CPU ET MÉMOIRE

Design, améliorations, et implémentations

Chapitre 8.2

CPU et Mémoire
Design, améliorations, et implémentations

- Techniques et caractéristiques modernes qui permettent de donner aux ordinateurs actuels toute leur puissance

Architectures CPU – Instruction Set Architecture (ISA)

- Architecture de CPU = caractéristiques de processeur
  - *Nombre et types des registres, modes d'adressage, nombre et types d'instructions, etc.*
- Design de l'architecture CPU
  - *Architecture traditionnelle (CISC & RISC)*
  - *VLIW (Transmeta) – Very Long Instruction Word*
  - *EPIC (Intel) – Explicitly Parallel Instruction Computer*
- Architectures de CPU
  - *IBM System/360*
  - *Famille Intel x86*
  - *Famille IBM POWER/PowerPC*
  - *Famille Sun SPARC*

Architectures traditionnelles

- Architecture CISC & RISC
  - *Architecture traditionnelle des microprocesseurs se composent de deux grandes familles:*
    - CISC – Complex Instruction Set Computer
    - RISC – Reduced Instruction Set Computer
- Chacune de ces deux architectures est consistante avec les caractéristiques d'une architecture selon Von Neumann

Microprocesseur, l'architecture CISC (famille Intel)

- Motivation
- La mémoire travaillait très lentement => soumettre au microprocesseur des instructions complexes qui demanderaient autant d'accès mémoire que plusieurs petites instructions
- Le développement des langages de haut niveau posa de nombreux problèmes quand à la conception de compilateurs. On a donc eu tendance à incorporer au niveau processeur des instructions plus proches de la structure de ces langages.

Microprocesseur, l’architecture CISC (famille Intel)

- Grand nombre d’instructions ou le microprocesseur doit exécuter des taches complexes par instruction unique.
- Pour une tache donnée, une machine CISC exécute ainsi un petit nombre d’instructions mais chacune nécessite un plus grand nombre de cycles d’horloge.
- Le code machine de ces instructions varie d’une instruction à l’autre et nécessite un décodeur complexe (microcode)

Microprocesseur, l'architecture RISC (Power PC, Sun Sparc, Motorola 68000)

- Motivation

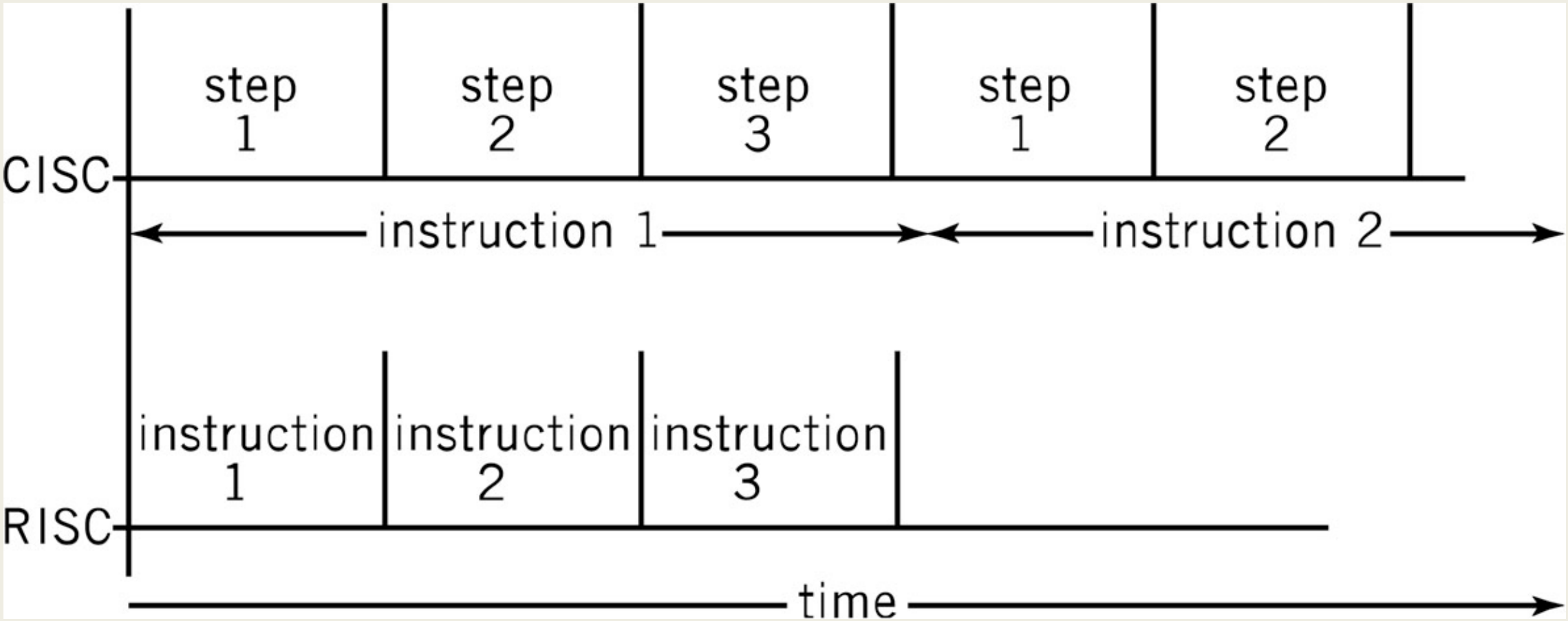
Statistique: 80% des traitements des langages de haut niveau faisaient appel a seulement 20% des instructions du microprocesseur

=> Réduire le jeu d'instructions à celles le plus couramment utilisées et d'en améliorer la vitesse de traitement

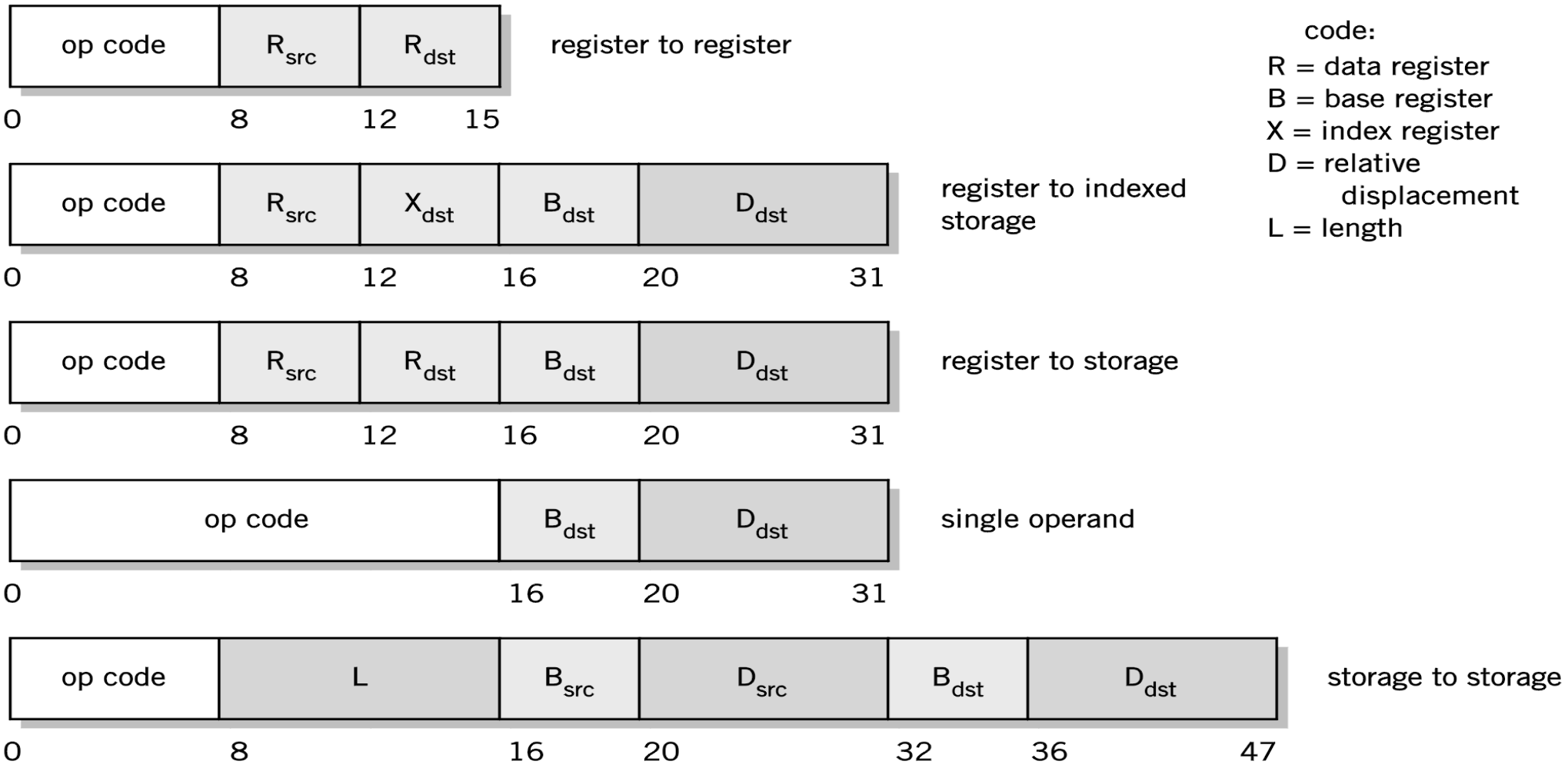
Microprocesseur, l’architecture RISC (Power PC, Sun Sparc, Motorola 68000)

- Les instructions sont en nombre réduit => une diminution de la complexité de la partie unité de commande
- Une implantation d’instructions de longueurs fixes
- Chacune de ces instructions s’exécutent ainsi en un cycle d’horloge

Exécution CISC vs RISC

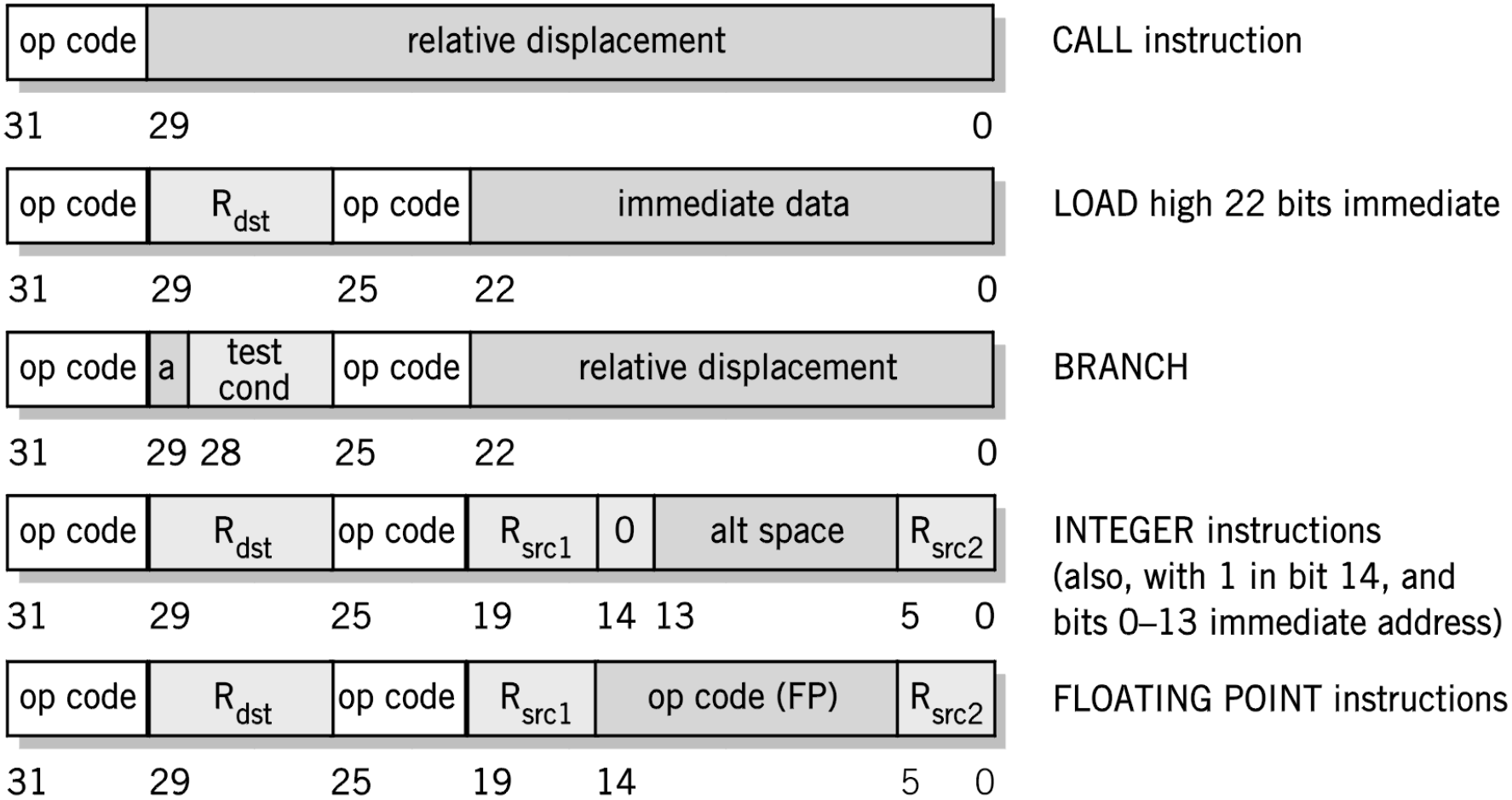


Formats d'instructions: CISC



IBM mainframe formats (partial set)

Formats d'instructions: RISC



SPARC RISC formats (complete set)

RISC contre CISC

CISC	RISC
Compilateur simple	Compilateur complexe
Beaucoup de modes d'adressage	Peu de modes d'adressage
Toutes les instructions sont susceptibles d'accéder à la mémoire	Seules les instructions LOAD et STORE ont accès à la mémoire
Peu de registres	Beaucoup de registres
Décodeur complexe (microcode)	Décodeur simple (câblé)
Instructions au format variable	Instructions au format fixe
Instructions complexes prenant plusieurs cycles	Instructions simples ne prenant qu'un seul cycle

Architectures VLIW - EPIC

- VLIW – Very Long Instruction Word
- EPIC – Explicitely Parallel Instruction Computer
- Le but de ces deux types d'architecture est d'augmenter la vitesse d'exécution du processeur en traitant des instructions/opérations en parallèle

Architecture VLIW

- CPU Transmeta Crusoe
  - *Caractéristiques:*
    - Molécule de longueur 128 bits
      - *Divisée en 4 atomes de 32-bits (atome = instruction)*
      - *4 instructions pouvant être exécutées simultanément*
    - 64 registres d'usage général
    - Logiciel de Codage Morphing
      - *Traduction du code machine d'autres CPUs en molécules*

Architecture EPIC

- CPU Intel Itanium - Même but mais avec caractéristiques différentes
- Paquet de longueur 128-bits
  - *Comprend 3 instructions de 41-bits*
  - *Comprend 5 bits pour identifier le type d'instruction*
- 128 registres d'usage général de 64-bits
- 128 registres flottants de 82-bits
- Instructions famille Intel X86
- Les 5 bits sont des bits d'informations qui permettent d'identifier les dépendance potentielles entre exécutions

Architectures VLIW vs EPIC

- L’ordonnancement des opérations (+ gestion des priorités, dépendance, etc.) dans l’instruction de 128 bits est Intégré dans l’architecture matériel pour le VLIW
- Géré par le programmeur ou le compilateur (logiciel) - EPIC

Amélioration des performances

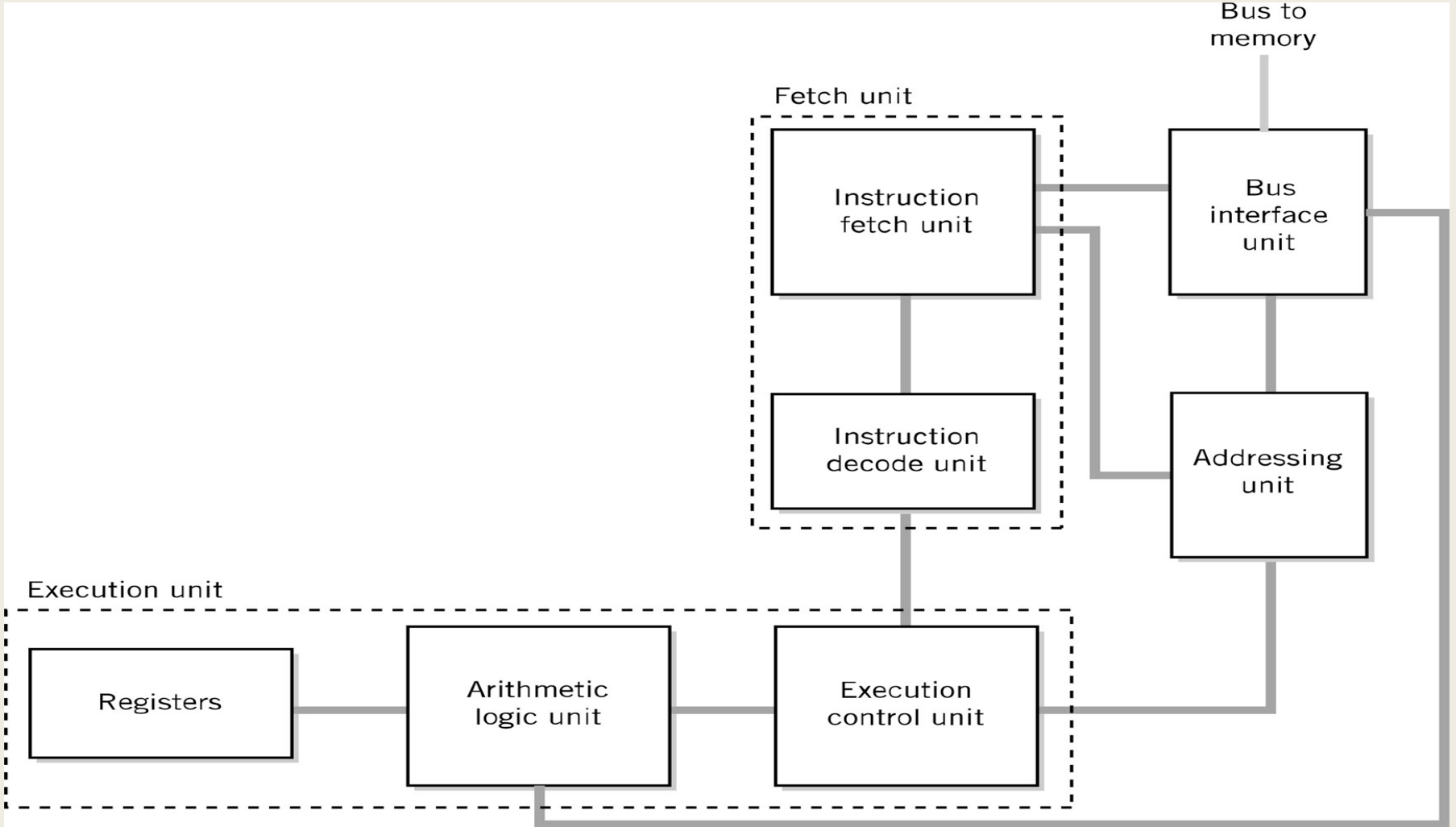
- Augmenter la vitesse des microprocesseurs
  - *L'augmentation de la fréquence d'horloge*
- Travailler sur l'architecture interne du microprocesseur
- Améliorer les performances du bus de communication entre processeur et mémoire centrale
- Parallélisme de certains opérations; faire plusieurs choses en même temps
  - *Le parallélisme au niveau des instructions*
  - *Le parallélisme au niveau des processeurs*

Conception des ordinateurs modernes

- Unités Fetch et Execute séparées
- Technique de pipeline
- Unités d’exécutions parallèles
- Traitement Scalaire
- Traitement Superscalaire
- Traitement d’instructions de branchement

Conception des ordinateurs modernes

- Unités Fetch et Execute séparées



Unités Fetch et Execute séparées

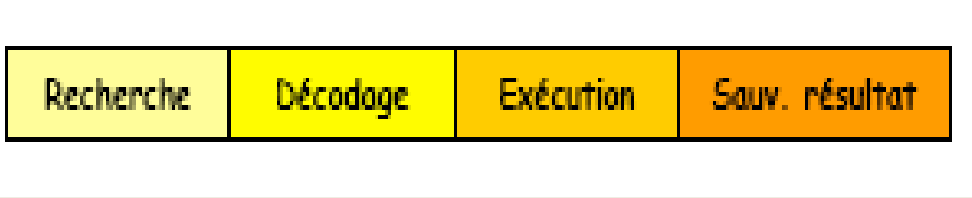
- Unité Fetch
  - *Unité de recherche d'instruction*
  - *Unité de décodage*
    - Détermine le code d'opération
    - Identifie le type d'instruction et les opérandes
- Plusieurs instructions sont recherchées en parallèle et placées dans le tampon
- Unité Execute
- Reçoit les instructions de l'unité de décodage
- Unité d'exécution appropriée serve l'instruction

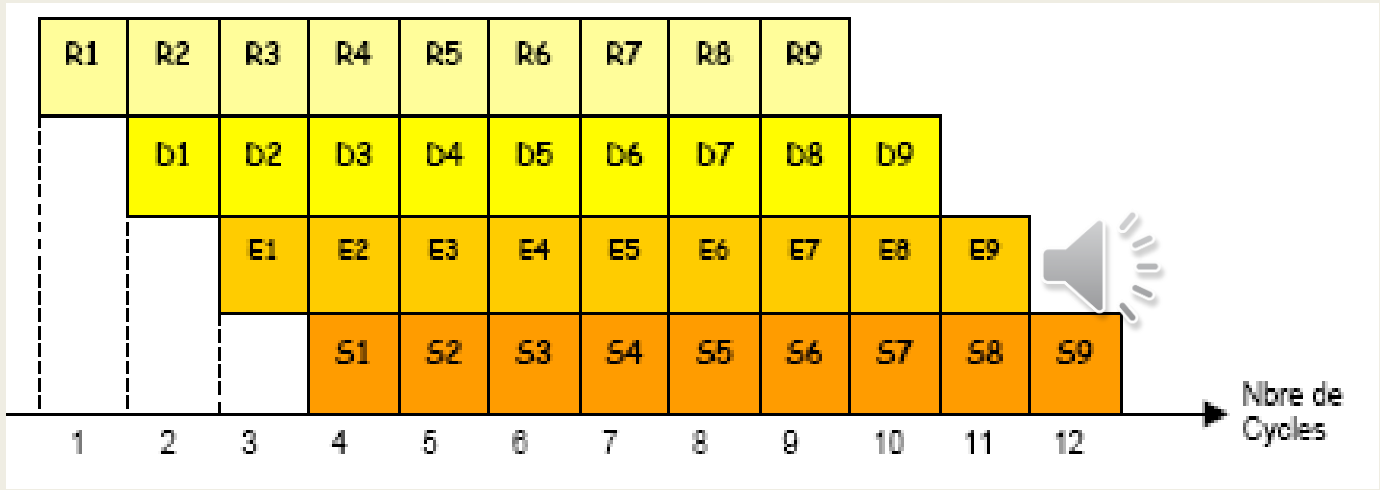
Technique de pipeline

- Le fonctionnement d'un microprocesseur simple n'est pas efficace
- Idée inspirée de l'organisation du travail à la chaîne
- L'exécution d'une instruction peut être décomposée en plusieurs phases qui s'exécutent indépendamment les unes des autres si l'on dispose d'unités fonctionnelles (du matériel) le permettant
- La séquence d'instructions fetch/execute est exécutée comme si celle-ci était dans une chaîne de montage.

Technique de pipeline

- Exemple de l'exécution en 4 phases d'une instruction

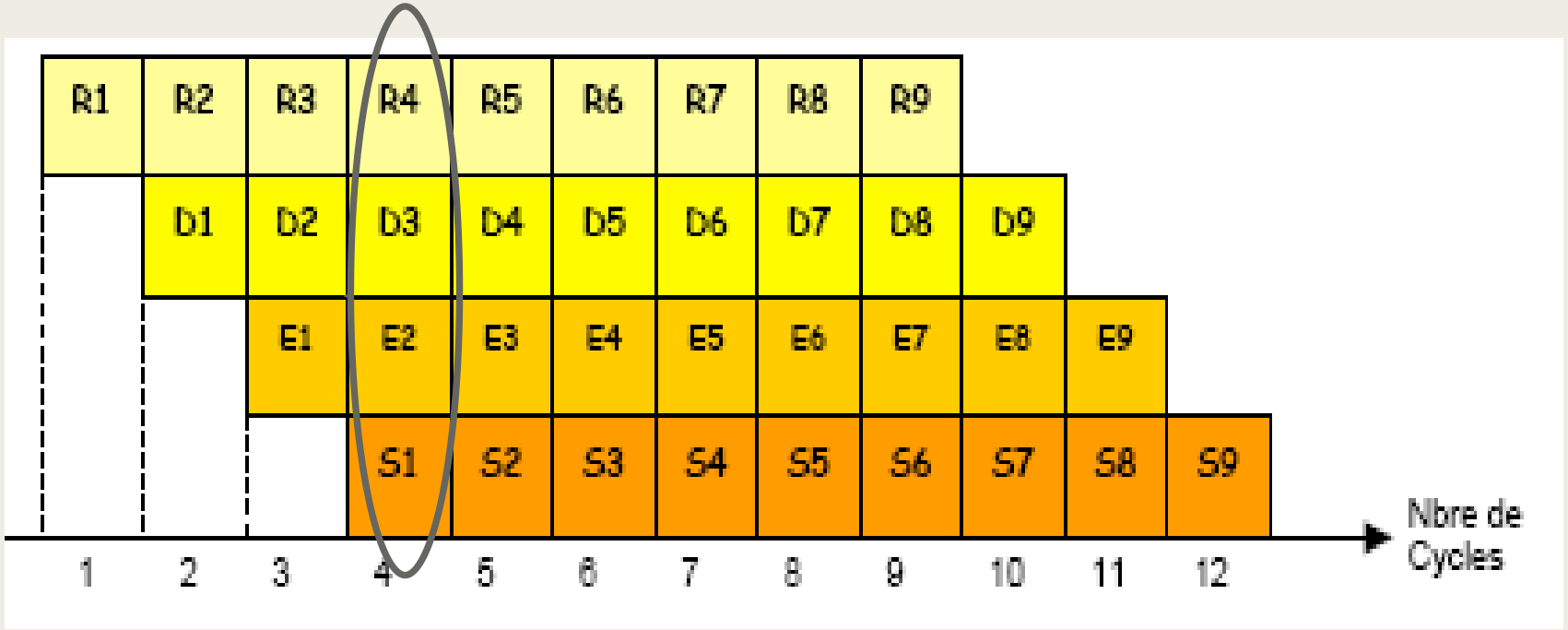




- Le temps d'exécution d'une instruction n'est pas réduit mais le débit d'exécution des instructions est considérablement augmenté

Technique de pipeline

- Si la machine débute l'exécution d'une instruction à chaque cycle et le pipeline est pleinement occupé à partir du quatrième cycle le gain obtenu dépend donc du nombre d'étages du pipeline

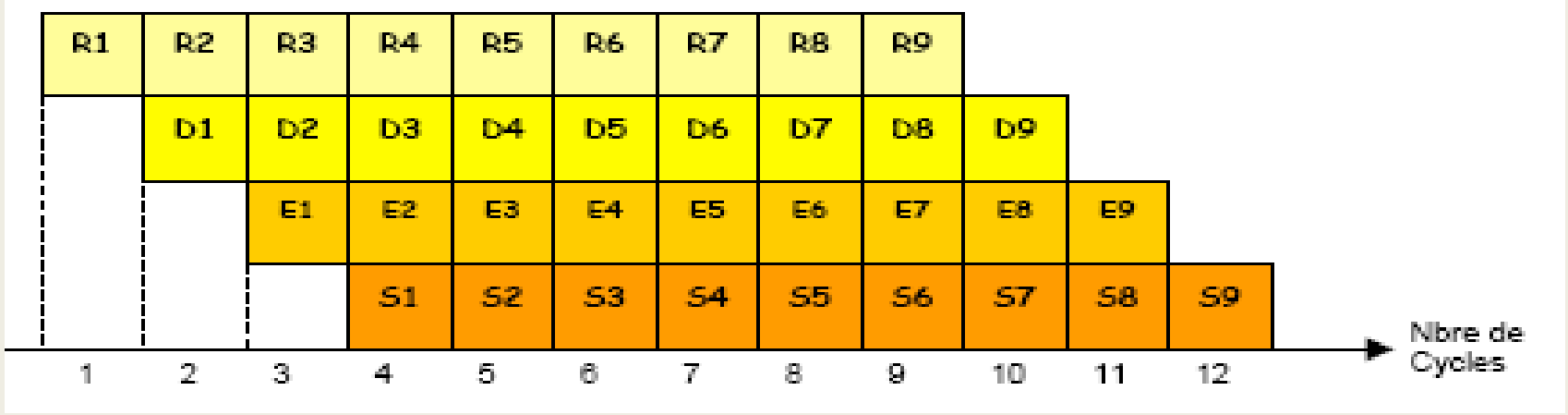


Technique de pipeline

Pour exécuter **n** instructions, en supposant que chaque instruction s'exécute en **k** cycles d'horloge, il faut :

- ***n * k*** *cycles d'horloge pour une exécution séquentielle*
- ***k*** *cycles d'horloge pour exécuter la première instruction puis* ***n-1*** *cycles pour les* ***n-1*** *instructions suivantes si on utilise un pipeline de* ***k*** *étages*

- Gain: G = (n*k)/(k+n-1)



Technique de pipeline

- Le temps de traitement dans chaque unité doit être a peu près égal sinon les unités rapides doivent attendre les unités lentes

- L'Athlon d'AMD comprend un pipeline de 11 étages

- Les Pentium 2, 3 et 4 d'Intel comprennent respectivement des pipelines de 12, 10 et 20 étages

Technique de pipeline

- Le bon fonctionnement du pipeline peut être perturbé par plusieurs événements appelés aléas (pipeline hazard)
- Trois categories
  - *Aléas structurels*
  - *Aléas de données*
  - *Aléas de contrôle (les instructions de branchement)*
    - Les instructions de sauts inconditionnels et de branchement
- Malgré ces circonstances la technique de pipeline reste très efficace

Technique de pipeline

- Aléas structurels
  - *survient lorsque deux instructions dans des étages différents du pipeline nécessitent la même ressource*
- Aléas de données
  - *Survient lorsqu'une instruction nécessite une donnée qui n'a pas encore été calculée par une instruction précédente*
    - Provient du fait que les instructions lisent leurs arguments dans les premiers étages du pipeline alors qu'elles produisent leur résultat dans les derniers étages
  - *Solution*
    - Réarrangement de la séquence d'instructions pour maintenir le pipeline plein

Technique de pipeline

- Aléas de contrôle
  - *Survient dès qu'une instruction de branchement est exécutée*
    - Si le branchement est effectué, les instructions qui suivent dans le pipeline ne doivent pas être exécutée
- Solutions
  - *Pipelines séparés pour les deux possibilités*
  - *Prédiction basée sur les branchements effectués à l'exécution précédente*

Plusieurs unités d’exécutions

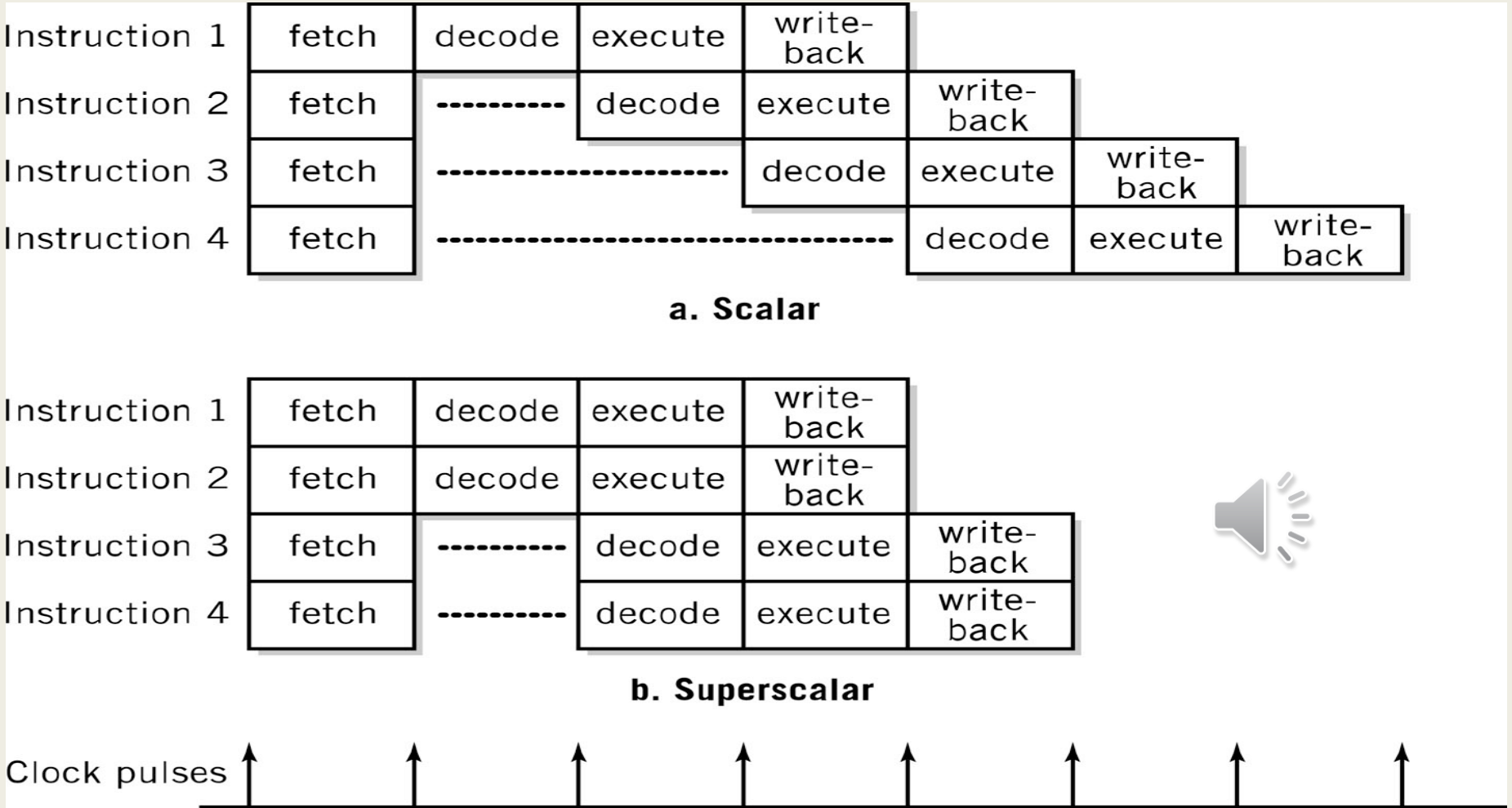
- Différentes instructions ont différent nombre des étapes dans leur cycle
- Différences dans chaque étape
- Chaque unité d’exécutions est optimisée pour un type général d’instruction
- Plusieurs opérations sont traitées à la fois

Architecture super scalaire

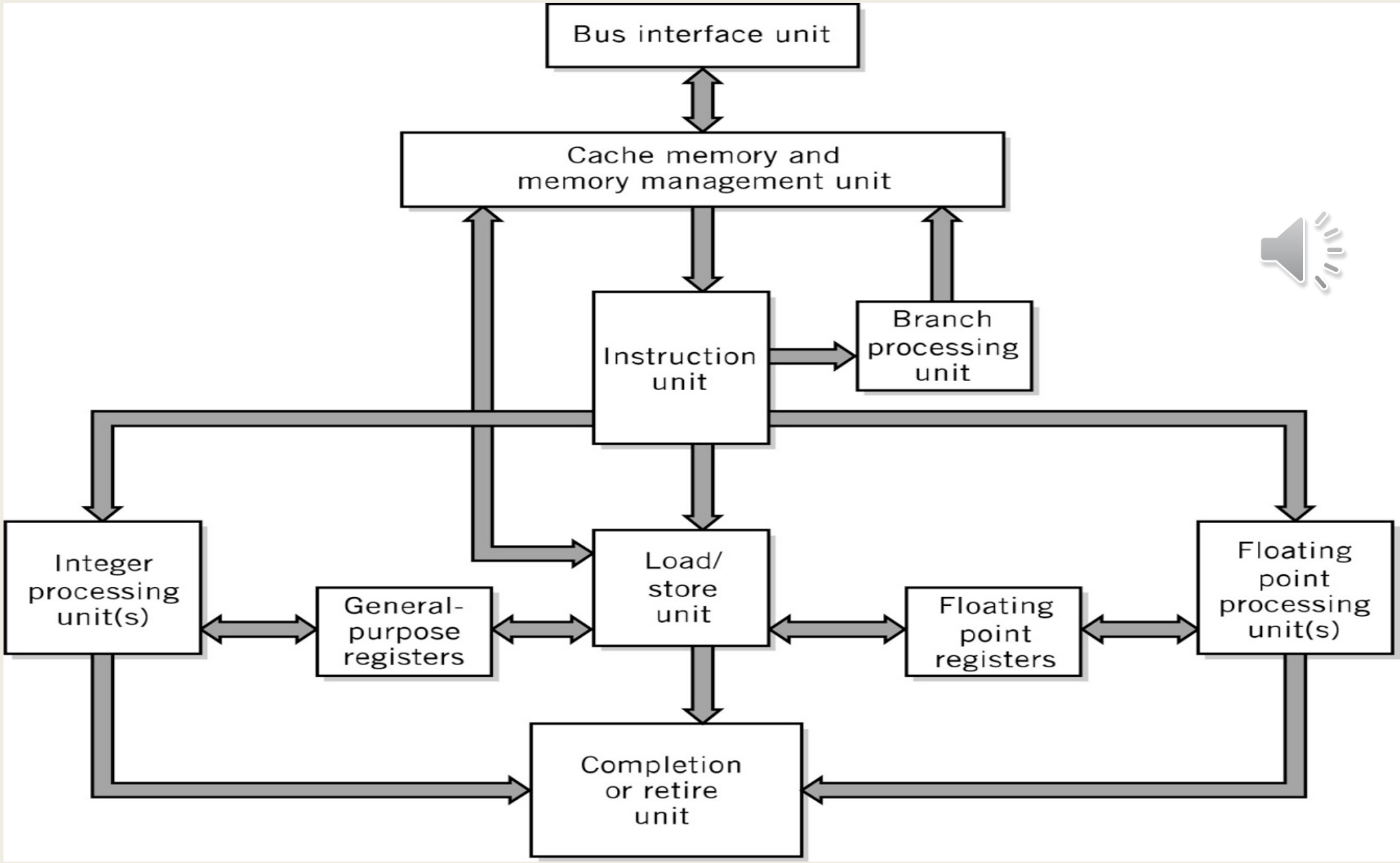
- Exécute plus qu'une instruction par un cycle d'horloge
  - *Séparer cycles d'extraction de l'instruction et d'exécution*
  - *Garder les données de phases Extraction et Décodage*
  - *Disposer dans le pipeline de plusieurs unités d'exécutions*
- Exemples: Pentium et Atom d'Intel, les 21064 et 21164 de Digital, le Cortex A8 d'ARM et le cœur Power 6 d'IBM

Scalaire vs. Super scalaire

- Processeur Scalaire - processeur pour lequel la vitesse d'exécution moyenne d'une instruction égale approximativement à la vitesse d'un cycle d'horloge



Bloc Diagramme du CPU super scalaire



Problèmes de traitement, Super scalaire

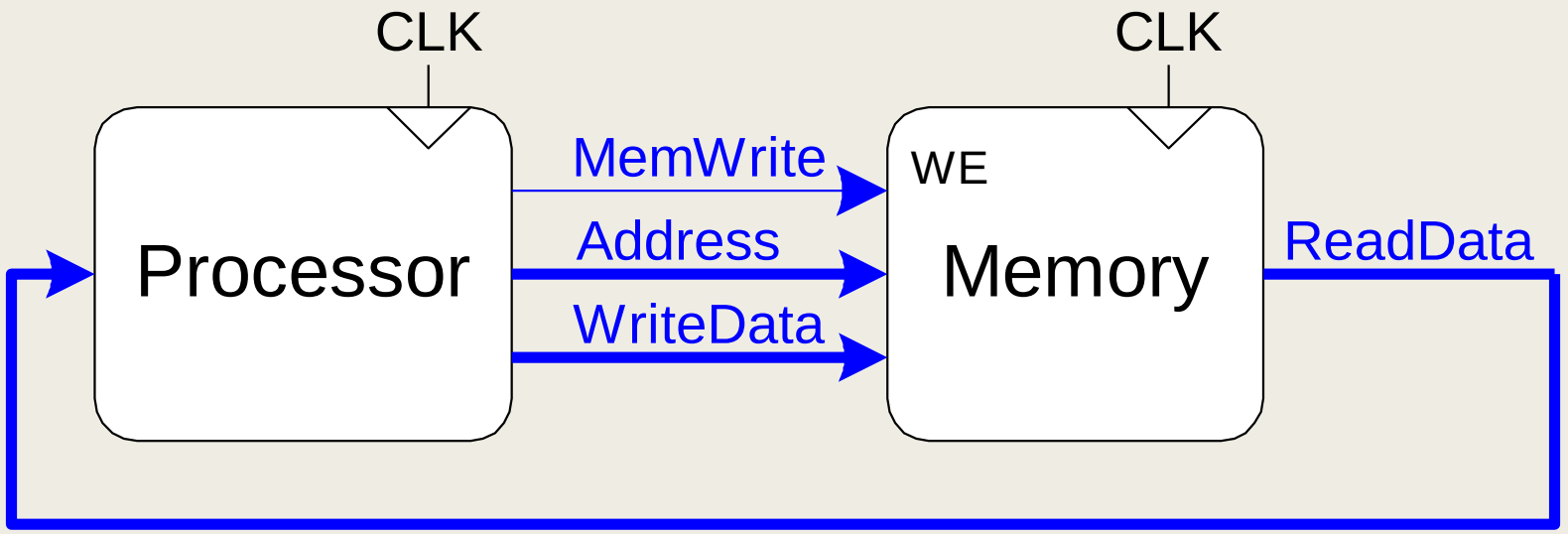
- Traitement “Out-of-order” – aléas (hazards)
  - *Aléas de données*
  - *Aléas de contrôle*
    - Exécutions spéculatives parallèles ou prédiction de branchement
    - Table historique de branchement
- Conflit d’accès aux registres
  - *Renommer ou utiliser les registres logiques*

Introduction

Comment améliorer les performances ?

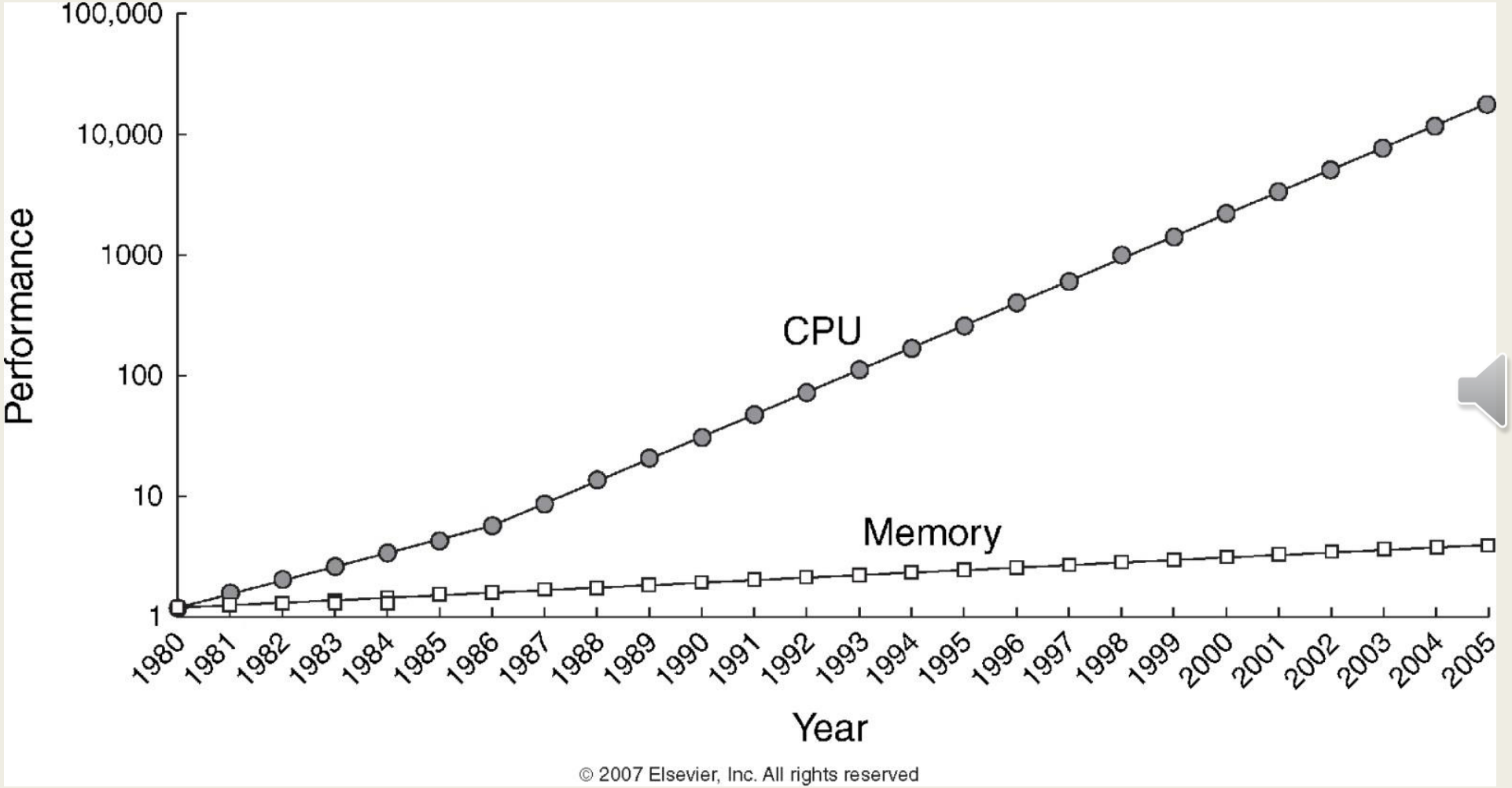
- Performances globales sont définies par:
  - Performance d'un processeur
  - Performance de système de mémoire

Interface de mémoire



Mémoires

- Mémoire principale ne peut pas être accédée en 1 cycle d'horloge depuis 1980



Défis du système de mémoire

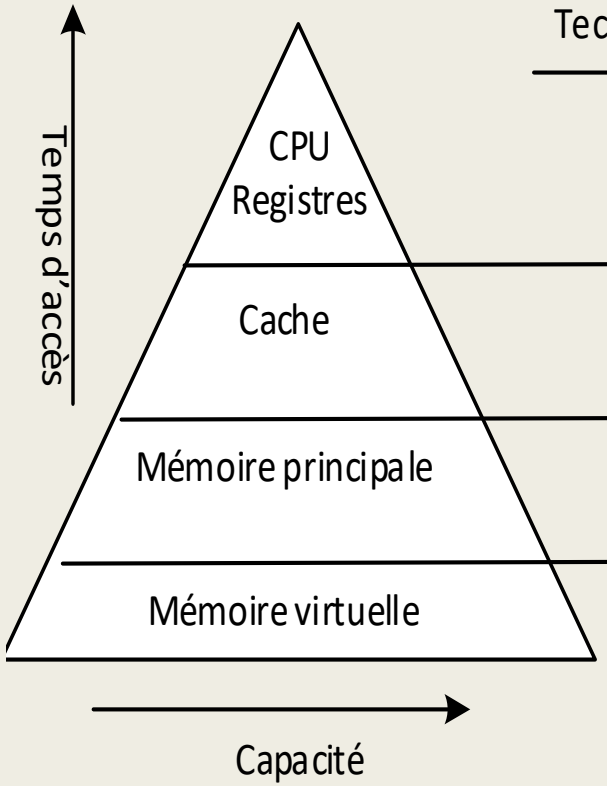
- Faire le système de mémoire rapide comme un processeur
- Solution:
  - Utiliser une hiérarchie des mémoires
- Mémoire idéale:
  - Rapide
  - Pas chère
  - Grande (capacité)

Mais, on peut choisir seulement les deux caractéristiques!

Mémoires

- Les éléments de mémoire d'un ordinateur se répartissent en plusieurs niveaux caractérisés par
  - *Leurs capacités*
  - *Leurs temps d'accès*
- Hiérarchie
  - *Les niveaux sont ordonnés en fonction*
    - Temps d'accès
    - Capacité et coût par bit

Hiérarchie des mémoires



Technologie	Coût/GB	Temps d'accès
Bascules		~ 0.25 ns
SRAM	~ $10,000	~ 1 – 10 ns
DRAM	~ $100	~ 10-50 ns
Disque dur	~ $1	~ 10,000,000 ns

Mémoire centrale

- Mémoire centrale ou principale contient les instructions et les données des programmes que l'on désire exécuter, ainsi qu'une partie du système d'exploitation nécessaire au bon fonctionnement de l'ordinateur
  - *Depuis le début des années 70, les mémoires à semi-conducteurs constituent les éléments de base de toute mémoire centrale*

Mémoires à semi-conducteurs

- RAM - Mémoire à accès aléatoire, mémoire volatile
  - *Le temps d'accès est indépendant du numéro de la cellule adressée*
- On distingue différents types de mémoires RAM
  - *DRAM – « Dynamic RAM », mémoire vive dynamique*
  - *SRAM – « Static RAM », mémoire vive statique*

DRAM

- Pas chères, consommation électrique fiable, grande densité d'intégration
- Les boitiers de mémoire dynamique enferment une pastille de silicium sur laquelle est intégré un très grand nombre de cellules binaires.
- Chaque cellule binaire est réalisée à partir d'un transistor relié à un petit condensateur
- L'inconvénient de cette technique simple est que le condensateur se décharge seul au cours du temps (courants de fuite). Il est donc nécessaire de rafraichir tous les condensateurs

SRAM

- Les mémoires statiques n'utilisent pas de condensateurs : chaque cellule binaire est réalisée à l'aide de 4 transistors formant un *bistable*
- SRAM permettent des temps d'accès plus court que les DRAM, mais sont plus couteuses car leur construction demande 4 fois plus de transistors que les DRAM.
- Les SRAM sont utilisées lorsque l'on désire maximiser les performances
  - *Mémoires caches*

Mémoires non volatiles, ROM

- *« Read-only Memory » - Mémoire à lecture seul*
  - *EEPROM - « Electrically Erasable Programmable ROM » - ROM programmable et Électriquement Effaçable*
- *Mémoire Flash*
  - *Basée sur le principe des EEPROM*
  - *Plus performantes que les disques mais chères*
  - *Se programme électriquement par blocs*

Amélioration reposant sur l’accès mémoire

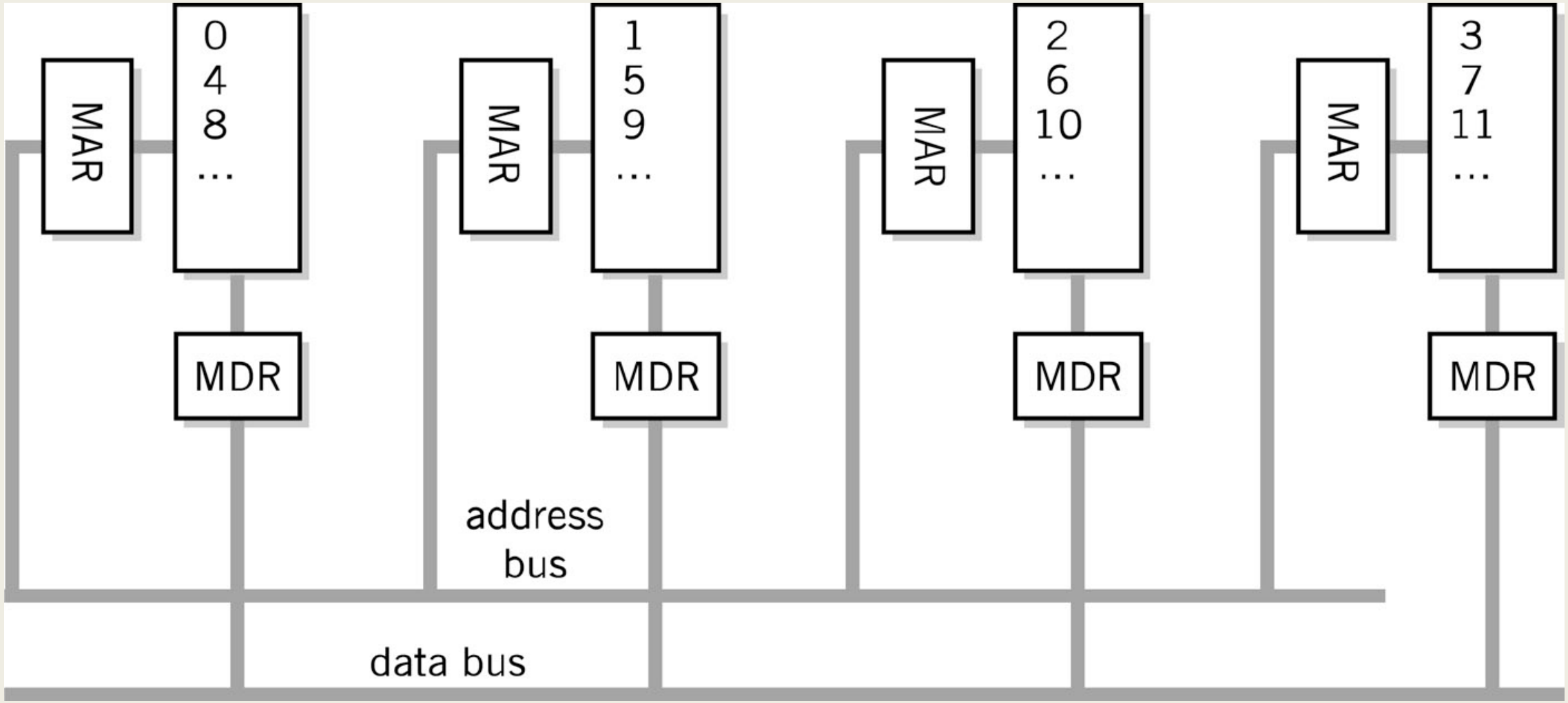
- Les échanges entre le processeur et la mémoire sont très nombreux
  - *Un programme et ses données doivent êtres placés en mémoire centrale afin d’être exécutés par le processeur*

Amélioration reposant sur l'accès mémoire

- Un accès mémoire est lent comparativement à la vitesse de processeur
  - *CPU 2Ghz = 1 cycle en 0.5 ns*
  - *30ns DRAM = 1 accès en 50 cycles*

- Méthodes pour diminuer le temps d'accès à la mémoire
  - *Accès à la mémoire avec un bus de données plus grand*
    - Extraire plusieurs octets au lieu de 1 octet chaque fois
  - *Entrelacement de Mémoire*
    - Partitionner une mémoire en sou sections, chaque avec ses registres de données et d'adresse
  - *Mémoire Cache*

Entrelacement de mémoire



Les mémoires caches

- Pourquoi a-t-on besoin de mémoire cache ?
  - *CPU 2Ghz = 1 cycle en 0.5 ns*
  - *30ns DRAM = 1 accès en 50 cycles*

- Même le plus rapide des disques dur a un temps d'accès de 10 millisecondes
  - *Avec un CPU de 2 GHz, le CPU attendant 10 ms gaspille 20 millions de cycles d'horloge!*
- Le processeur ne fonctionne pas au meilleur rythme

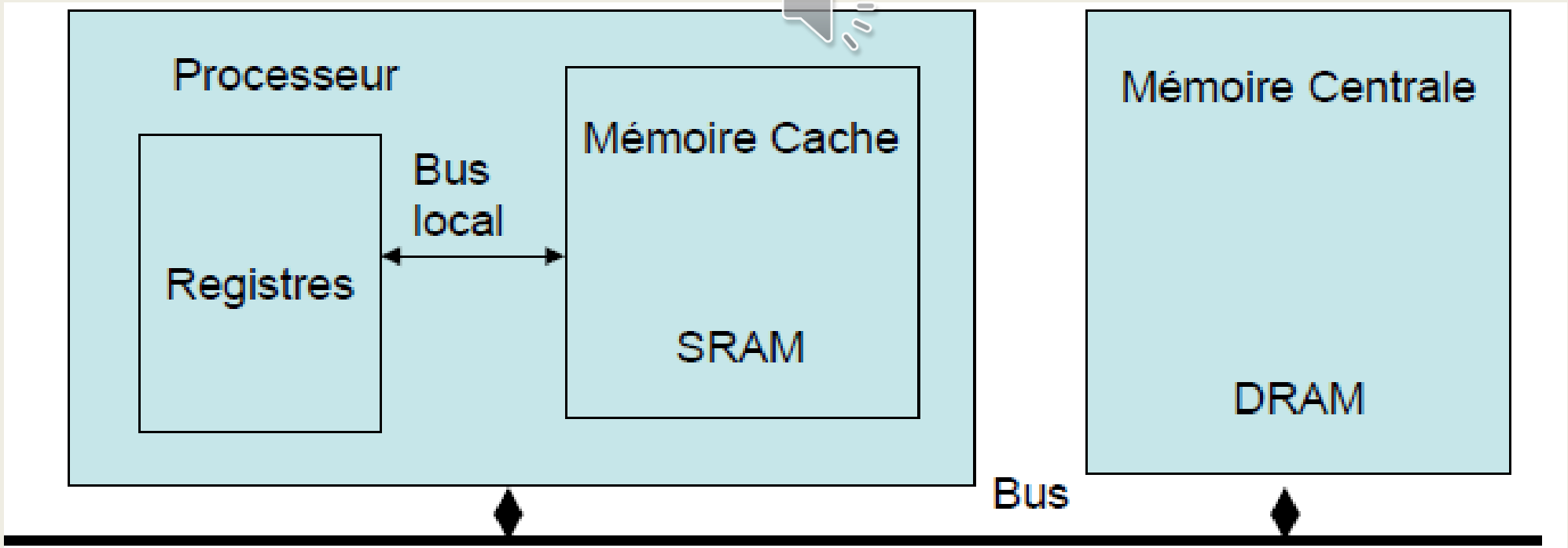
Les mémoires caches

- Typiquement, 90 % du temps d'exécution d'un programme est dépensé dans juste 10 % du code => principe de localité

  - *Localité Temporelle*
    - Une cellule mémoire référencé a plus de chance d'être référencée encore une autre fois (variable)
  - *Localité Spatiale*
    - Une cellule mémoire voisine a plus de chance d'être référencée (données stockées de manière contiguë, tableau)

Les mémoires caches

- Ajout d'un bloc mémoire rapide dans le CPU
- Principe de fonctionnement
  - *Faire coopérer des mémoires de faible capacité très rapides et à proximité du processeur avec des mémoires plus lentes et de grandes capacités*

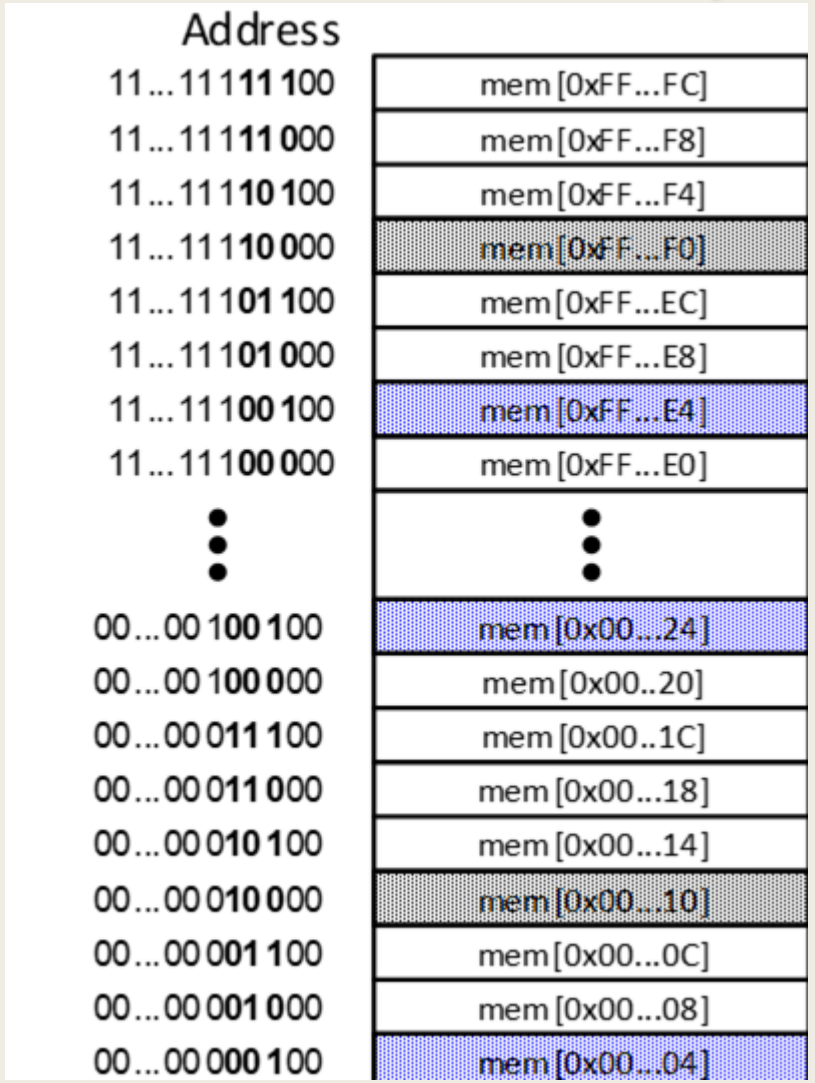


Les mémoires caches

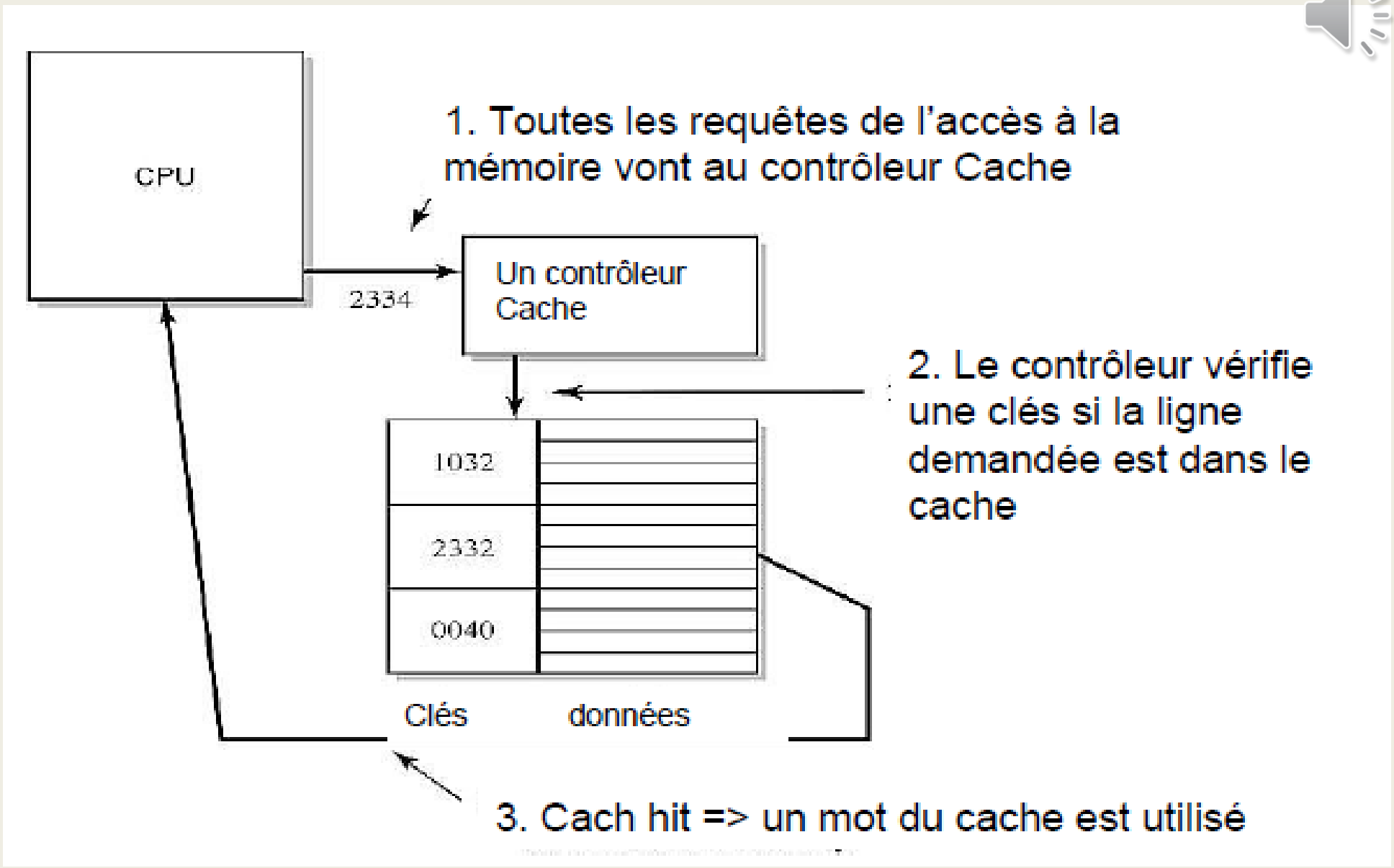
- Lecture d'un mot
  - *Si l'information est présente dans le cache on parle de succès (cache hit)*
  - *L'information n'est pas dans le cache – un échec (cache miss)*
- L'efficacité du cache dépend de son taux de succès

Les mémoires caches

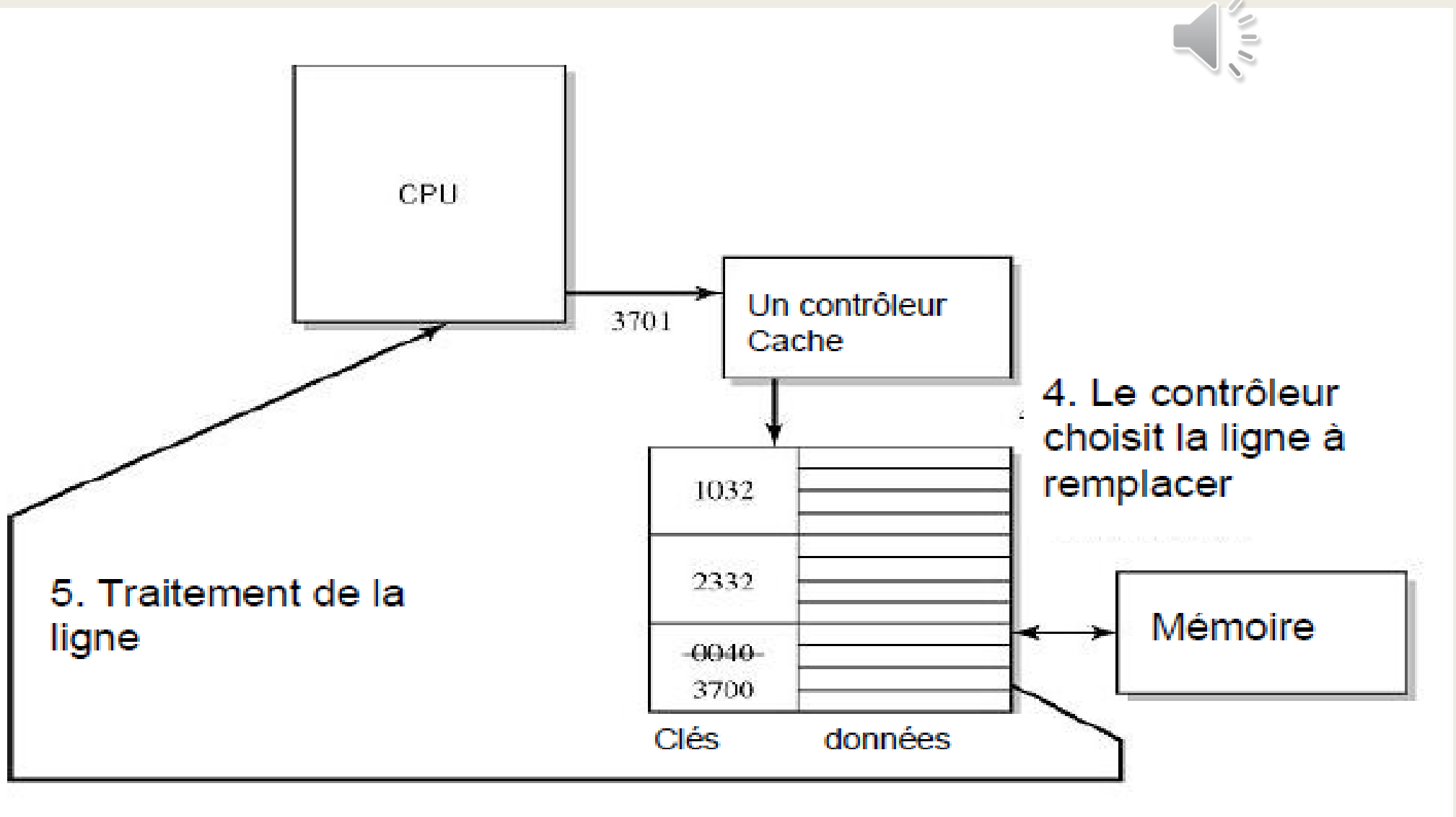
- Organisation et fonctionnement
- Le principe de localité => considérer la mémoire centrale comme une suite des blocs mémoires



Cache hit


CPU
1. Toutes les requêtes de l'accès à la mémoire vont au contrôleur Cache
2334
Un contrôleur Cache
2. Le contrôleur vérifie une clés si la ligne demandée est dans le cache
1032
2332
0040
Clés
données
3. Cach hit => un mot du cache est utilisé

Cache miss


CPU
3701
Un contrôleur
Cache
4. Le contrôleur
choisit la ligne à
remplacer
1032
2332
0040
3700
Clés
données
Mémoire
5. Traitement de la
ligne

Cache

- Facteurs déterminants une bonne conception de cache
  - *Taille*
  - *Longueur des blocs de cache*
  - *Mode de gestion du cache*
    - Minimiser le temps de vérification si l'information est présente dans le cache
  - *Nombre et la localisation du ou des caches*

Questions sur la hiérarchie mémoire Niveau - cache

- Quelles données doivent être gardées dans le cache ?
- Comment trouver un bloc s'il est présent dans le niveau supérieur ? (identification de bloc)
- Où peut-on placer un bloc dans le niveau supérieur ? (placement de bloc)
- Quel bloc doit être remplacé en cas d'échec ? (remplacement de bloc)
- Qu'arrive-t-il lors d'une écriture ? (stratégie d'écriture)

Cache, organisation

- Types de cache
  - *Cache direct*
  - *Cache purement associatif*
  - *Cache mixte ou Cache associatif par ensemble de blocs*

Identification et Placement de bloc

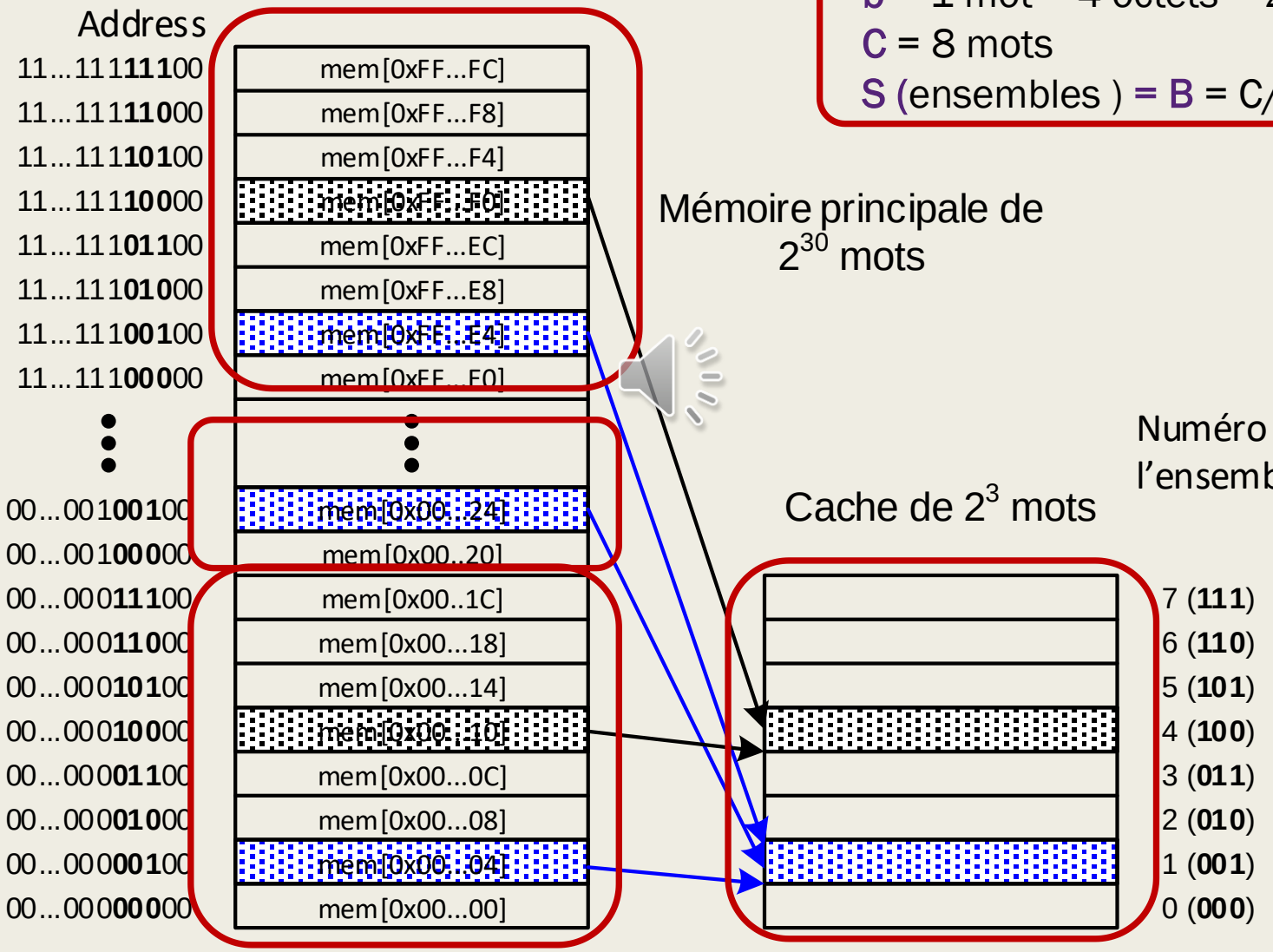
- Les blocs de mémoire cache sont organisés en ensembles
  - **S** - *Nombre d'ensembles*
- **Chaque adresse mémoire est associée à un seul ensemble dans le cache**
- Les organisations des caches se catégorisent par le nombre des blocs associés à un ensemble:
  - Cache direct : 1 bloc par ensemble => **S = B** ensembles
  - Cache associatif par ensemble de blocs : **N** blocs par ensemble => **S = B/N** ensembles
  - Cache purement associatif : tous les blocs de cache sont en un seul ensemble => **S = 1** ensemble
- Examinons chaque organisation de cache avec:
  - Capacité (C = 8 mots)
  - Taille d'un bloc (b = 1 mot = 4 octets, chaque octet est adressable)
  - Donc, nombre de blocs (B = 8)

Cache, organisation

- Cache à correspondance directe
  - *Cache le plus simple*
  - *On va affecter à chaque ligne de notre mémoire cache une zone de mémoire RAM fixe et de taille fixe*
  - *Il y a une correspondance directe entre mémoire RAM et mémoire cache*
  - *Chaque bloc ne peut être qu'à une seule position dans le cache*
  - *La position est déterminée modulo la taille du cache (i.e., le nombre des blocs)*
- Désavantage: Des conflits possibles entre lignes allant à la même position peuvent créer un effet « ping-pong »

Cache à correspondance directe

b = 1 mot = 4 octets = 2^2
C = 8 mots
S (ensembles) = B = C/b = 8 = 2^3



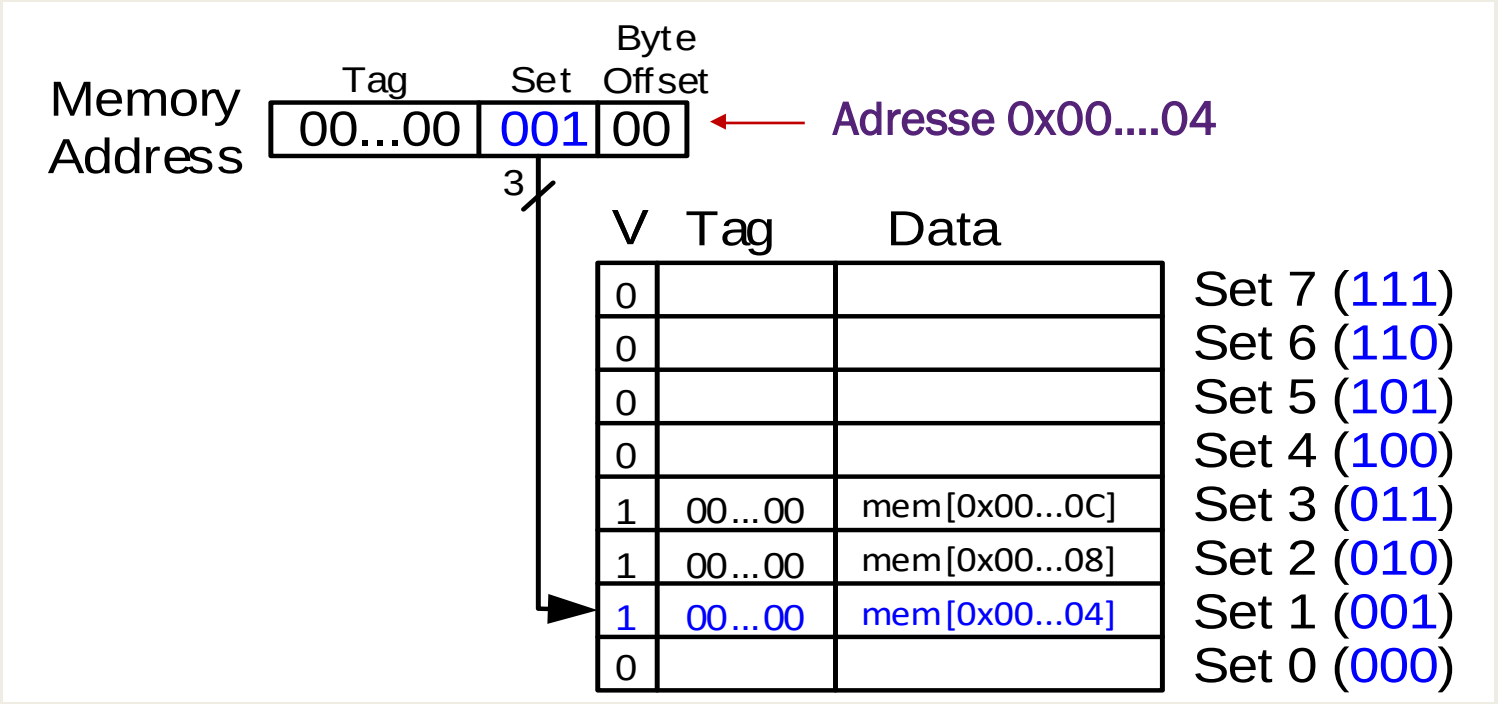
Cache direct

- Lorsqu'une adresse est présentée par CPU au cache, le contrôleur de cache décompose cette adresse

b = 1 mot = 4 octets = 2^2
C = 8 mots
S (ensembles) = B = C/b = 8 = 2^3

Boucle : 3 accès mémoire
Adresses 0x00…4, 0x00…C et 0x00…8

00…00	0000	0100
00…00	0000	1000
00…00	0000	1100

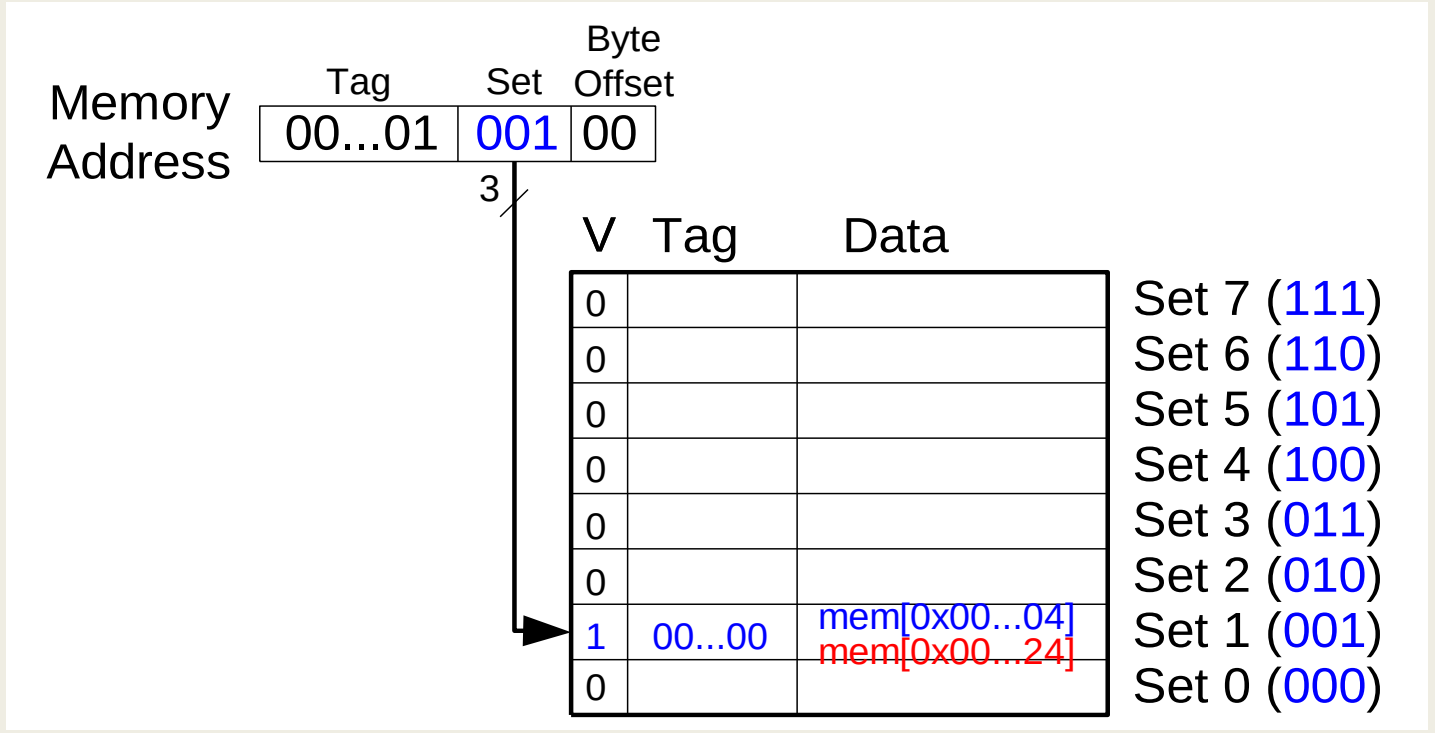


Cache direct

← Adresse 0x00....04

- Lorsqu'une adresse est présentée par CPU au cache, le contrôleur de cache décompose cette adresse

b = 1 mot = 4 octets = 2^2
C = 8 mots
S (ensembles) = B = C/b = 8 = 2^3



Boucle 5 itérations : 2 accès mémoire chaque itération
Adresses 0x00...4, 0x00...24

00...00	0000	0100
00...00	0010	0100

Cache

- Cache purement associatif
  - *Plus complexe et plus cher à construire*
  - *Il y a autant de comparateurs que de blocs de cache*
  - *Un bloc de données entre dans n'importe quelle entrée libre du cache*
  - *L'adresse est interprétée comme une étiquette et un offset*

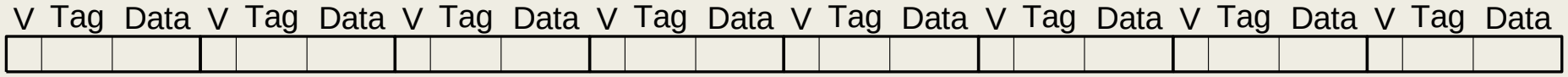
Cache purement associatif

- Gestion plus complexe
  - *Vérification si le cache est plein*
  - *Algorithme de remplacement*
    - LRU (remplacer la ligne la moins récemment utilisée)

b = 1 mot = 4 octets = 2^2
C = 8 mots ; N = B
S (ensembles) = B/N = B/B = 1

- Nécessite, pour être efficace, une recherche en parallèle des divers blocs
- N (degré d'associativité = Nombre de blocs dans un ensemble) = B
- S = 1; 0x00…4

00 …000001	00
TAG	OFFSET



Cache mixte (Cache associatif par ensemble de blocs)

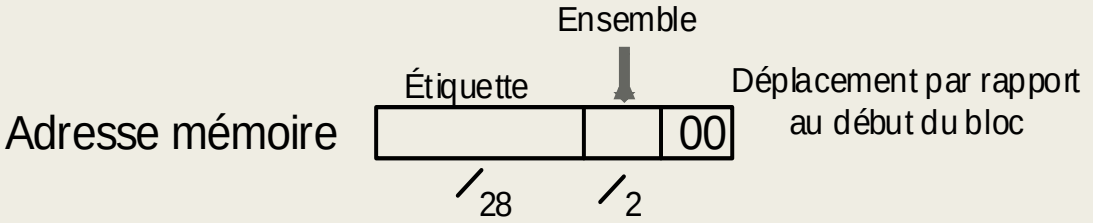
- Utilise les techniques des deux caches précédents
  - *Le cache est divisé en ensembles gérés comme des caches directs*
    - Associativité N = Nombre de blocs dans un ensemble
    - Un bloc peut aller n'importe où parmi un sous-ensemble de *N* blocs
    - Utilisation d'un algorithme de remplacement LRU
- S = B/N

Cache mixte (associatif par ensemble)

Taux d'échec = 2/10 = 20%

Associativité réduit les échecs de conflit (conflict miss)

Accès mémoire
boucle 5 itérations,
chaque itération :
0x00…4 et 0x00…024



b = 1 mot
C = 8 mots
B ensembles = C/b = 8
N = 2
S = B/N = 8/2 = 4

00...00	0000	0100
00...00	0010	0100

	Way 1			Way 0		
V	Tag	Data	V	Tag	Data	
0			0			Set 3
0			0			Set 2
1	00...10	mem[0x00...24]	1	00...00	mem[0x00...04]	Set 1
0			0			Set 0

Organisations de Cache

- Capacité: *C*
- Taille d'un bloc: *b*
- Nombre des blocs dans un cache : $B = C/b$
- Nombre des blocs dans un ensemble : *N*
- Nombre d'ensembles : $S = B/N$

Organisation	**Degré d'associativité (*N*)**	**Nombre d'ensembles ($S = B/N$)**
Correspondance directe	**1**	***B***
Cache associatif par ensemble de N blocs	**$1 < N < B$**	***B* / *N***
Cache totalement associatif	***B***	**1**

Les mémoires caches: Écriture d'un mot

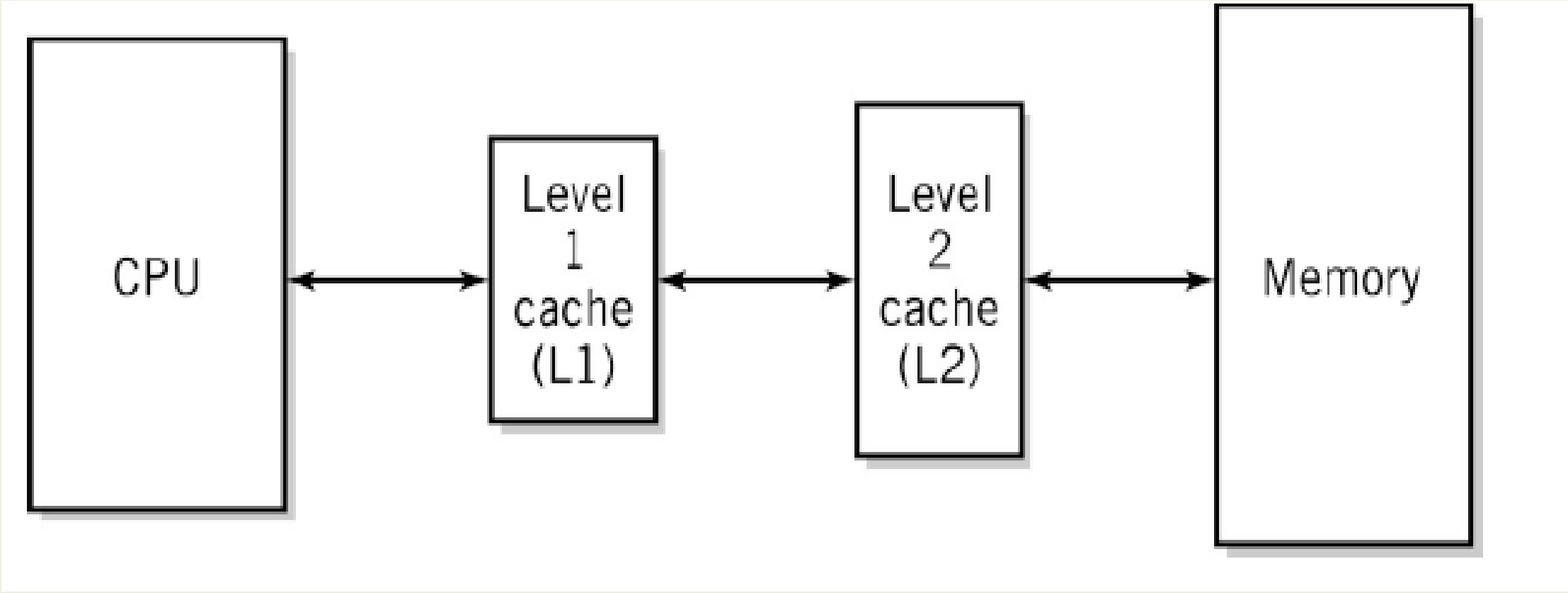
- Accéder au cache pour vérifier si l'information est dans le cache et éventuellement la modifier
- Une écriture dans le cache modifiant l'information => la modification de cette information dans la mémoire centrale
  - *Écriture immédiate (Write through)*
    - On écrit simultanément dans le cache et la mémoire principale
  - *Écriture différée (Write Back)*
    - Mettre à jour la mémoire centrale quand l'information de la mémoire cache doit être remplacée
    - Le bus de communication est libre

Nombre et localisation des caches

- Hiérarchie de mémoires cache
  - *Le premier niveau de mémoire cache, petite et très rapide, est placé dans le processeur (cache de niveau 1)*
  - *Le deuxième niveau, de capacité plus importante et d'accès également rapide, est mis à l'extérieur du processeur (cache de niveau 2)*
  - *Le troisième niveau est encore plus grand mais fait de mémoire encore mois rapide*

Caches de deux niveaux

- Pour être utile, le second niveau de cache doit avoir plus de mémoire que le premier niveau



Performance de la mémoire cache

- “Hit ratios” de 90% couramment obtenu
- Gain de plus de 50 % en rapidité d’exécution
- Technique de mémoire cache utilisée dans les disques durs

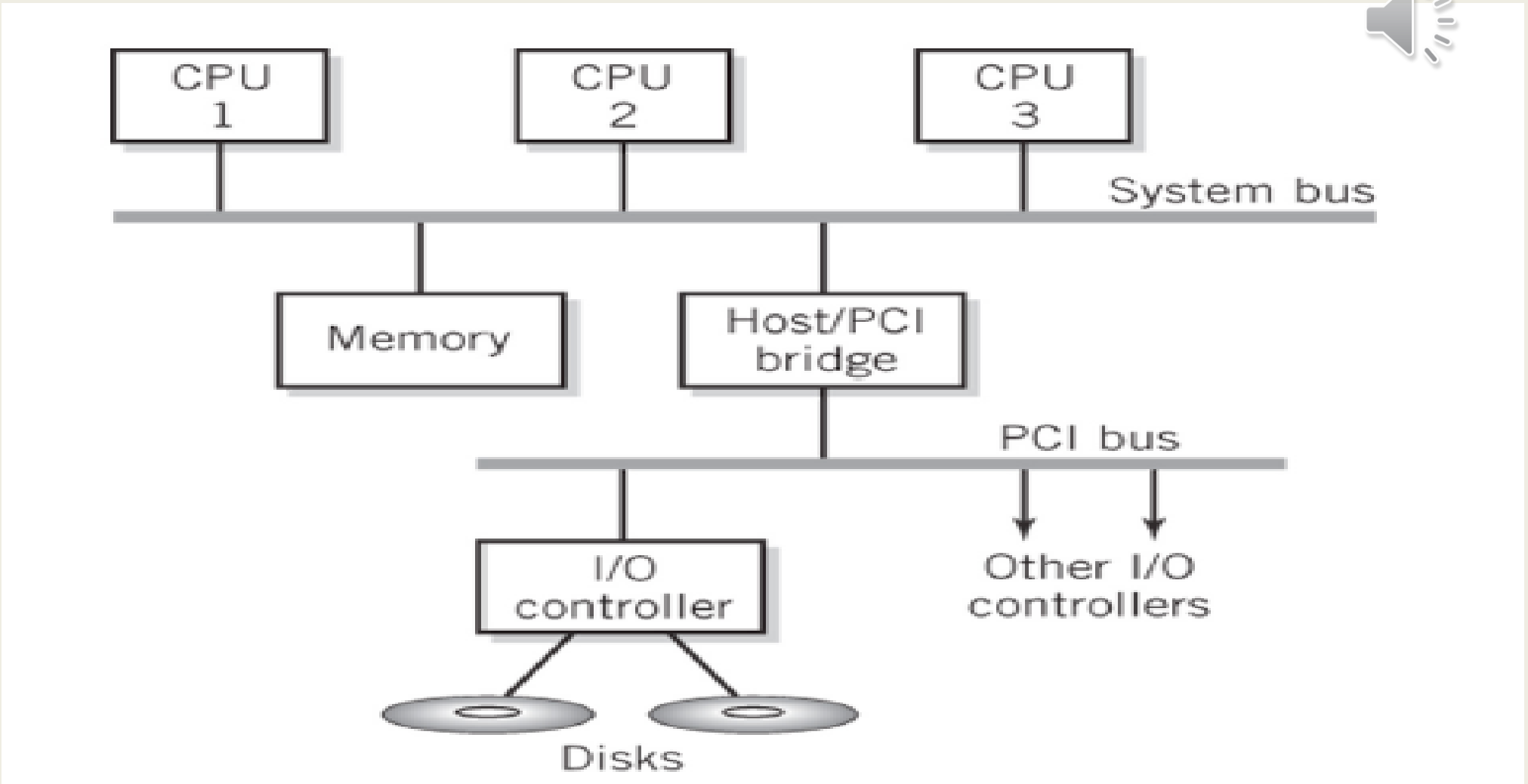
Parallélisme au niveau du processeur

- Raisons
  - *Le parallélisme au niveau des instructions multiplie la vitesse par un facteur de 5 à 10 (pipeline)*
  - *Plus de parallélisme*
    - Développement des ordinateurs équipés de plusieurs unités centrales
    - Gains en vitesse de cinquante, cent et plus
- Systèmes Multiprocesseurs
  - *CPU multiples dans un ordinateur*
  - *Processeurs multicoeurs - CPUs sont intégrés sur un seul chip*

Les Systèmes Multiprocesseurs

- Accès identique aux programmes, données, mémoire partagée, I/O, etc.
  - *Facilement étendent exécution multitâche, et exécution redondante de programmes*
- Deux méthodes de configuration
  - *Maître-esclave – approche centralisée*
  - *Symétrique - approche distribuée, Symmetrical multiprocessing - SMP*

Configuration typique du système multiprocesseurs



Approche centralisée

- CPU Maître
  - *Centralise et gère les appels systèmes*
  - *Ordonnancement (cherche à équilibrer la charge)*
- Avantages
  - *Simplicité*
  - *Protection de système et données*
- Désavantages
  - *CPU Maître peut saturer*
  - *Fiabilité – si CPU Maître tombe en panne tout le système tombe en panne*

Approche distribuée

- Tous les processeurs sont équivalents
- Ils peuvent tous exécuter le système
- Avantages
  - *Fiabilité accrue*
  - *Support de tolérance aux fautes est simple*
  - *Charge équilibrée*
- Désavantages
  - *Conflits de ressources - mémoire, i/o, etc.*
  - *Implémentation complexe*