du cache.