Pipeline et gestion des mémoires

Dr. Doudou DIONE

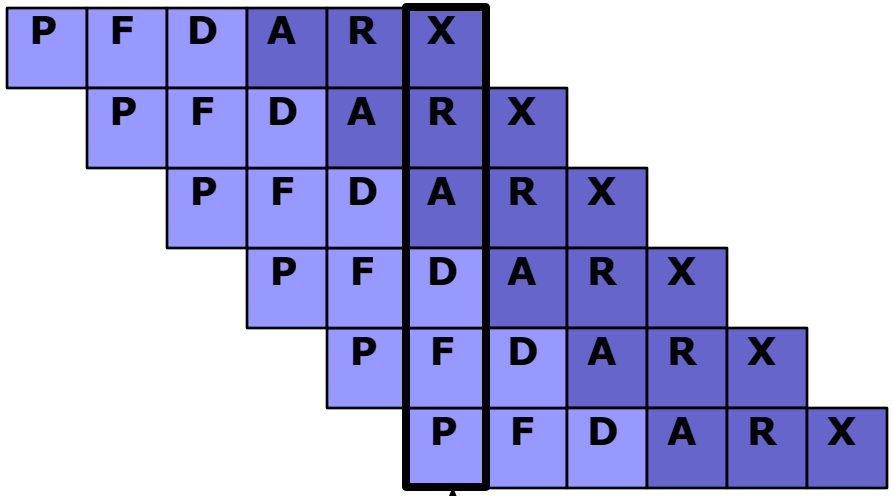
❑ Pipeline

Définition

- La technique du pipeline est une technique de mise en œuvre qui permet à plusieurs instructions de se chevaucher pendant l'exécution.
- Une instruction est découpée dans un pipeline en petits morceaux appelés étage de pipeline.
- La technique du pipeline améliore le débit des instructions plutôt que le temps d’exécution de chaque instruction
- La technique du pipeline exploite le parallélisme entre instructions d’un flot séquentiel d’instructions. Elle présente l’avantage de pouvoir, contrairement à d'autres techniques d'accélération, être rendue invisible du programmeur.

❑ Pipeline

P (Prefecth) - Generate program address	Incrémentation du compteur ordinal (PC)
F (Fetch) - Get Opcode	Lecture du code de l'instruction en mémoire
D (Decode) - Decode instruction	Décodage de l'instruction
A (Access) - Generate read address	• Calcul des adresses des opérandes • Calcul de l'adresse du résultat
R (Read) - Read operands	Lecture des opérandes en mémoire
X (Execute)	• Exécution de l'instruction • Ecriture du résultat à l'adresse calculée

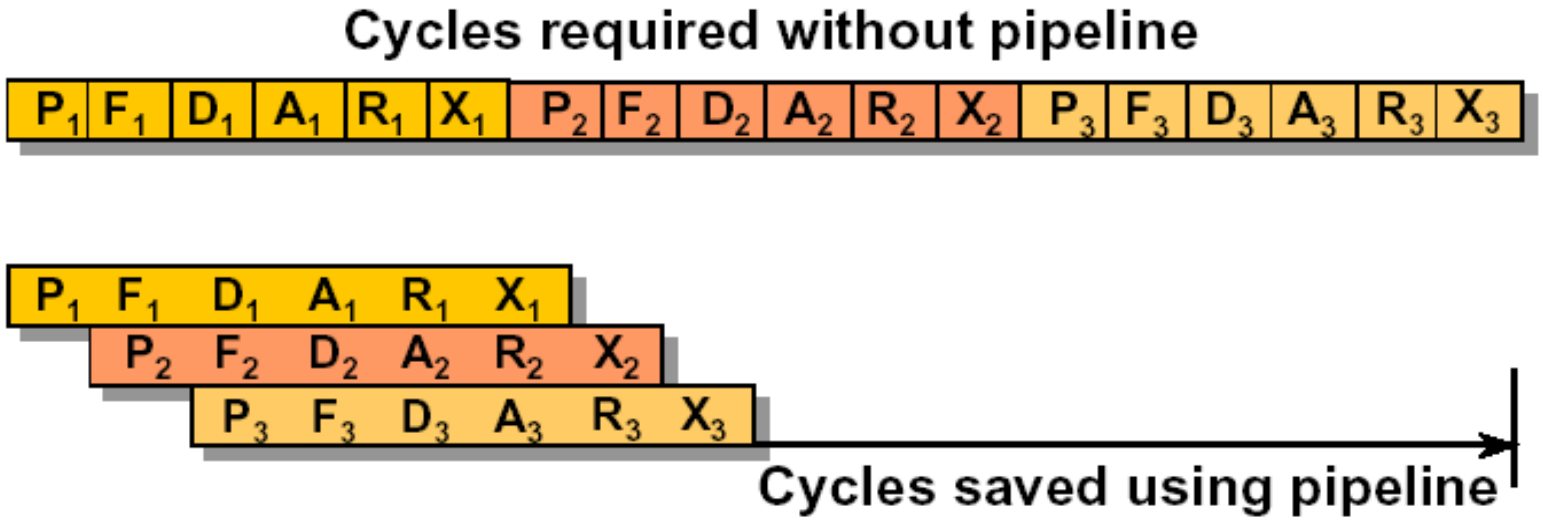


Full Pipeline : Toutes les unités matérielles du DSP sont en activités

❑ Pipeline

- Les étages d'un pipeline

Comparaison avec et sans pipeline



- Moins de cycles par instruction
- Consommation réduite

❑ Pipeline

Les étages d'un pipeline

Utilisation des ressources par le pipeline

Etage pipeline	Description	Partie hardware utilisée
P	Generate program address	PC
F	Get Opcode	Program memory
D	Decode instruction	Decoder
A	Generate read address	ARs, ARAU
R	• Read Operand • Generate write address	• Data memory • ARs, ARAU
X	• Execute instruction • Write result	• MAC, ALU • Data Memory

ARAU = Auxiliary Register Arithmetic Unit

❑ Pipeline

Les étages d'un pipeline

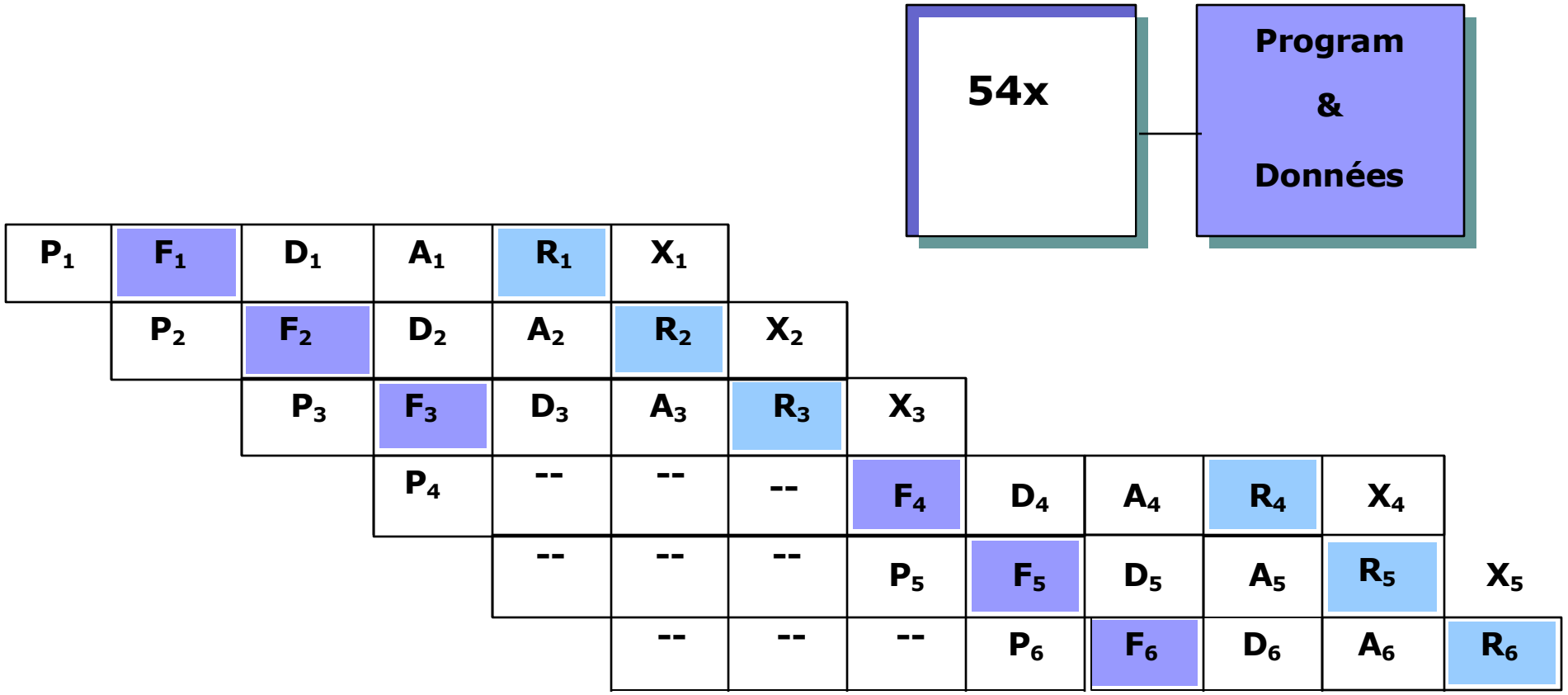
Les retards

- Le pipeline atteint son plein rendement une fois qu'il est "rempli"
- Un retard peut se produire :
  - S'il existe un conflit de ressources (retard ponctuel) :
    - ✓ accès à la mémoire
    - ✓ utilisation des bus
  - En cas de rupture de séquence (vidange du pipeline) :
    - ✓ branchement non prévu
    - ✓ appel de sous-programme
    - ✓ interruption

❑ Pipeline

Les étages d'un pipeline

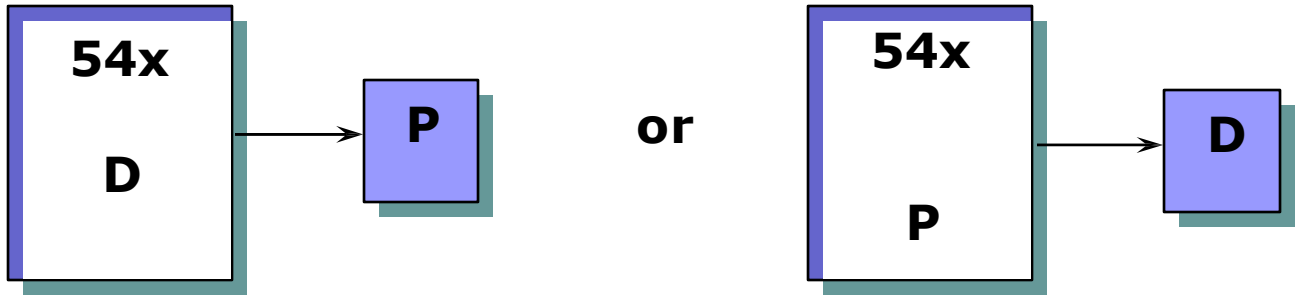
Exemple de rupture

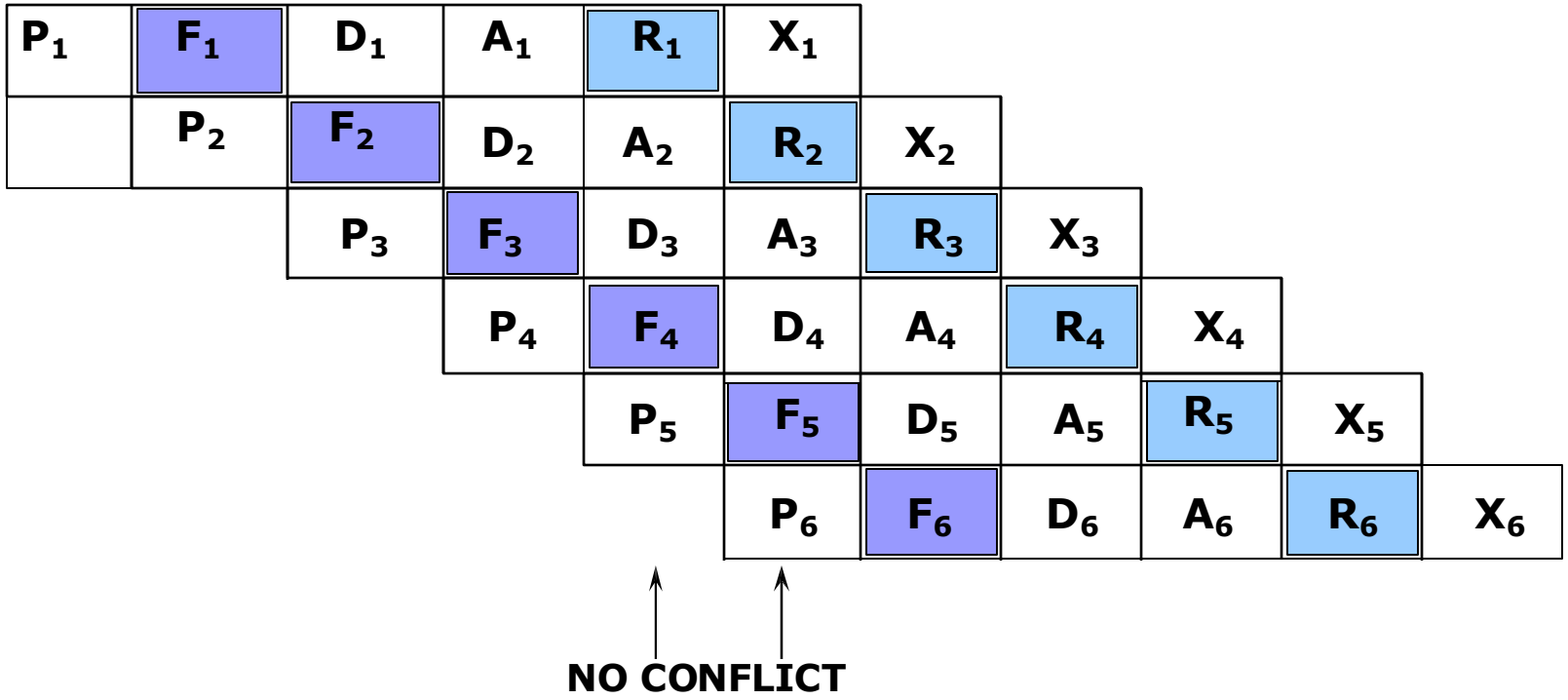


❑ Pipeline

Les étages d'un pipeline

Solution de l'organisation d'un code



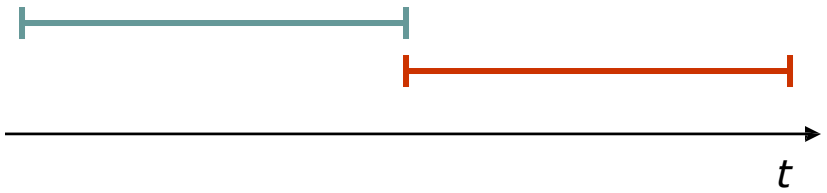


❑ Pipeline

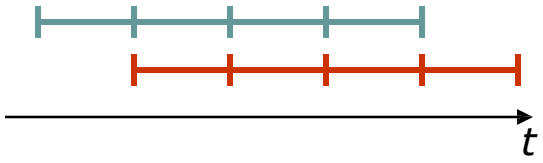
Les étages d'un pipeline

Types de pipelining

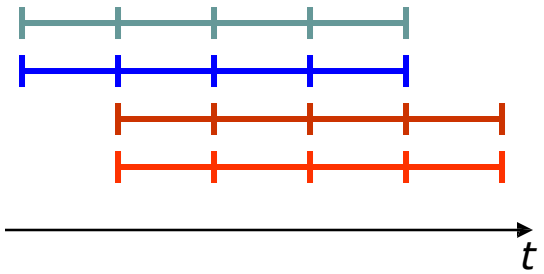
Séquentiel :
Pas de pipeline
(ex: Motorola 56000)



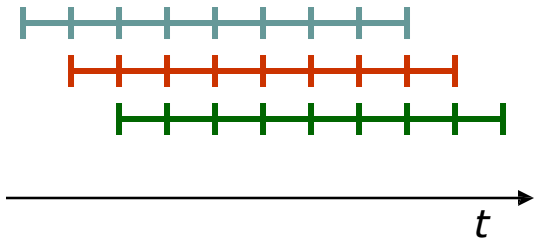
Pipeline simple
(plupart des DSP)



Double pipeline
(ex: Pentium)



Superpipeliné :
Nombre d'étages plus élevé
(ex: TMS320C6000)



❑ Pipeline

Les étages d’un pipeline

Remarques sur les performances

Certaines phases sont inutiles pour certaines instructions (p.ex. un LOAD ne nécessite pas d'exécution), mais toutes les instructions doivent traverser tout le pipeline. Ce gaspillage est nécessaire pour simplifier le contrôle.

❑ Pipeline

Les étages d'un pipeline

Processeur	***Profondeur du pipeline***
Intel Pentium 4 Prescott	31
Intel Pentium 4	20
AMD K10	16
Intel Pentium III	10
AMD Athlon	12
PowerPC G4 (PPC 7450)	7
IBM POWER5	16
IBM PowerPC 970	16
Intel Itanium	10

❑ Pipeline

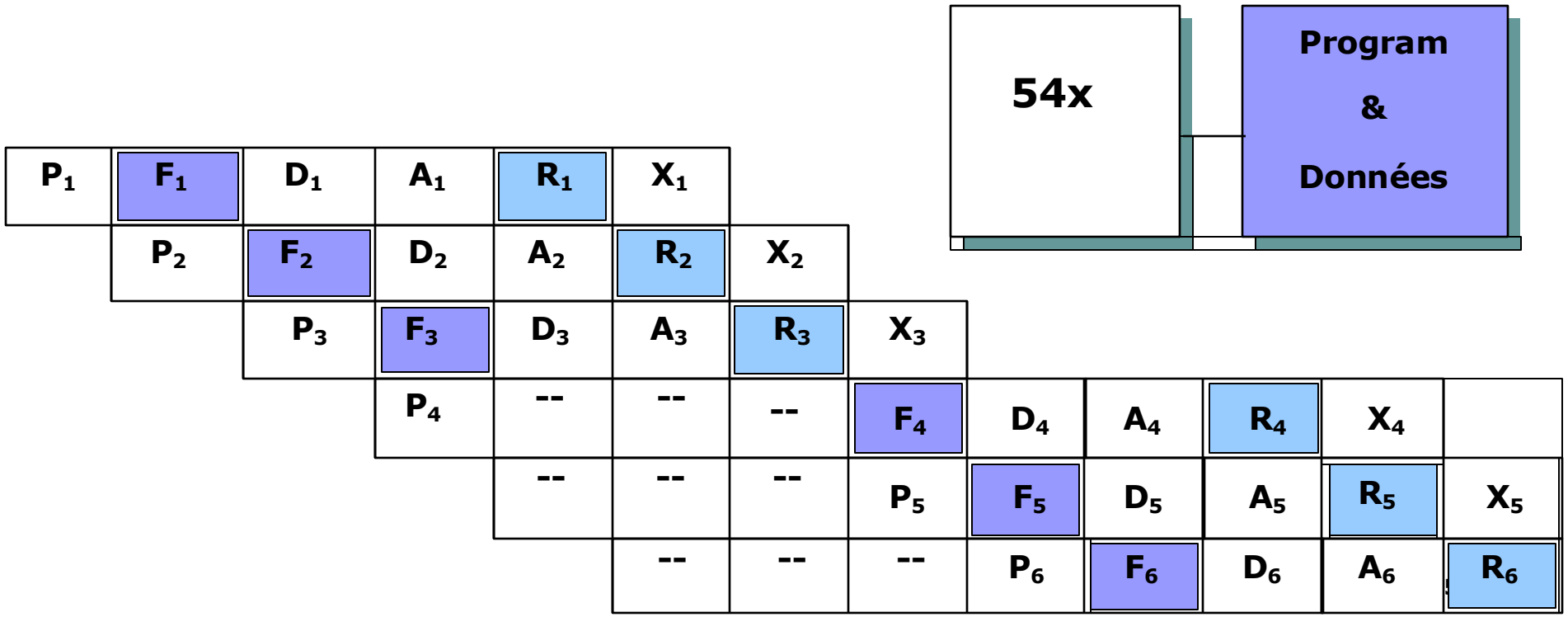
Les étages d’un pipeline

- La présence d'un pipeline (et donc le partage de l'exécution d'une instruction en plusieurs étages) introduit des aléas :
  - Aléas de structure : L'implémentation empêche une certaine combinaison d'opérations (lorsque des ressources matériels sont accédées par plusieurs étages).
  - Aléas de données : Le résultat d'une opération dépend de celui d'une opération précédente qui n'est pas encore terminée.
  - Aléas de contrôle : L'exécution d'un saut conditionnel ne permet pas de savoir quelle instruction il faut charger dans le pipeline puisque deux choix sont possibles.

❑ Pipeline

Les aléas de structure

Les aléas de structure peuvent être éliminés en agissant sur l'architecture du processeur lors de sa conception.



❑ Pipeline

Les aléas de données

- Une instruction ne peut récupérer le résultat de la précédente car celui-ci n'est pas encore disponible.

  Exemple :

  ADD R1, R2, R3 // R1 = R2 + R3

  STORE R1, 1000 // C(1000) = R1

- Cette séquence ne stocke pas à l'emplacement mémoire 1000 la valeur de R1 contenant la somme R2 + R3, mais la valeur de R1 contenue avant l'instruction ADD.

❑ Pipeline

Les aléas de données

- Prenons par exemple la séquence suivante. Cette suite d'instruction possède une dépendance directe simple. En effet A ne peut pas être disponible pour la partie droite de la seconde instruction, puisqu'elle n'est pas encore exécuter lorsque les opérandes de la seconde instruction sont chargés dans le pipeline.
  - ✓ A = B + C
  - ✓ D = A + C
  - ✓ E = F + B

- Une solution consiste à réarranger les instructions. Dans cet exemple, l'opération de la ligne 3 n'a aucune interdépendance avec les deux précédentes. Le code modifié sera :
  - ✓ A = B + C
  - ✓ E = F + B
  - ✓ D = A + C

❑ Pipeline

Les aléas de données

- Si necessaire, les instructions intercalées peuvent être des NOP (No Operation).
  - ✓ A = B + C
  - ✓ NOP
  - ✓ D = A + C
  - ✓ E = F + B

Un NOP permet de garantir que l'instruction **D = A + C** ne s'exécutera qu'après que **A** ait été correctement calculé.

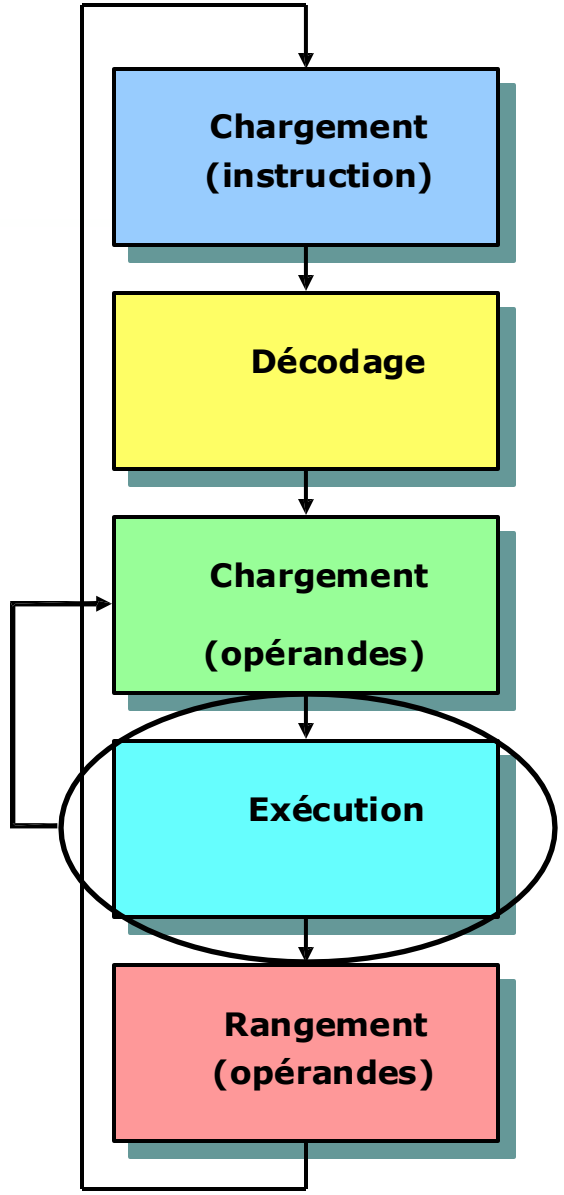
Remarques :

- Le compilateur n'est pas toujours en mesure de détecter les aléas (par exemple, si les aléas concernent des pointeurs).
- Le nombre d'instructions à intercaler dépend de la structure (nombre d'étages) du pipeline.
- La complexité du compilateur en est fortement augmentée.

❑ Pipeline

Les aléas de données

La fréquence élevée d'aléas de données peut justifier l'introduction de matériel supplémentaire. On peut par exemple introduire une connexion directe entre la sortie de l'étage d'exécution et l'étage de chargement des opérandes. Ceci permet au résultat d'une instruction d'être un opérande de l'instruction suivante.

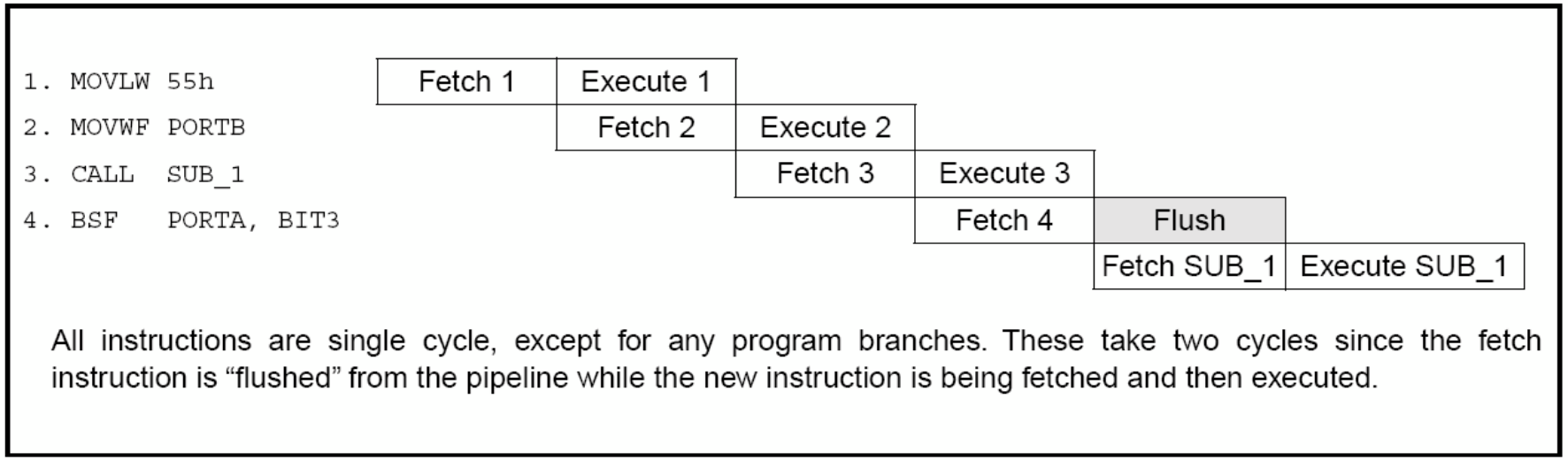


Pipeline pouvant réaliser cette solution.

❑ Pipeline

Les aléas de contrôle

La présence d'un pipeline introduit des complications lors de l'exécution d'un saut ou d'un saut conditionnel. L'étage de décodage de l'instruction n'est pas en mesure de calculer l'adresse de l'instruction suivante avant de connaître le résultat de l'instruction précédente.



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

❑ Pipeline

Les aléas de contrôle

- Une solution possible est de faire en sorte que le processeur devine si le branchement sera pris ou pas pris (branch prediction) et commencer à exécuter les instructions correspondant à cette décision.
  - ✓ Si le choix se révèle correct, la pénalité de branchement est éliminée.
  - ✓ Si le choix se révèle incorrect, il faudra vider le pipeline et charger l'instruction correcte.

- Pour faire de la prédiction de branchement il y a deux possibilités
  - ✓ Solution statique : La direction du branchement est fixe, définie en matériel au moment de la conception du processeur.
  - ✓ Solution dynamique : La direction du branchement est définie au moment de l'exécution du programme, sur la base d'une analyse du code.

La **prédiction de branchement** permet d'améliorer les performances des processeurs en réduisant les temps d'attente associés aux branchements conditionnels. Cependant, si la prédiction est incorrecte, il y a un coût en termes de performance, car le pipeline doit être vidé et réajusté

Les aléas de contrôle

Solution statique

- Les sauts en arrière (boucles) sont plus souvent pris que pas pris. En effet, une boucle est souvent réaliser avec plus que 2 itérations.

On peut donc faire une prédiction selon la direction:

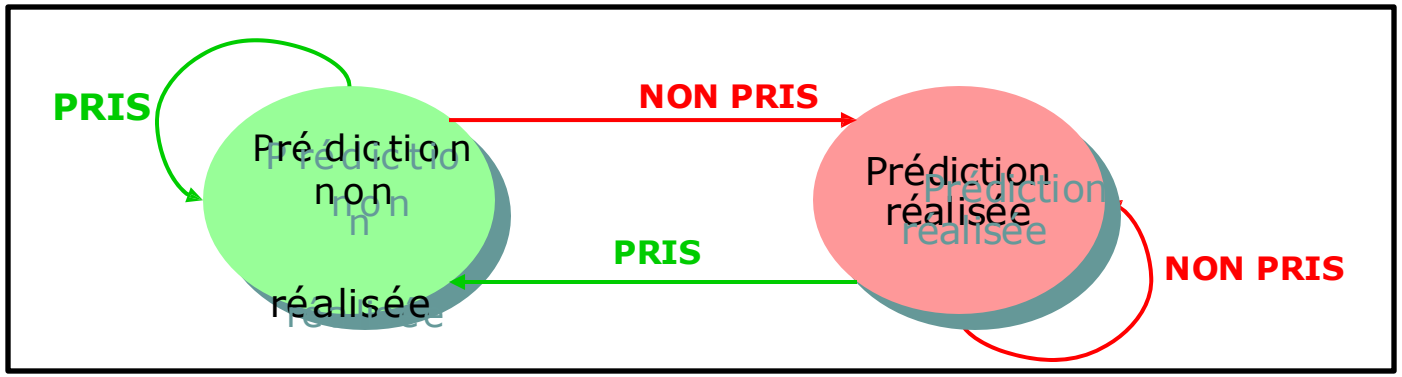
- ✓ si le saut est en arrière, il est pris,
- ✓ s'il est en avant, il n'est pas pris.

Cette stratégie donne des très bons résultats (70-80%) avec une augmentation relativement restreinte de la logique de contrôle: elle est utilisée dans plusieurs processeurs (p.ex., MicroSparc, HP-PA).

Les aléas de contrôle

Solution dynamique

- Pour réaliser une prédiction dynamique le processeur mémorise le comportement du programme lors de l'exécution des sauts. À chaque exécution d'un branchement dans un programme, le processeur mémorise si le saut était pris ou pas pris dans un tampon de prédiction de branchement. Sur la base du comportement passé du programme pour un branchement donné, le processeur prédit son comportement pour l'exécution suivante du même saut.
- Par rapport à la prédiction statique, la prédiction dynamique est plus performante, mais nécessite une quantité très importante de logique de contrôle.

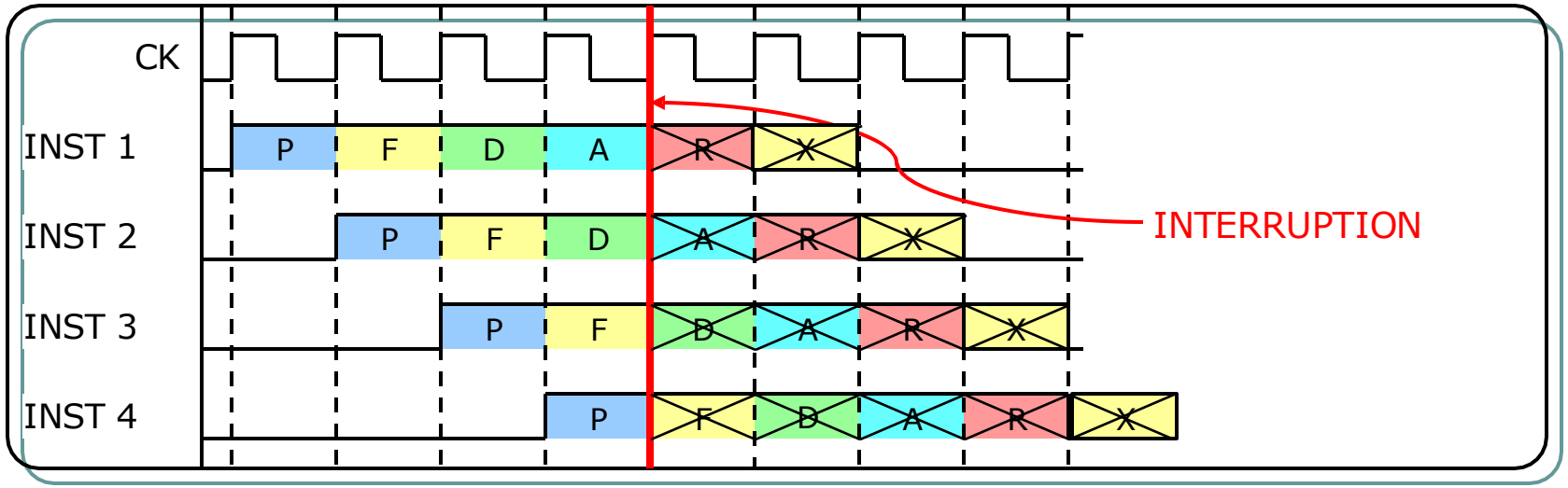


Machine d'état présente dans le processeur pour la prédiction de branchement statistique

Les aléas de contrôle

Gestion des interruptions

La présence d'un pipeline complique le traitement des interruptions: lors du déclenchement d'une interruption non-masquable, la routine de traitement doit parfois être lancée immédiatement. Le pipeline contiendra alors les instructions partiellement exécutées.



Résumé

- Le pipeline améliore le débit mais pas le temps par instruction : il faut toujours cinq cycles à une instruction d'un pipeline à cinq étages pour s'exécuter.

- Les dépendances de données et de contrôle dans les programmes imposent une limite supérieure au gain que peut générer le pipeline car le processeur doit parfois attendre la fin d'une instruction pour que les dépendances soit résolues.

- On peut élever cette limite, mais pas l'éliminer, en réduisant les aléas de contrôle par des optimisations, et les aléas de données par un ordonnancement des instructions par le compilateur.

❑ Gestion des mémoires

Principes du cache

- Les mémoires doivent répondre à deux contraintes contradictoires :
  - ✓ Taille importante
  - ✓ Temps d'accès court
- Principe de base du cache :
  - ✓ Les mots mémoires les plus fréquemment utilisés sont conservés dans une mémoire rapide (cache) plutôt que dans une mémoire lente (mémoire centrale).

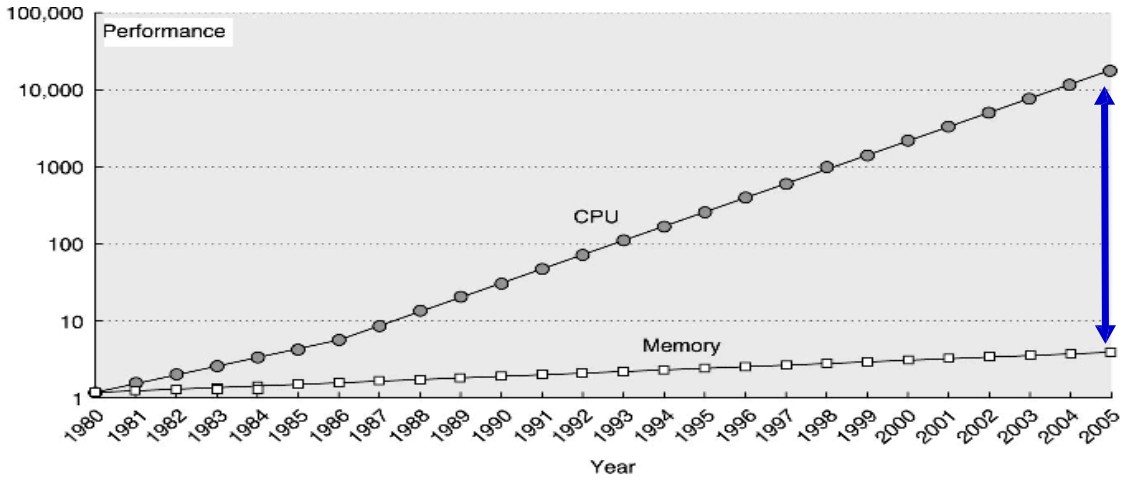
❑ Gestion des mémoires

Objectifs et principes du cache

Vitesse des mémoires et des processeurs

- **Évolution**

Année	Temps de cycle processeur	Temps de cycle mémoire
1990	~100ns	~140ns
1998	~4ns	~60ns
2002	~0.6ns	~50ns



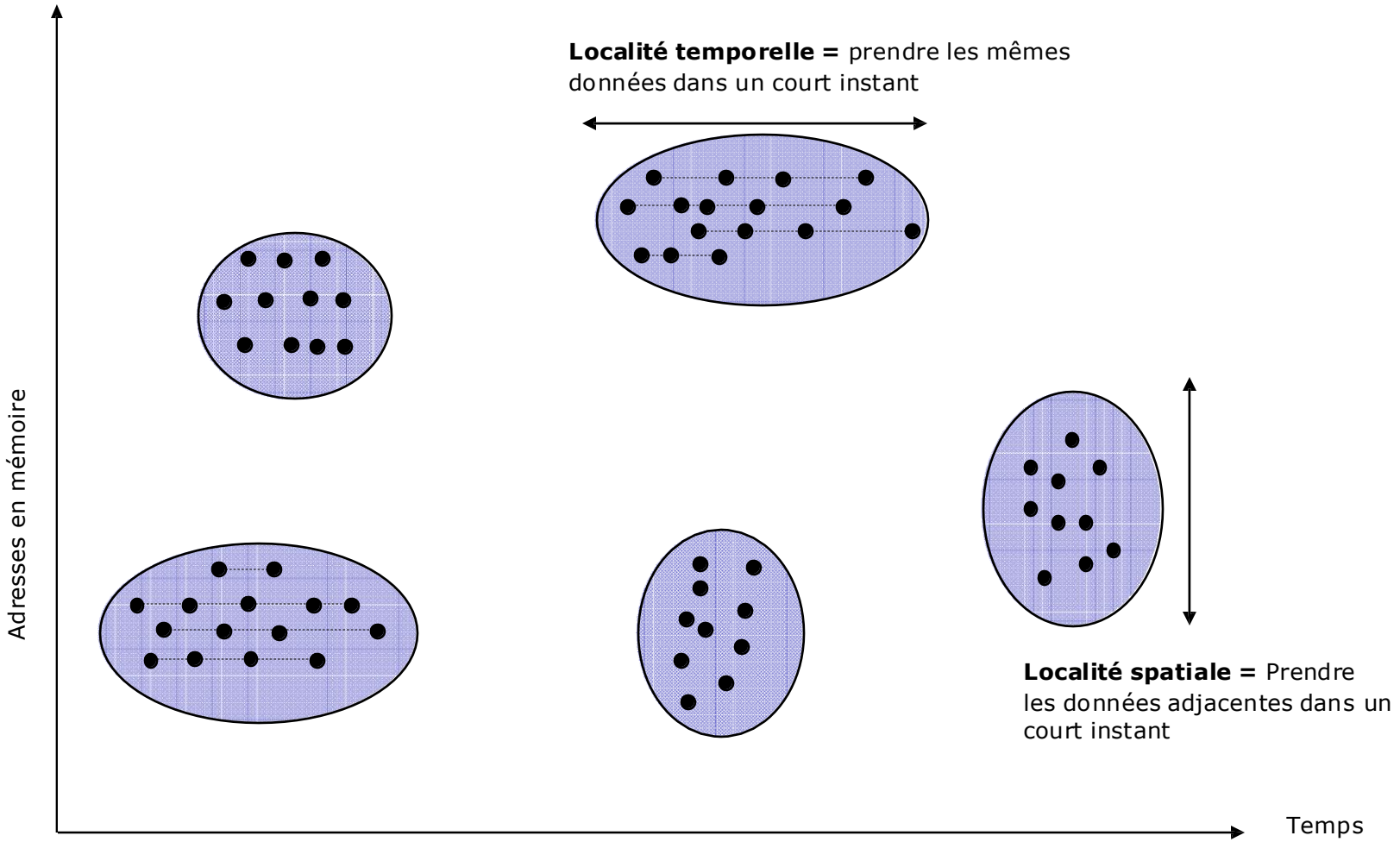
❑ Gestion des mémoires

Objectifs et principes du cache

Principe de localité

Localité spatiale : Tendance à accéder aux données qui sont proches de celles récemment utilisées

Localité temporelle : Tendance à réutiliser des données récemment utilisées



❑ Gestion des mémoires

Objectifs et principes du cache

Principe de localité

Les données

```
for (i=0; i<N; i++) {
  for (j=0; j<N; j++) {
    y[i] = y[i] + a[i][j] * x[j]
  }
}
```

- y[i]: propriétés de localités temporelle et spatiale.
- a[i][j]: propriété de localité spatiale.
- x[j]: propriété de localité temporelle et spatiale.

Le programme

```
                ...
05  LOOP        LDR R1, R0, #3
06              ADD R1, R1, #5
07              STR R1, R0, #30
08              ADD R0, R0, #1
09              ADD R3, R0, R2
0A              BRn LOOP
                ...
```

Boucle : réutilisation des instructions :

localité temporelle

Instructions consécutives en mémoire :

localité spatiale

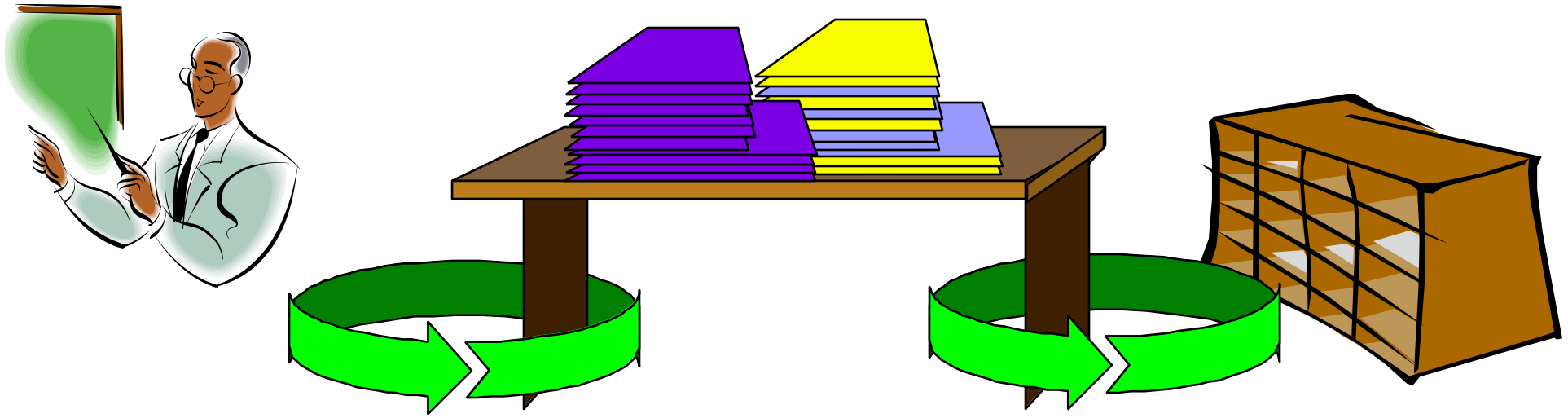
❑ Gestion des mémoires

Objectifs et principes du cache

Principe de localité

Analogie

- Homme = Unité de calcul
- Le bureau = mémoire cache
- La bibliothèque = mémoire centrale



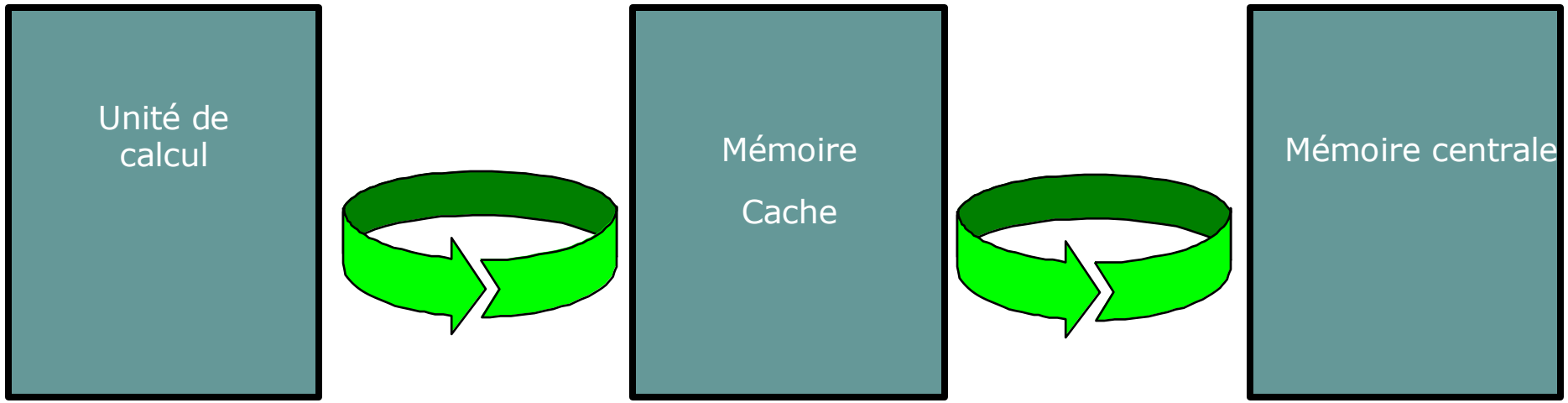
❑ Gestion des mémoires

Objectifs et principes du cache

Principe de localité

L'élément de base est le bloc

4 octet < nombre d'octet par bloc < 128 octets

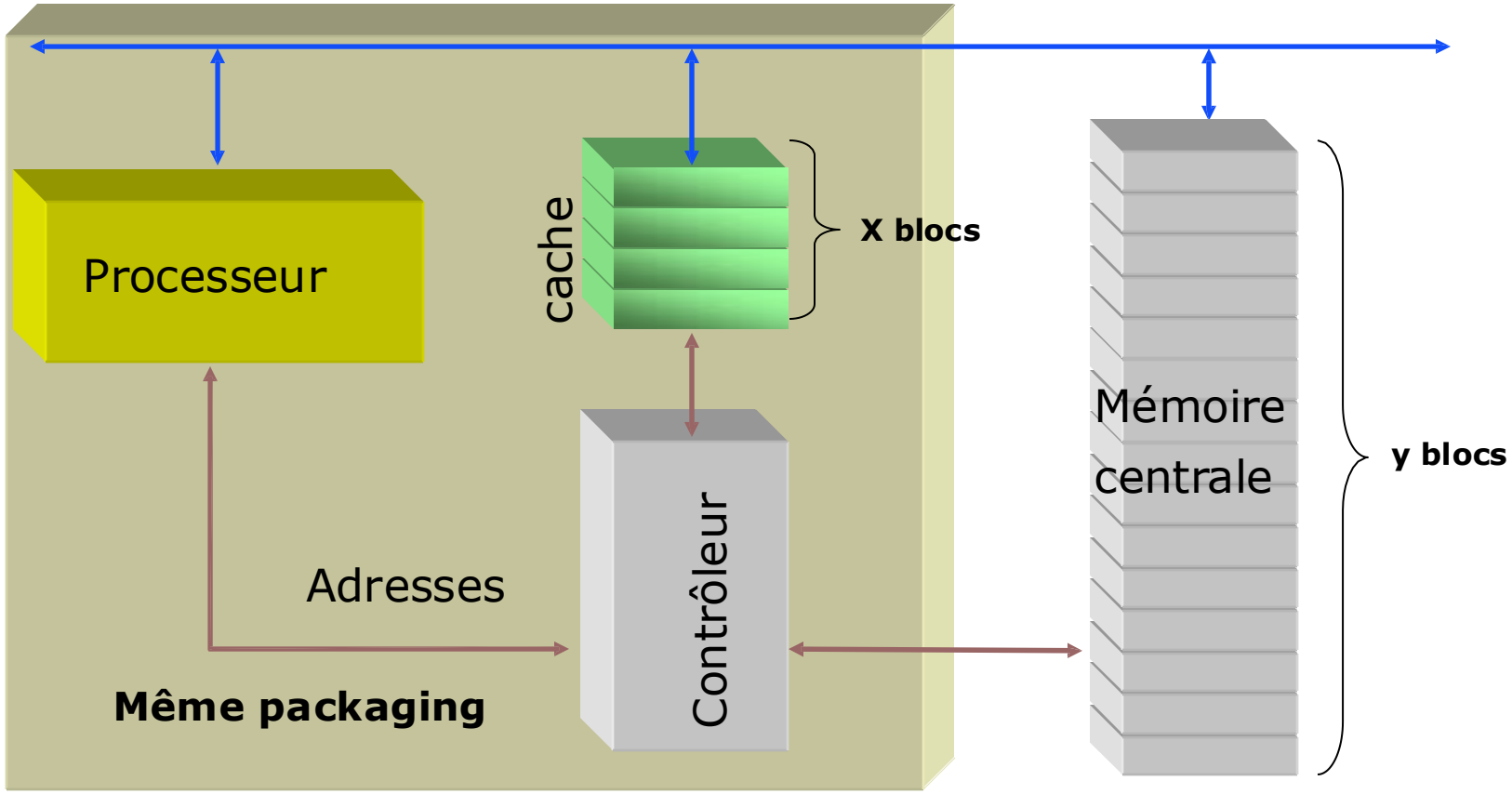


❑ Gestion des mémoires

Objectifs et principes du cache

Organisation de la mémoire

- La mémoire cache possède x blocs
- La mémoire centrale possède y blocs
- Y>>x



❑ Gestion des mémoires

Objectifs et principes du cache

Placement des données dans la mémoire

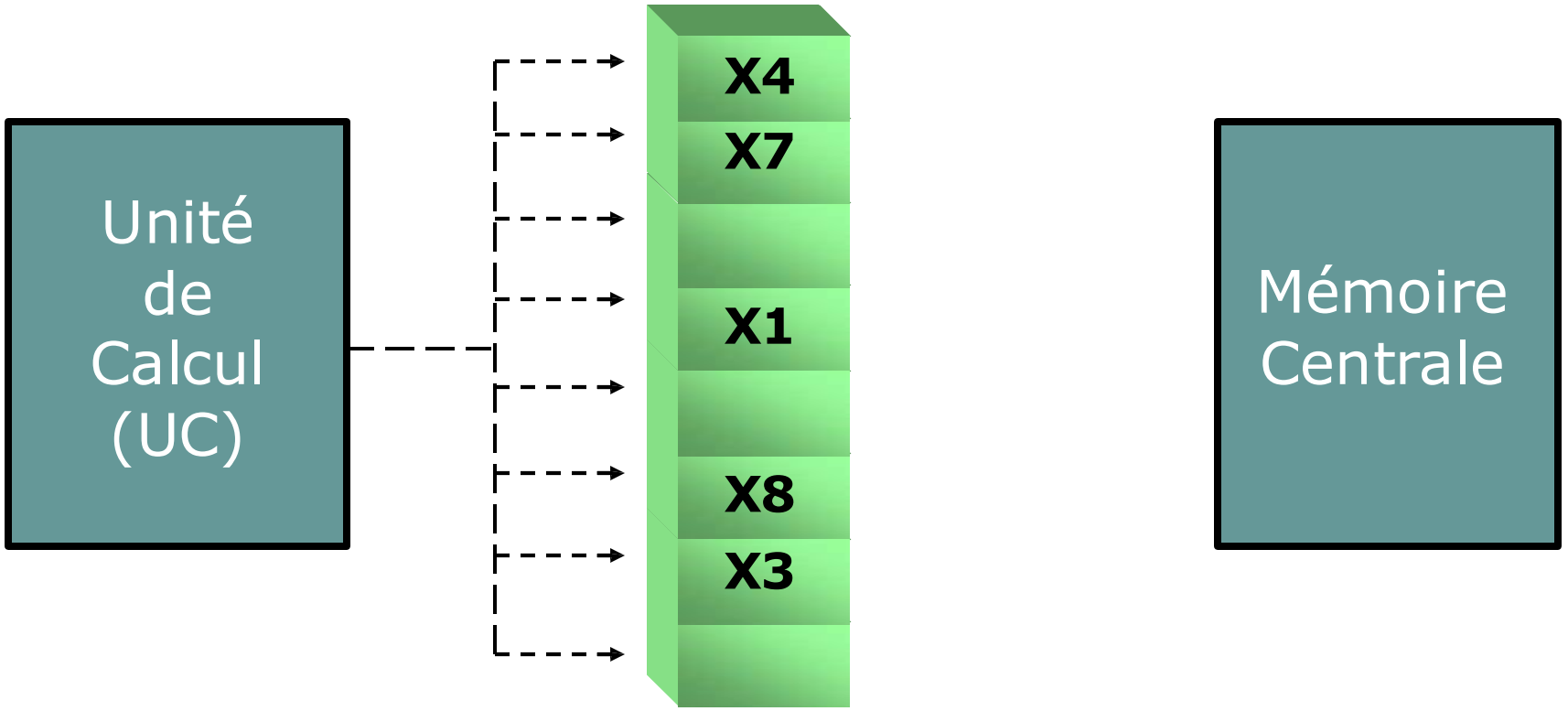
- Le placement des données dans le cache est géré par le matériel :
  - ✓ le programmeur n’a pas à se soucier du placement des données dans le cache
  - ✓ En revanche le programmeur devra prendre en considération la présence du cache pour optimiser les performances.
  - ✓ le fonctionnement du cache est transparent pour le programmeur.

❑ Gestion des mémoires

Objectifs et principes du cache

Principe général

- L'UC veut faire référence à un bloc X2 dans le cache
  - ✓ Recherche de X2 dans le cache
  - ✓ Défaut de cache

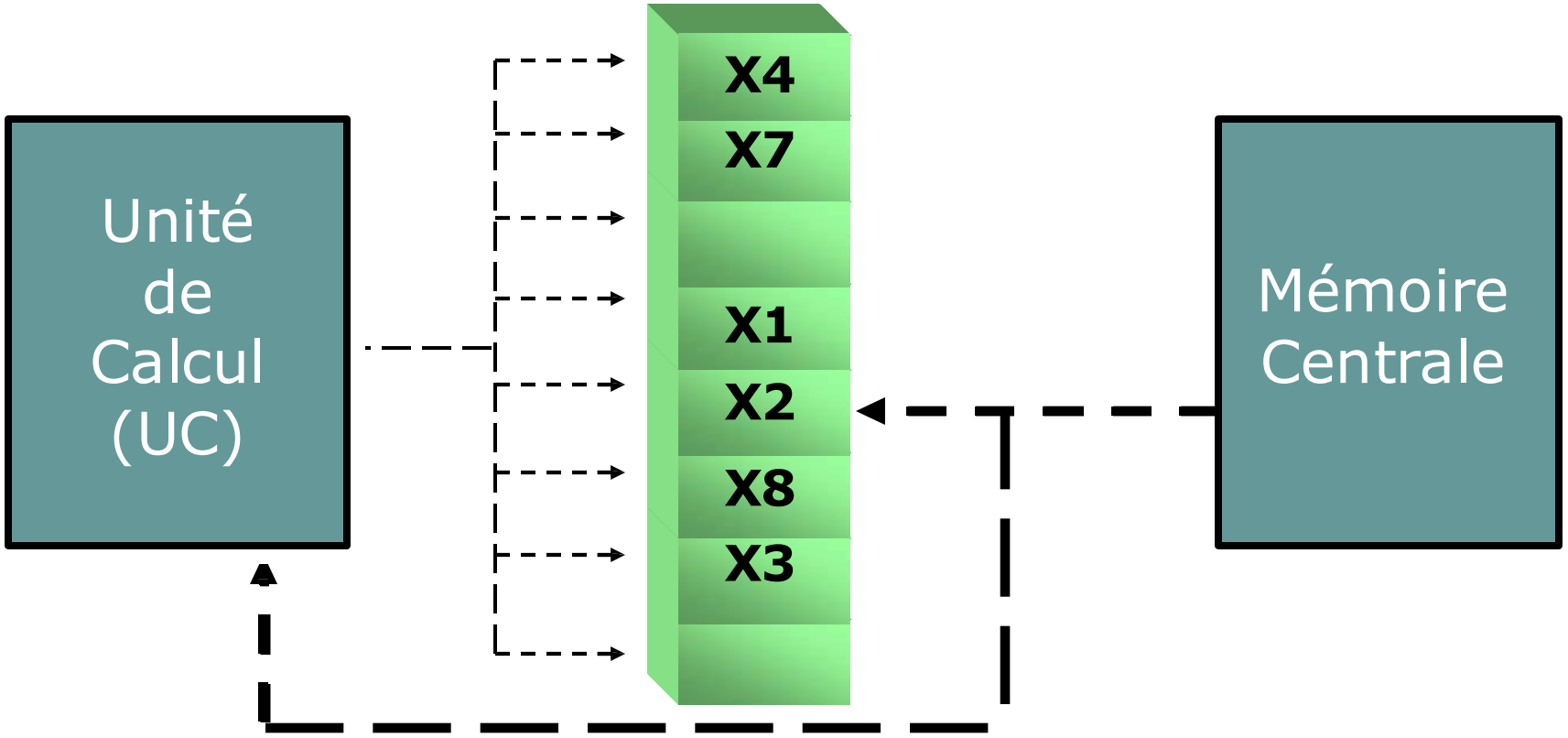


❑ Gestion des mémoires

Objectifs et principes du cache

Principe général

- L'UC veut faire référence à un bloc X2 dans le cache
  - ✓ Recherche de X2 dans le cache
  - ✓ Défaut de cache

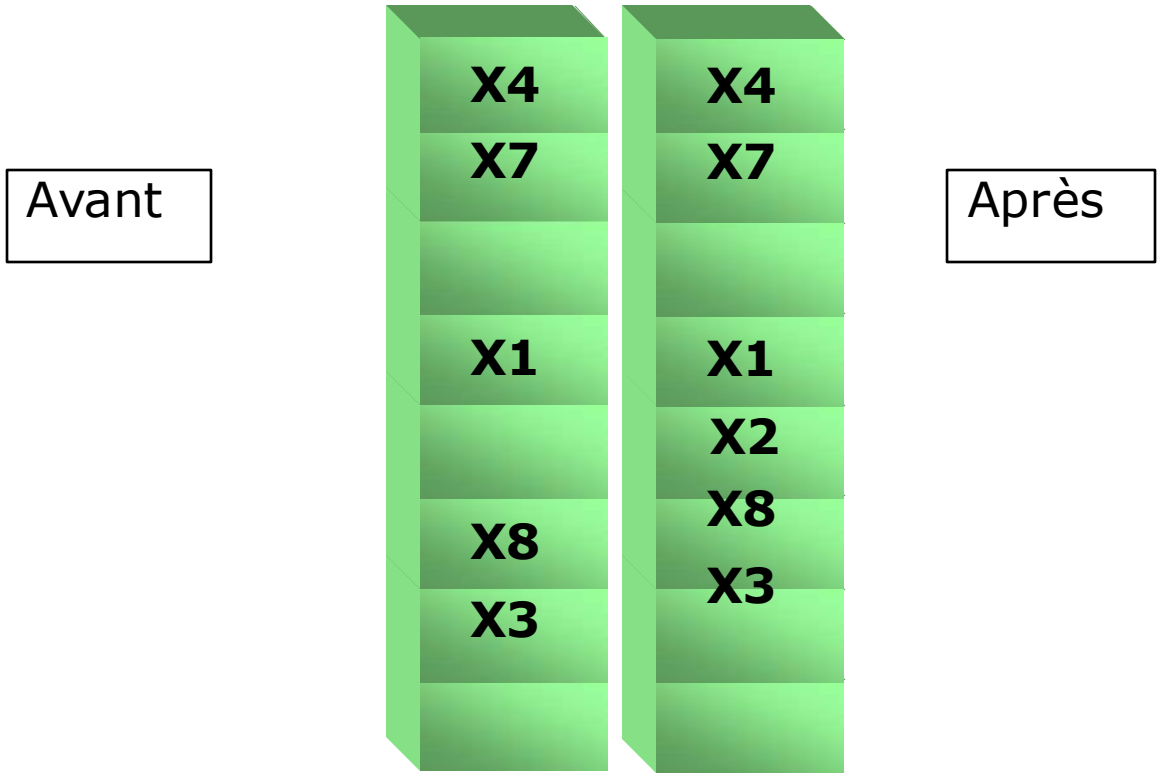


❑ Gestion des mémoires

Objectifs et principes du cache

Principe général

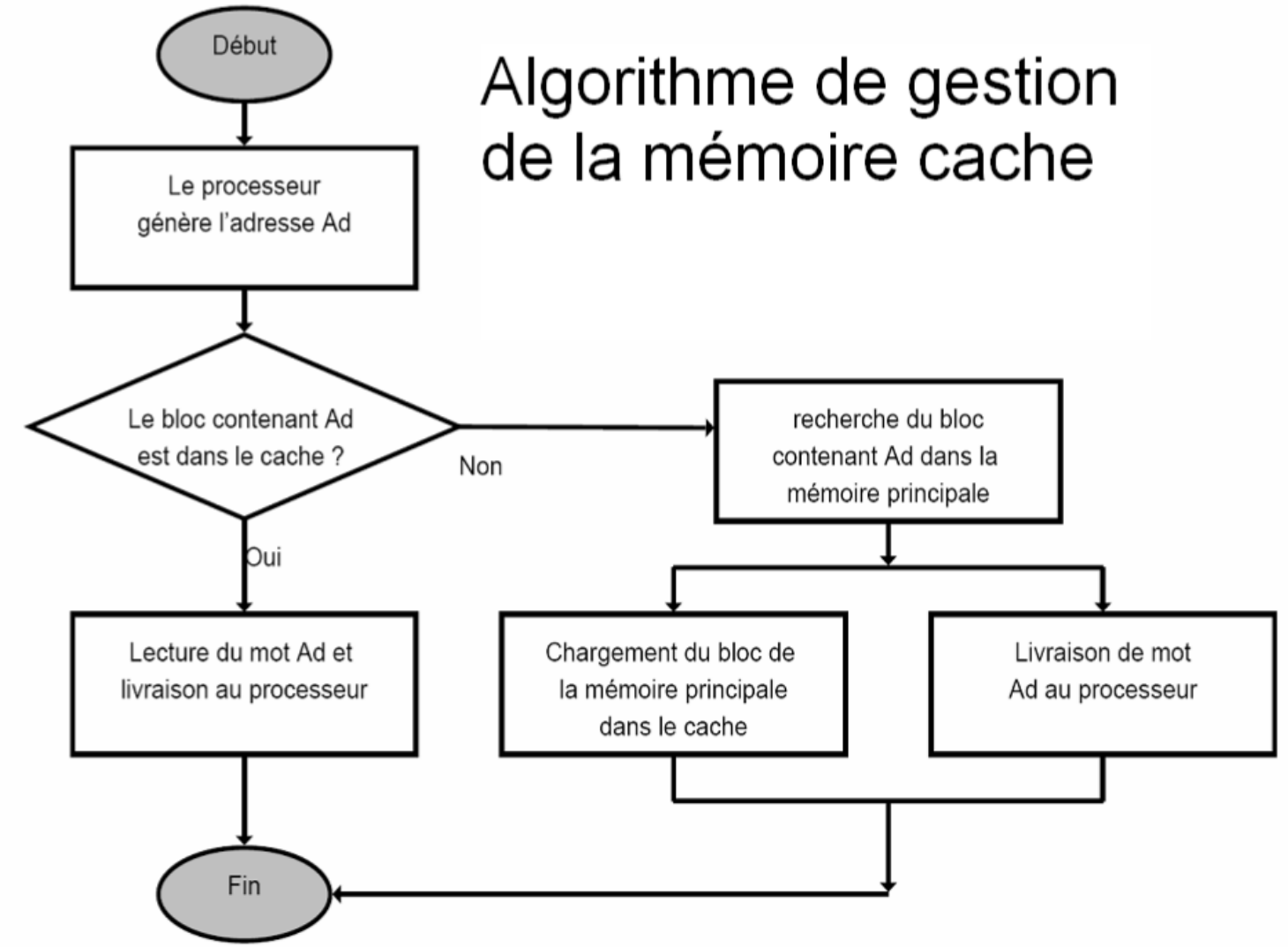
- Il y a eu transfert d'un nouveau bloc (X2) de la mémoire centrale, dans la mémoire cache.



❑ Gestion des mémoires

Objectifs et principes du cache

Principe général



❑ Gestion des mémoires

Objectifs et principes du cache

Bloc ou ligne de cache

- L'unité d'information qui peut être présente ou non dans le cache est appelée un bloc, qui constitue une ligne (ou rangée) du cache. Les blocs ont généralement entre 4 et 128 octets et sont tous de même taille dans un cache donné.
- La mémoire centrale et la mémoire cache ont impérativement les même tailles de blocs.

❑ Gestion des mémoires

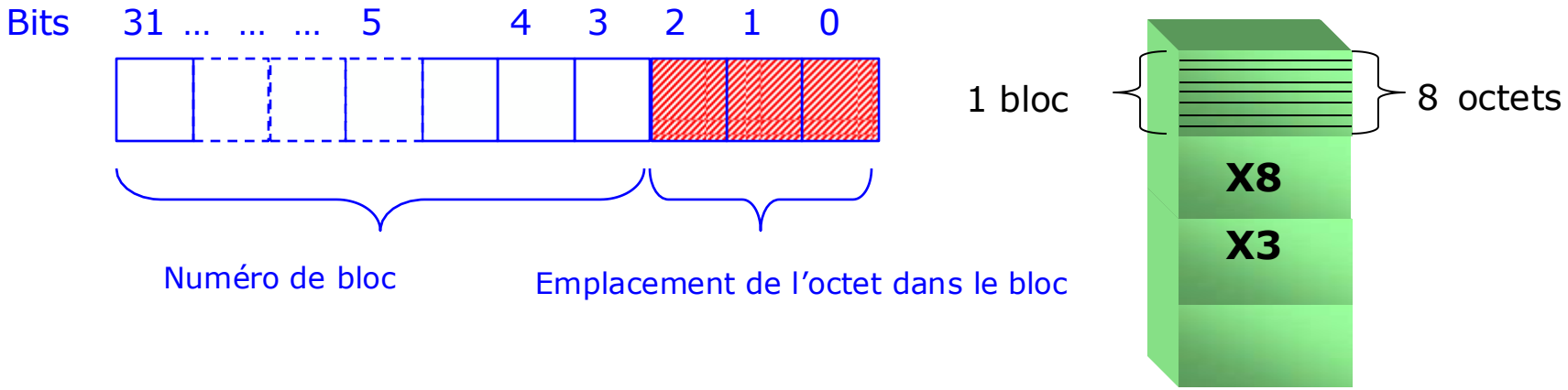
Objectifs et principes du cache

Bloc ou ligne de cache

L'adresse fournie par le processeur peut être scindée en deux partie :

- Le numero du bloc
- L'adresse du bloc

Exemple : @ sur 32 bits et blocs de 8 octets



❑ Gestion des mémoires

Objectifs et principes du cache

Bloc ou ligne de cache

À chaque bloc on associe

Une étiquette :

- La valeur de cette étiquette permettra de décider si un bloc donné est effectivement dans le cache ou non.

Un bit de validité :

- Il permet de savoir si les données du bloc ou sont obsolètes pas.

❑ Gestion des mémoires

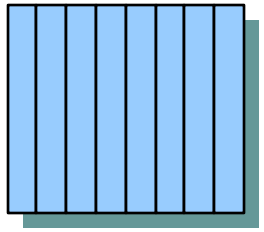
Objectifs et principes du cache

Différentes organisation de cache

Où placer une ligne de la mémoire principale dans le cache?

Complètement associatif

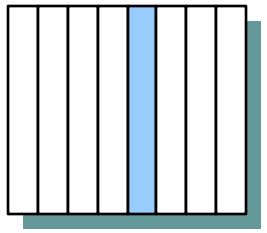
Bloc no. 0 1 2 3 4 5 6 7



Cache à correspondance direct

Bloc i: 4 (12 mod 8)

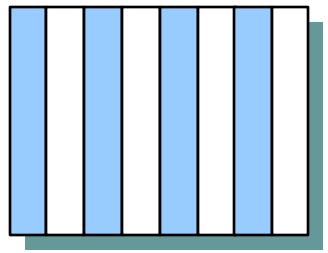
Bloc no. 0 1 2 3 4 5 6 7



Cache associatif par ensemble de bloc

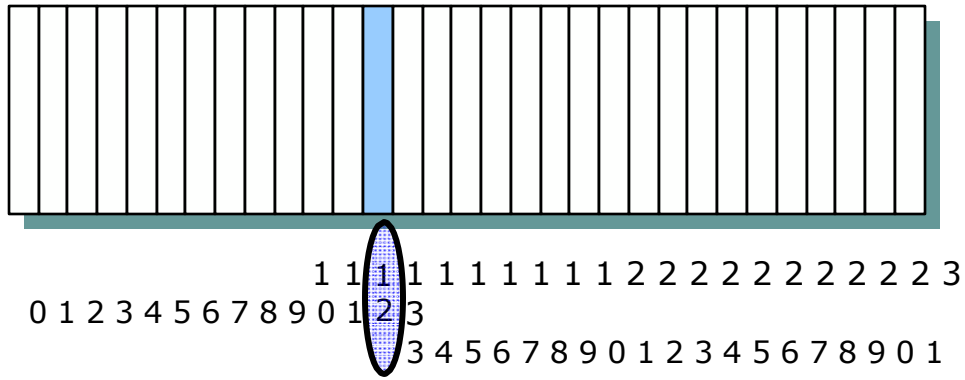
0 (12 mod 2)

Bloc no. 0 1 2 3 4 5 6 7



Set 0 Set 1 Set 2 Set 3

Block-frame address



Bloc no.

1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3

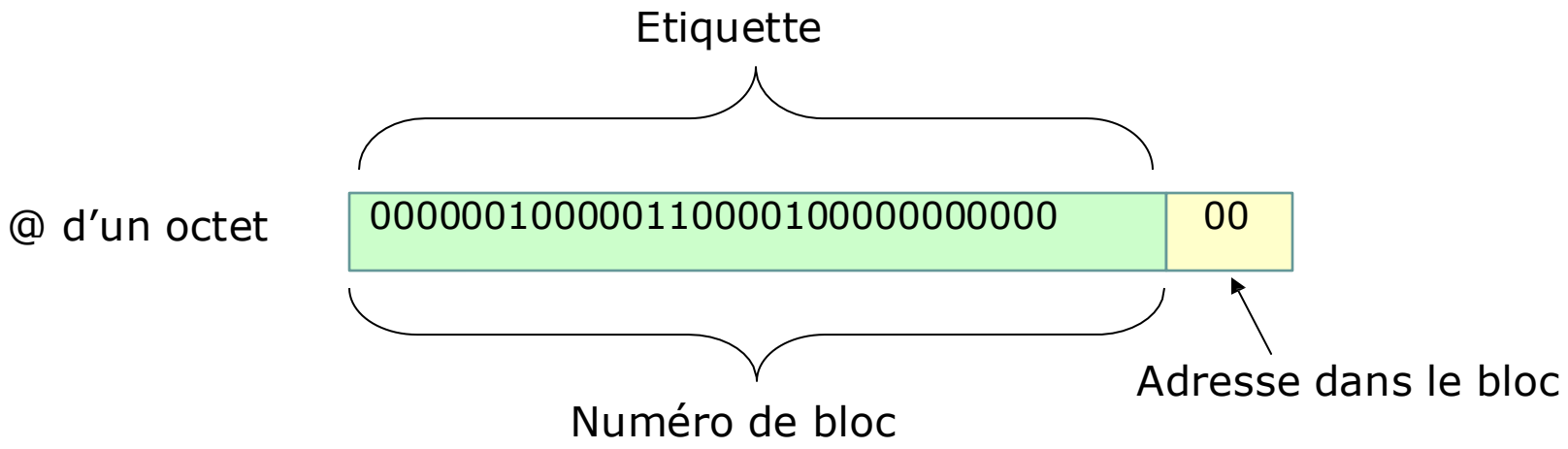
0 1 2 3 4 5 6 7 8 9 0 1 2 3

3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

❑ Gestion des mémoires

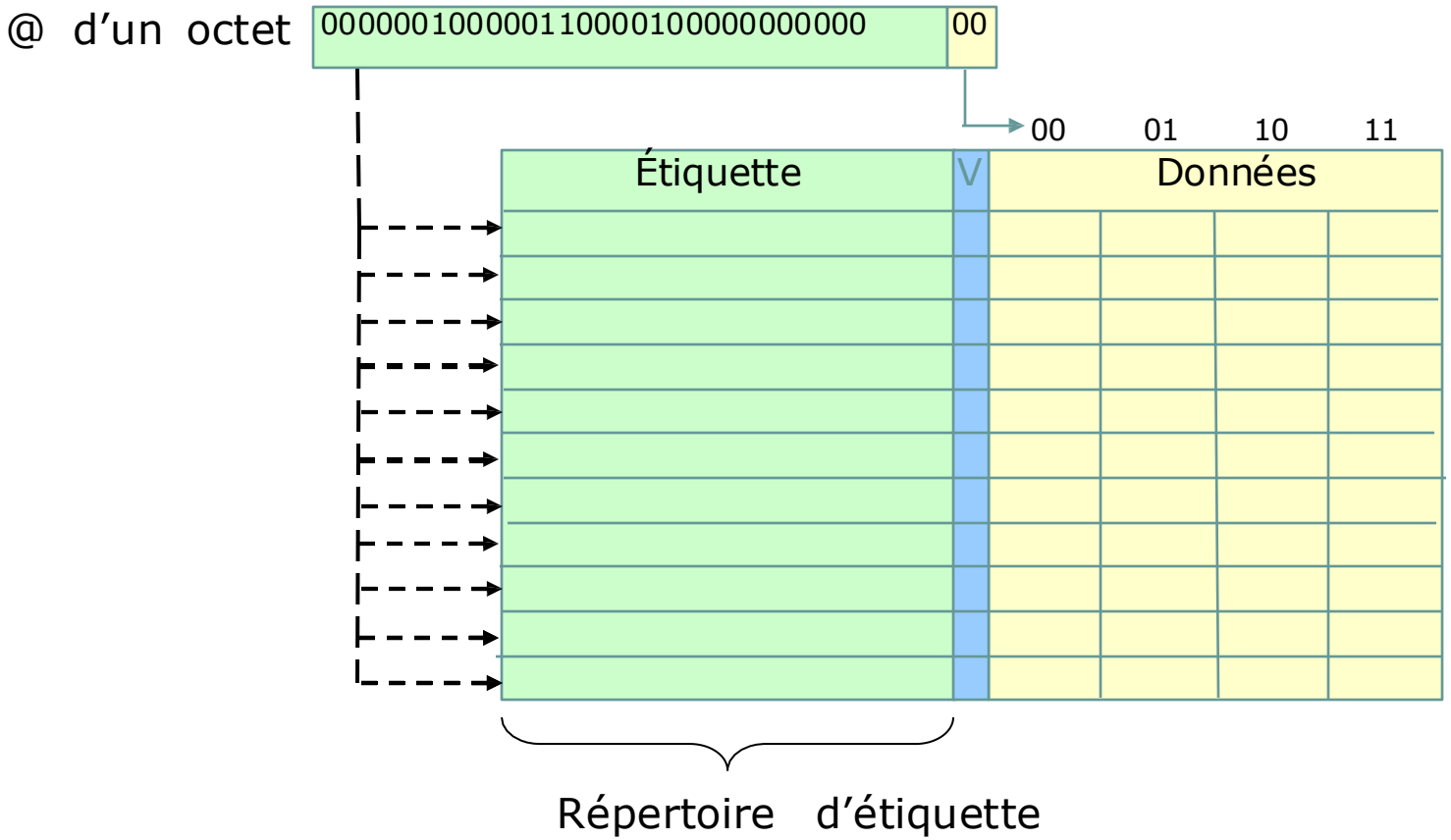
Cache associatif

- Le numéro de bloc est utilisé comme etiquette.
- Les étiquettes sont stockées dans un répertoire en même temps que les données
- Un bloc peut être placé n'importe où dans la mémoire cache.



❑ Gestion des mémoires

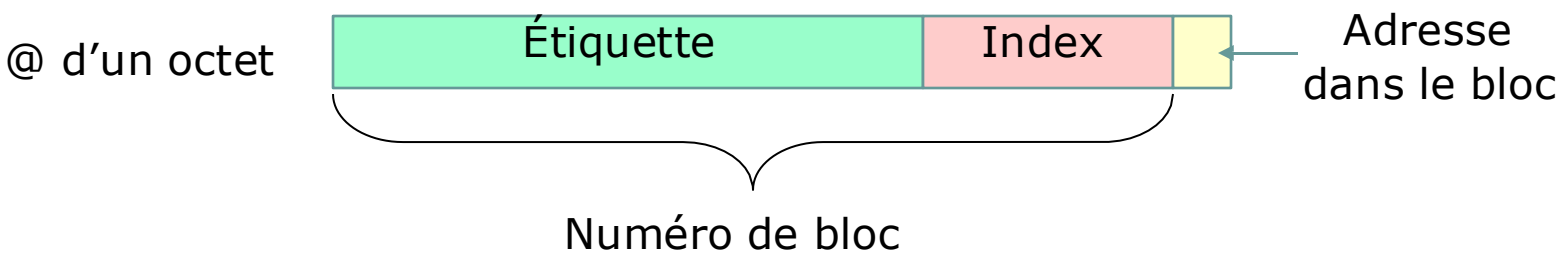
Cache associatif



❑ Gestion des mémoires

Cache à correspondance directe

Dans le cache à accès direct, le champ « numéro de bloc » est scindé en deux parties : l'étiquette et l'index. L'étiquette et les données correspondantes sont toujours enregistrées dans la rangée donnée par le champ index.

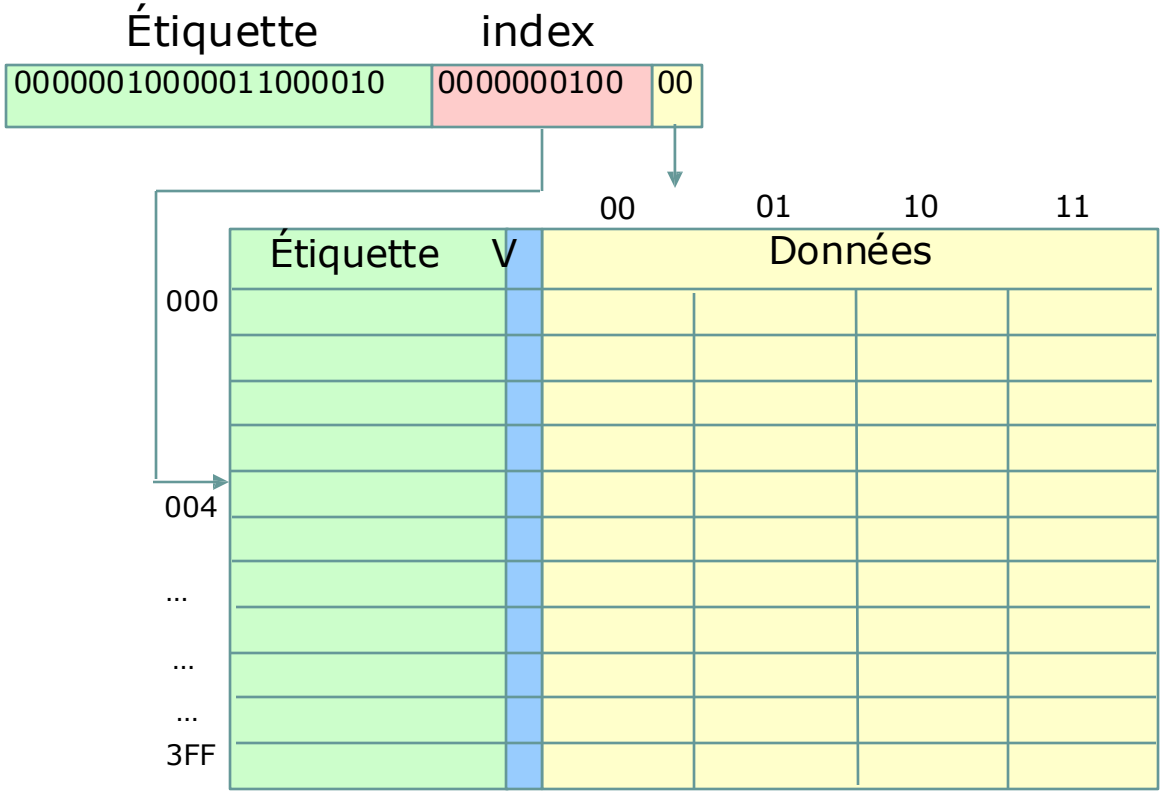


❑ Gestion des mémoires

Cache à correspondance directe

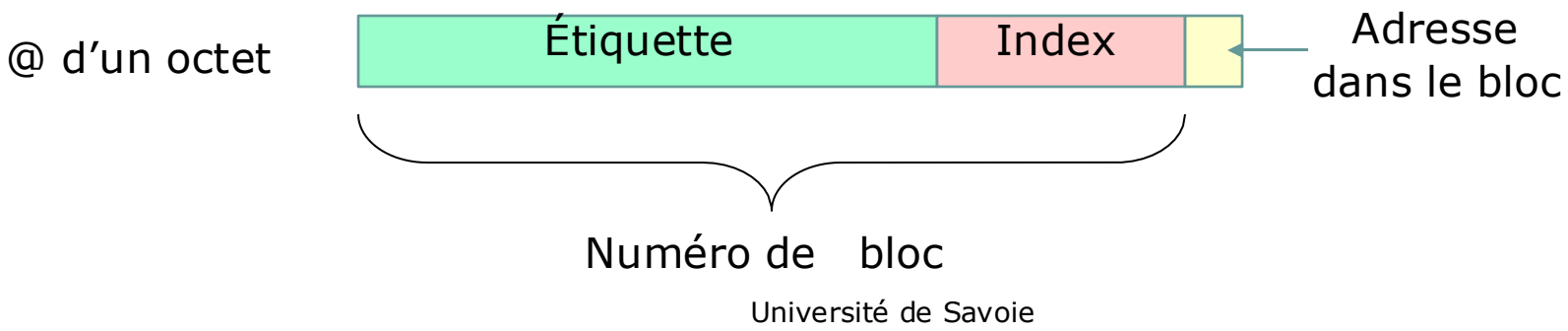
L'index donne directement la position dans le cache

$$\text{Nbr ligne de cache} = 2^{\text{Nbre bits d'index}}$$



Cache associatif par ensemble de blocs

Le cache associatif par ensemble de blocs est un compromis entre le cache purement associatif et le cache à correspondance directe. Le choix d'un ensemble est associatif. Cependant, chaque ensemble est géré comme dans le cache à correspondance direct.

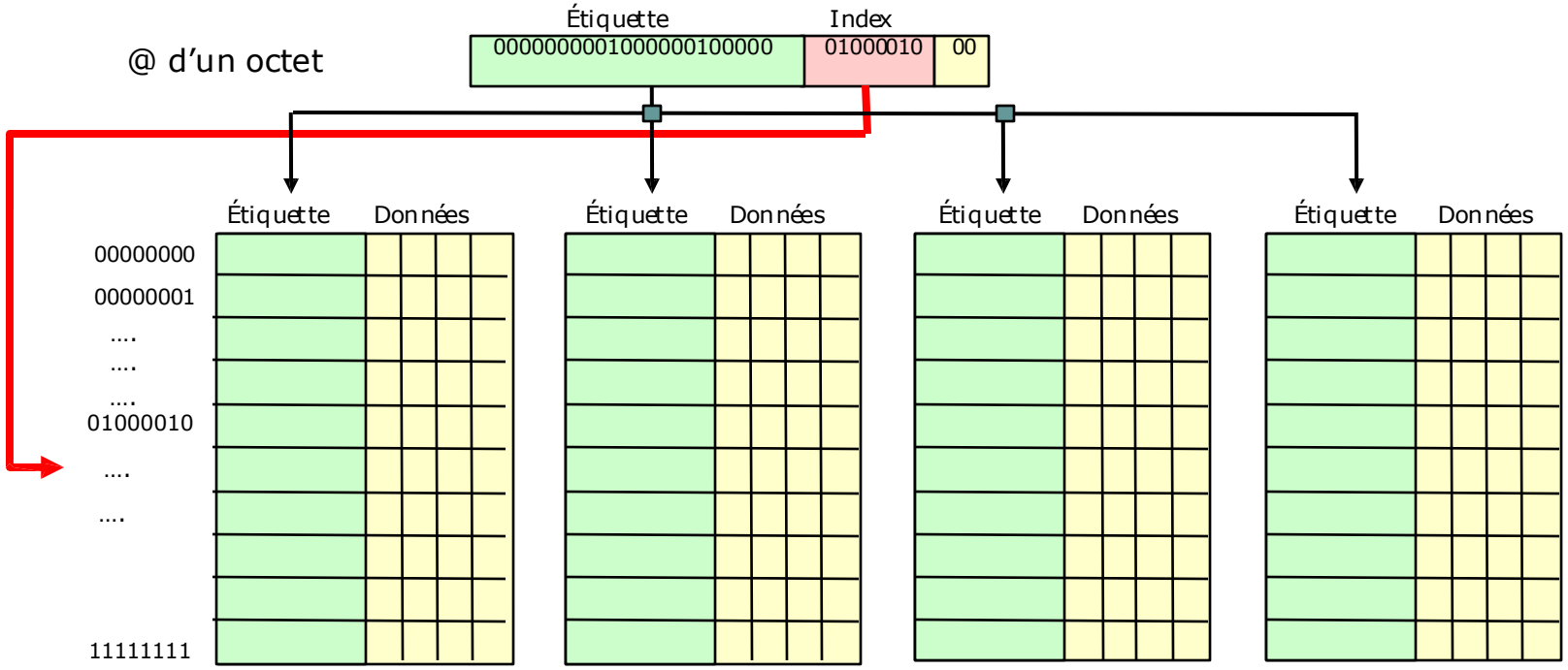


❑ Gestion des mémoires

Cache associatif par ensemble de blocs

Les données peuvent être rangées dans n'importe quelle ensemble (associatif)

Par contre l'index indique la ligne à laquelle on stocke la donnée (correspondance directe)



❑ Le microprocesseur

Les interruptions

Un système informatique n'est utile que s'il communique avec l'extérieur. L'objectif est de pouvoir prendre connaissance du fait que le périphérique sollicite le processeur. Cette sollicitation arrive de façon totalement asynchrone.

Deux modes sont possibles :

- Une méthode par scrutation (polling) permet d'interroger régulièrement les périphériques afin de savoir si une nouvelle donnée est présente.
- Une méthode par interruption permet au périphérique lui-même de faire signe au processeur de sa présence.

❑ Les interruptions

Scrutation Vs interruption

Scrutation (polling)

- Coûteux en temps (multiplier par le nombre de périphérique interroger)
- Implémentation : Appel classique à une fonction dans le programme

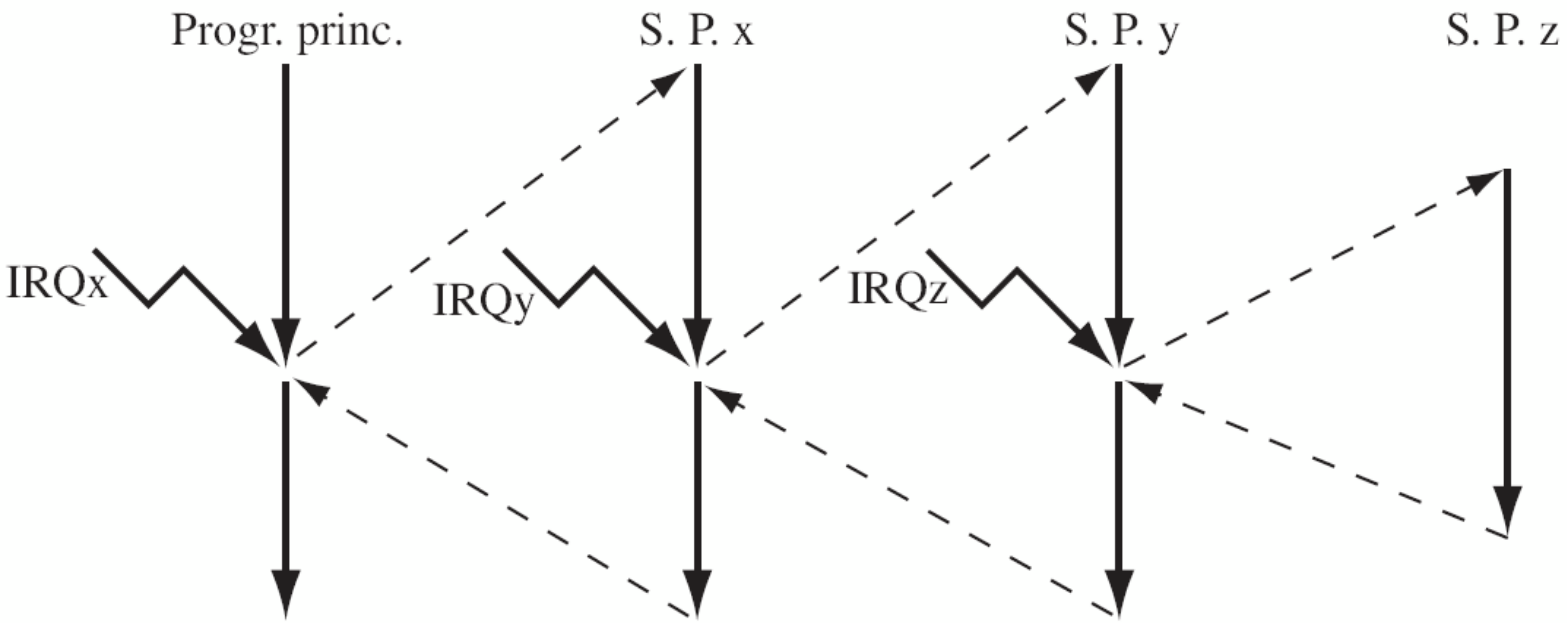
Interruption

- Demande à l'initiative du périphérique
- Prise en compte rapide de l'évènement
- Implémentation : Interruption asynchrone d'un programme puis retour au même endroit à la fin du traitement

❑ Les interruptions

Scrutation Vs interruption

Une **interruption** est un arrêt temporaire de l'exécution normale d’un programme informatique par le microprocesseur afin d'exécuter un autre programme (appelé routine d'interruption).



❑ Les interruptions

Scrutation Vs interruption

Types d’interruption

Interruption masquable

- Un masque d’interruption est un mot binaire de configuration du microprocesseur qui permet de choisir **(démasquer)** quels modules pourront interrompre le processeur parmi les interruptions disponibles.

Interruption non masquable

- Elles s ’exécutent quoi qu’il arrive, souvent avec une priorité élevé (ex : Reset)

❑ Les interruptions

Scrutation Vs interruption

Configuration

Un système peut accepter plusieurs sources d'interruption. Chacune est configurable par registre (registre d'interruption).

Méthode de configuration des interruptions

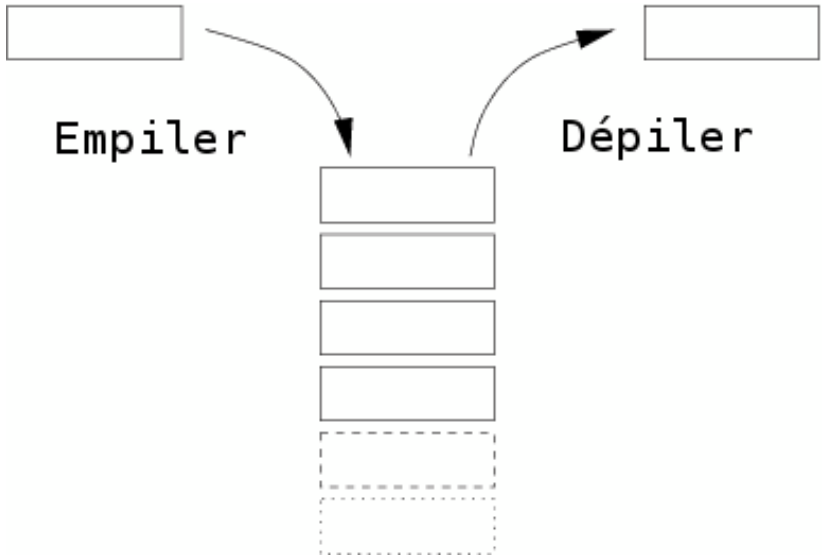
- Sélectionner les interruptions qui nous intéressent
- Valider les interruptions de façon globale
- Ecrire le/les sous programme d'interruption
- Définir les priorités entres interruptions

❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

La pile est une mémoire LIFO (Last In First Out) dans laquelle on stocke des variables temporaires (donnée ou adresse). Le haut de la pile est pointé par le registre SP (Stack Pointer)



❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Elle va servir à :

- **sauvegarder le contexte** l'environnement (adresse du programme et valeur des registres au moment de l'interruption).
- **restituer le contexte** à la fin de l'interruption

Une fonction d'interruption est noté spécifiquement

Exemple du PIC qui ne possède qu'un seul vecteur d'interruption :

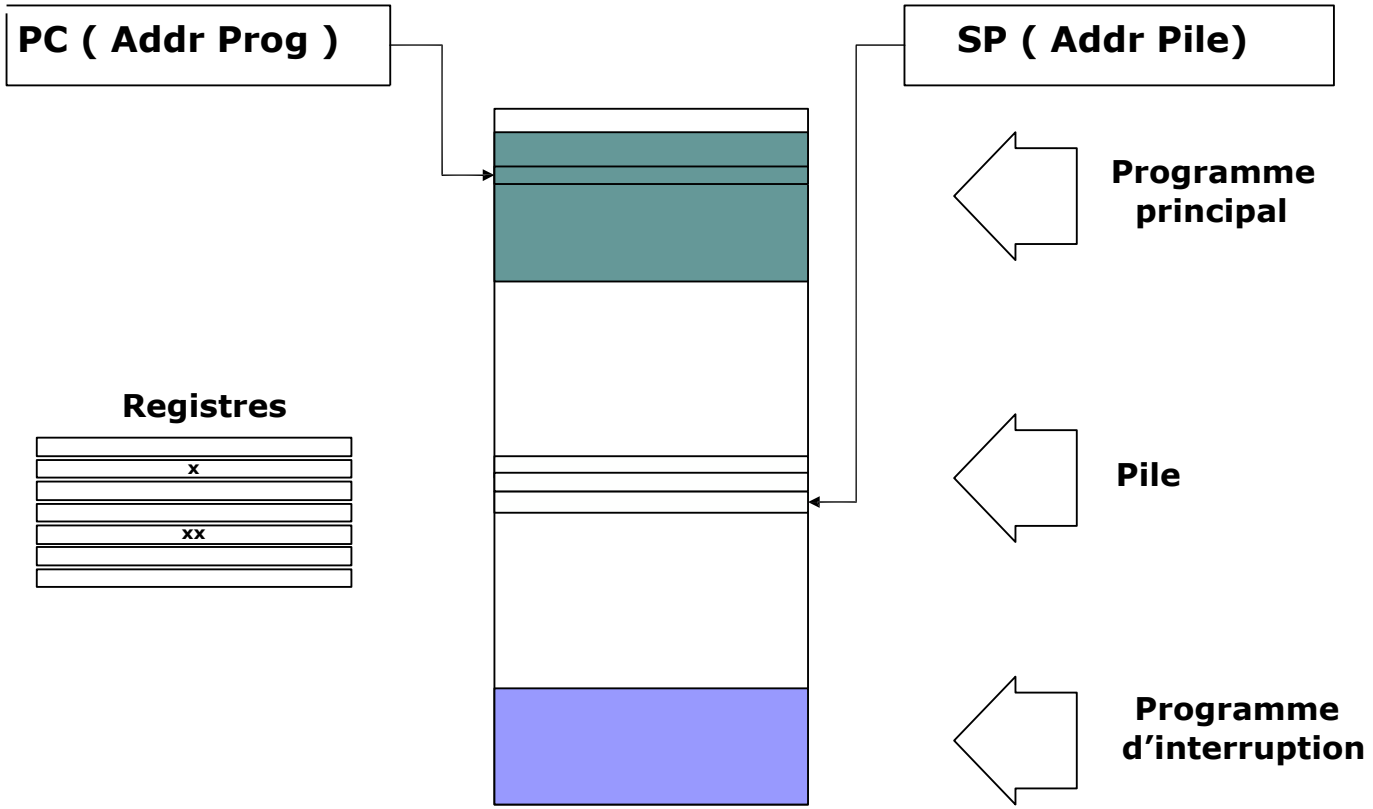
```
void interrupt() {
}
```

❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Avant interruption

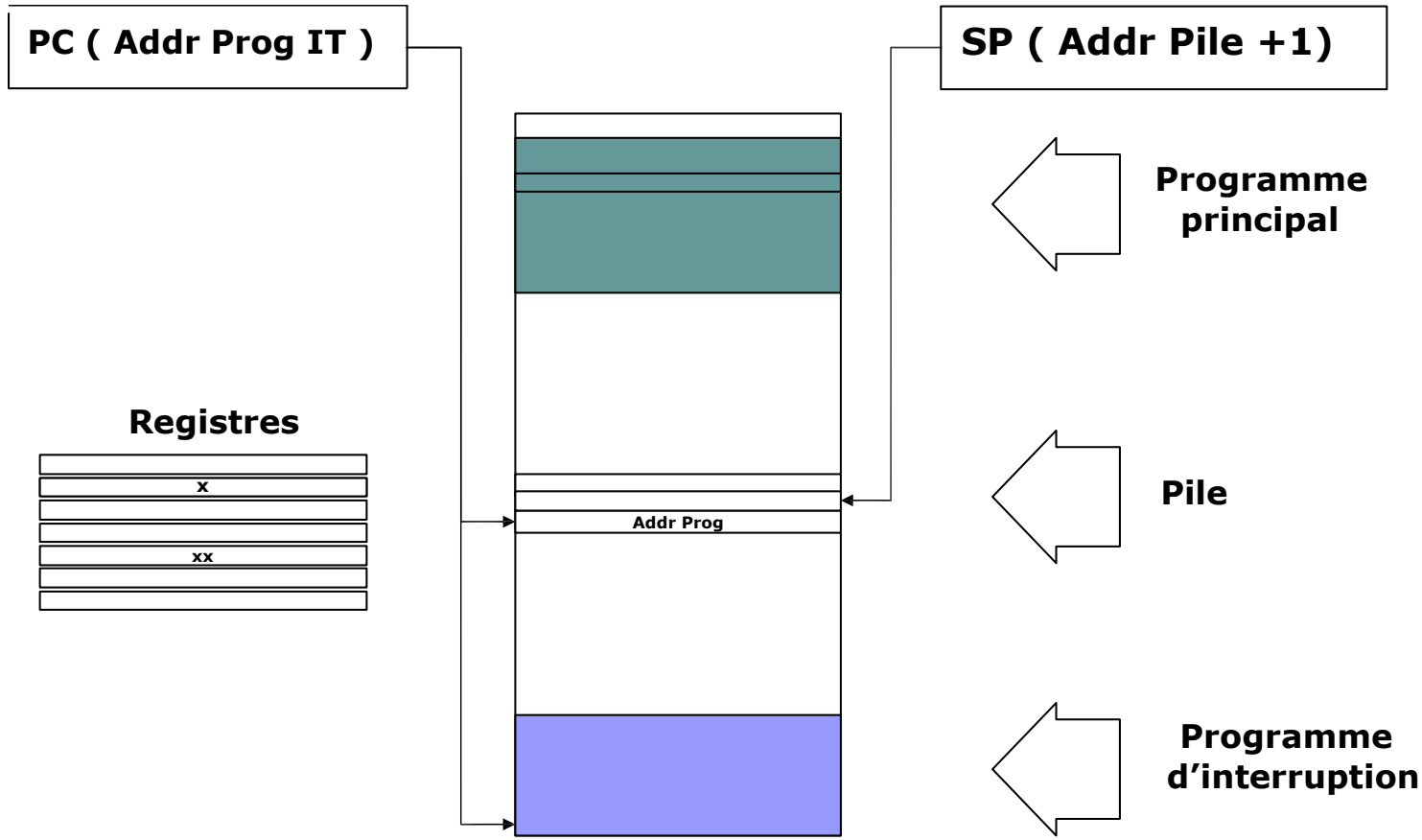


❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Arrivée d'une interruption

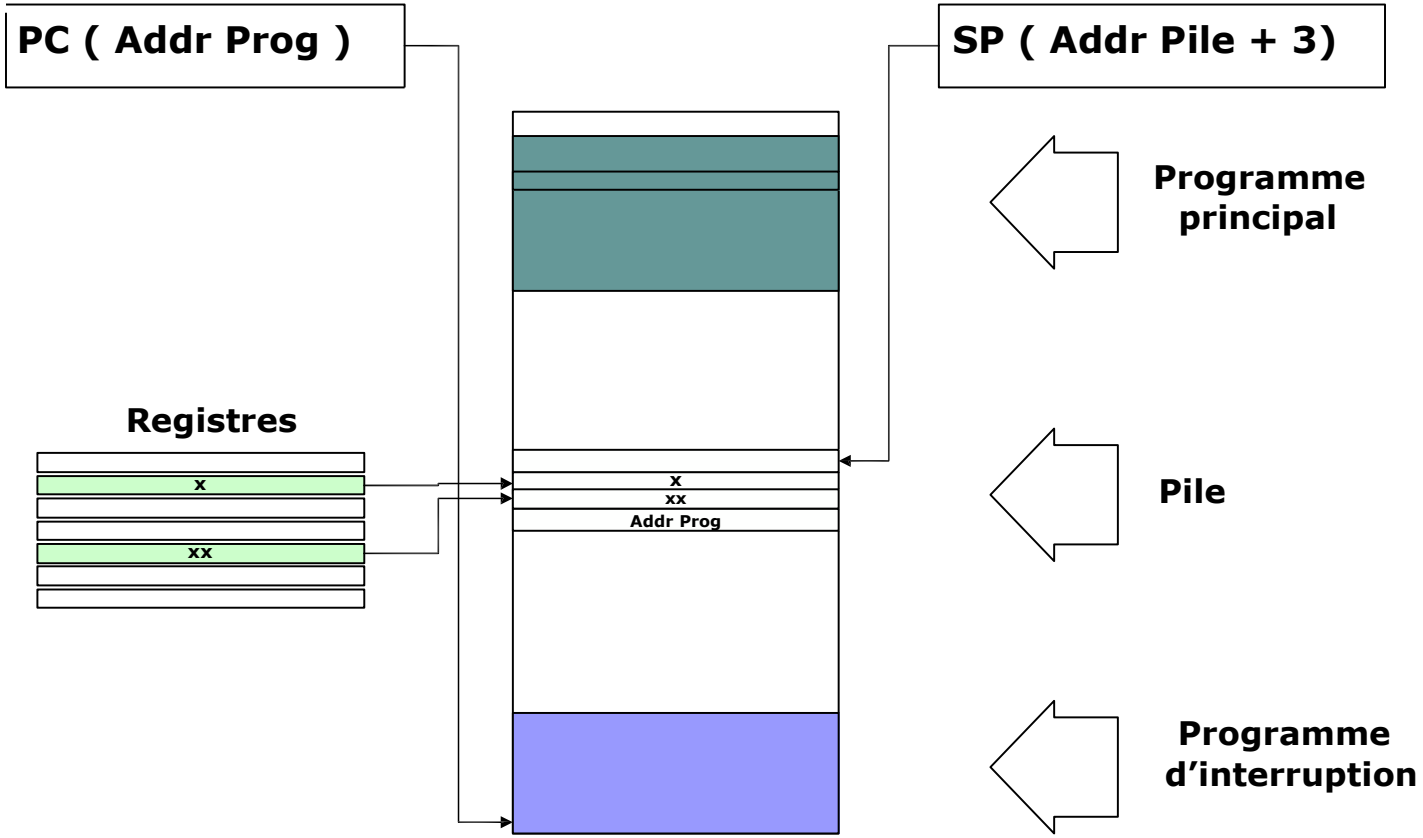


❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Arrivée d'une interruption: sauvegarde contexte

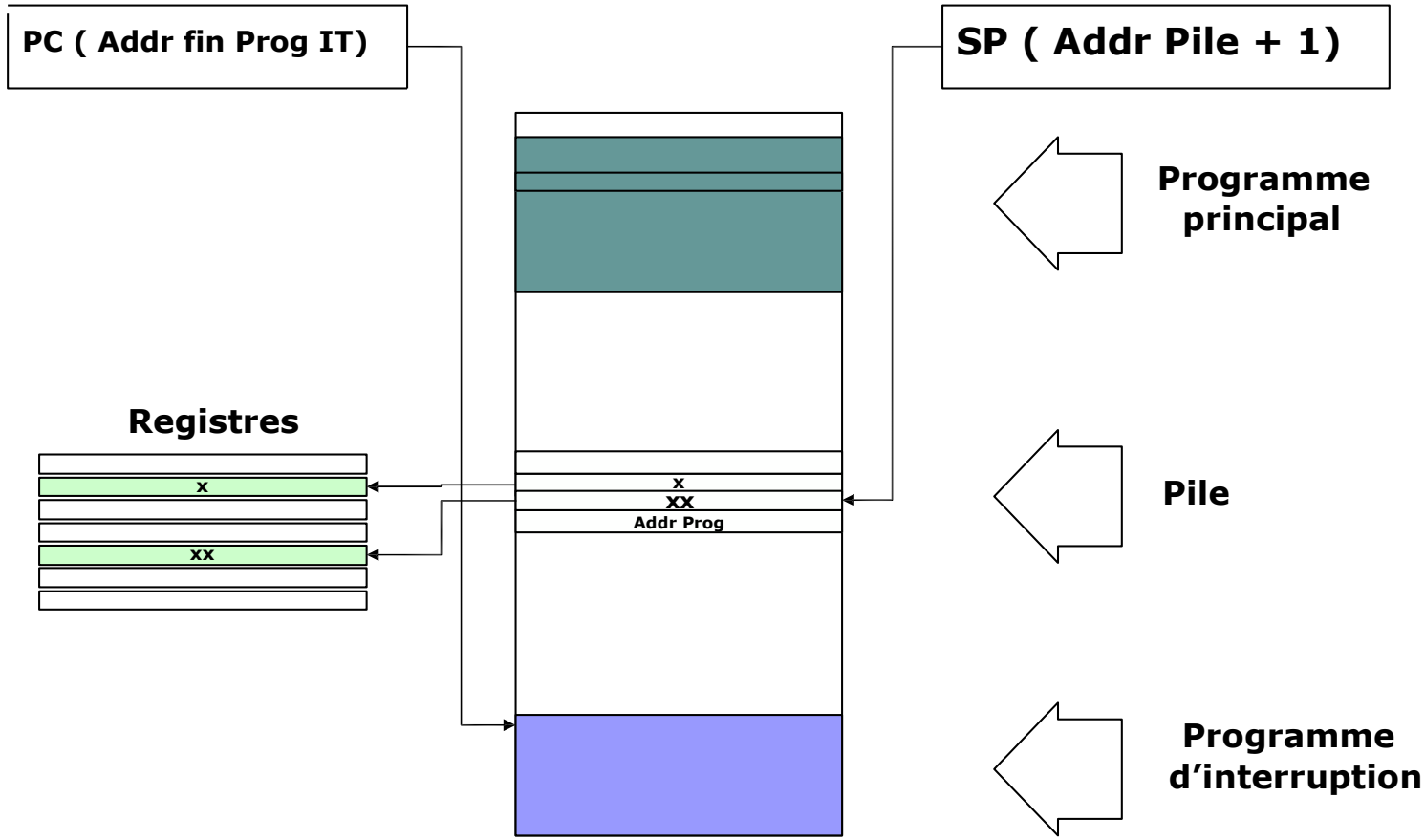


❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Fin d'une interruption: restitution contexte

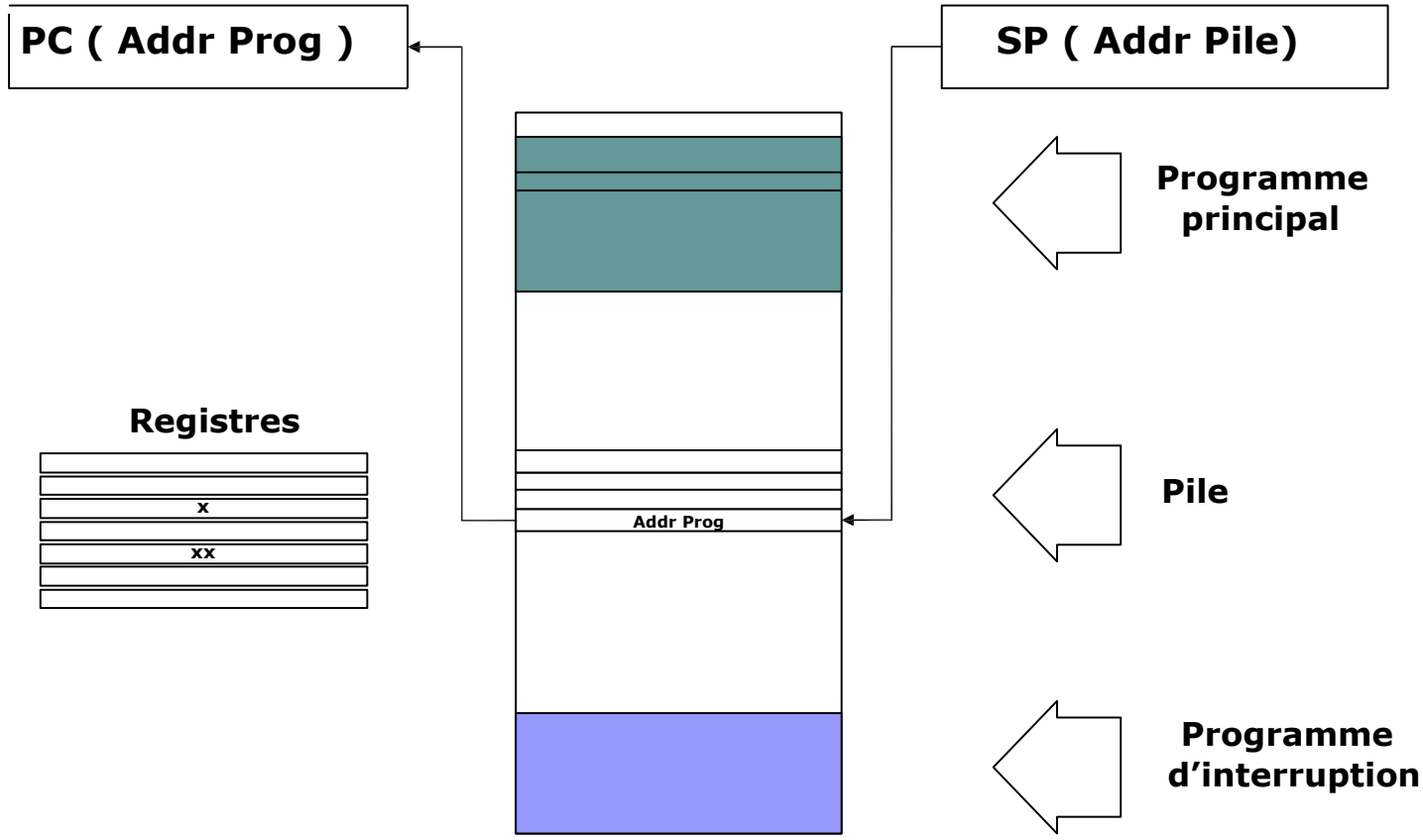


❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Fin d’une interruption

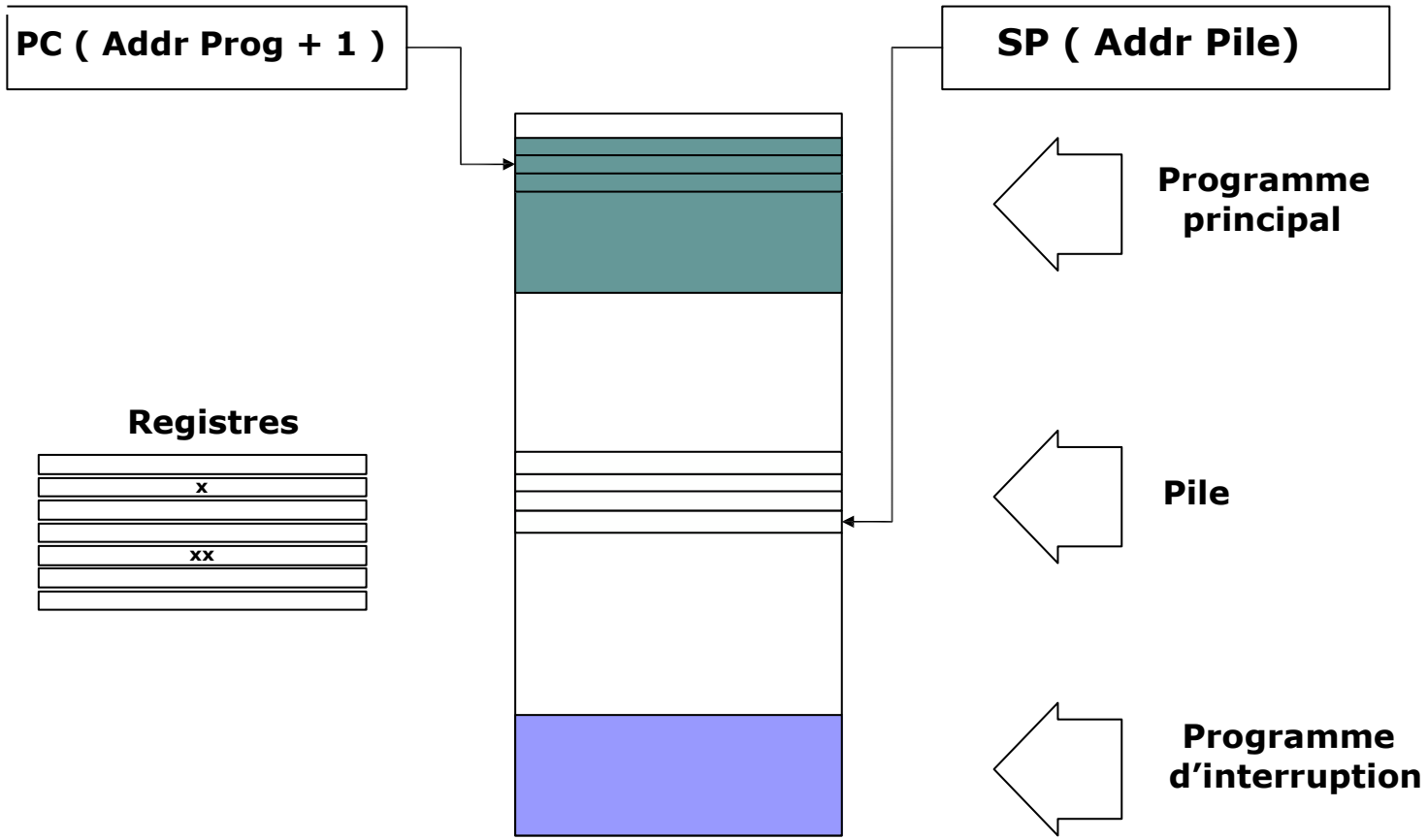


❑ Les interruptions

Scrutation Vs interruption

Le rôle de la pile

Retour au programme principal



❑ Les méthodes d’accès directs

Les modes de transfert

Mode PIO

- PIO : Programmed Input Output, ermet d’échanger des données avec la mémoire vive. Ces transferts sont gérés entièrement par le processeur.

Mode DMA

- La technique du DMA (Direct Memory Access) permet de désengorger le processeur en permettant à chacun des périphériques d'accéder directement à la mémoire.

❑ Les méthodes d’accès directs

Les modes de transfert

Mode PIO

Des commandes gérées directement par le processeur permette la gestion du transfert. Toutefois, de gros transferts de données peuvent rapidement imposer une grosse charge de travail au processeur et ralentir l'ensemble du système. Il existe 5 modes PIO définissant le taux de transfert maximal.

Mode PIO	Débit (Mo/s)
Mode 0	3.3
Mode 1	5.2
Mode 2	8.3
Mode 3	11.1
Mode 4	16.7

❑ Les méthodes d'accès directs

Les modes de transfert

Mode DMA

- ✓ La technique du DMA (Direct Memory Access) permet de désengorger le processeur en permettant à chacun des périphériques d'accéder directement à la mémoire. Deux types de DMA existent:
  - Le DMA dit "single word" permet de transmettre un mot simple à chaque session de transfert,
  - Le DMA dit "multi-word" permet de transmettre successivement plusieurs mots à chaque session de transfert.
- ✓ Le tableau suivant liste les différents modes DMA et les taux de transfert associés :

Mode DMA	**Débit (Mo/s)**
0 (Single word)	2.1
1 (Single word)	4.2
2 (Single word)	8.3
0 (Multiword)	4.2
1 (Multiword)	13.3
2 (Multiword)	16.7

❑ Les méthodes d’accès directs

Les modes de transfert

Tableaux comparatif

Caractéristique	Mode PIO	Mode DMA
Implication du processeur	Gère directement chaque transfert	Configure le transfert, puis se libère
Performance	Plus lente, consomme des cycles CPU	Plus rapide, libère le CPU
Complexité matérielle	Simple	Plus complexe
Applications	Systèmes simples, périphériques lents	Systèmes modernes, périphériques rapides

❑ Les méthodes d’accès directs

Les modes de transfert

Mode ultra DMA

L’idée est d’augmenter la fréquence du signal d’horloge pour augmenter la rapidité. Toutefois sur une interface où les données sont envoyées en parallèle l'augmentation de la fréquence pose des problèmes d'interférence électromagnétiques. Deux solutions ont été apporté qui vont être en étroite relation :

- Augmentation de la fréquence : Utilisation des front montants et descendant.
- Amélioration du connecteur ATA (a partir de l'Ultra DMA mode 4 un nouveau type de nappe a été introduit afin de limiter les interférences ; il s'agit d'une nappe ajoutant 40 fils de masse entrelacés avec les fils de données.
- Apparition du CRC

❑ Les méthodes d'accès directs

Les modes de transfert

Mode ultra DMA

Fonctionnement :

- La fréquence de transfert augmente tant que les données transmises se font sans erreur.
- Lorsque qu'une erreur est rencontrée, le transfert passe dans un mode Ultra DMA inférieur (voire sans Ultra DMA).

Mode Ultra DMA	Débit (Mo/s)
UDMA 0	16.7
UDMA 1	25.0
UDMA 2 (Ultra-ATA/33)	33.3
UDMA 3	44.4
UDMA 4 (Ultra-ATA/66)	66.7
UDMA 5 (Ultra-ATA/100)	100
UDMA 6 (Ultra-ATA/133)	133

❑ Les méthodes d'accès directs

Les vitesses de transfert

Architecture	Name	Time	Transfer Speed	Note
Serial	Serial ATA[2]	Mid-2007	600 MB/S (generation 3)	CRC / Package transfer (bits of data together)
		Mid-2004	300 MB/S (generation 2)	
		Fall-2002	150 MB/S (generation 1)	
Parallel	Ultra DMA	Current mainstream	16.7/25.0/33.3/44.4/66.7 100.0/133.0 MB/S	CRC / Multi-word
	DMA	1990	4.2/13.3/16.7MB/S	Multi-word
		1980	2.1/4.2/8.3 MB/S	Single word

Les vitesses de transfert (mode DMA ou Ultra DMA restent donc toujours en étroite relation avec l'architecture utilisée (ATA, Serial ATA, …)

❑ Les méthodes d’accès directs

Les canaux DMA

Un ordinateur de type PC possède 8 canaux DMA.

Les canaux DMA sont généralement assignés comme suit :

DMA0 - System Use : Memory (DRAM) Refresh
DMA1 - Libre
DMA2 - contrôleur de disquettes
DMA3 - port parallèle
DMA4 - contrôleur d'accès direct à la mémoire (renvoi vers DMA0)
DMA5 - (carte son)/ libre
DMA6 - (SCSI)/ libre
DMA7 - disponible

❑ Les méthodes d’accès directs

Les canaux DMA

Canal DMA	Utilisation	Description
DMA0	Utilisation système : Rafraîchissement de la mémoire	Réservé pour maintenir les données dans la DRAM (rafraîchissement).
DMA1	Libre	Disponible pour des périphériques ou applications spécifiques.
DMA2	Contrôleur de disquettes	Gère les transferts entre la mémoire et les disquettes.
DMA3	Port parallèle	Utilisé pour les transferts via le port parallèle (imprimantes, etc.).
DMA4	Contrôleur d'accès direct à la mémoire	Agit comme un renvoi vers **DMA0** pour coordonner d'autres canaux.
DMA5	Carte son / Libre	Généralement utilisé pour les transferts audio ou laissé libre.
DMA6	SCSI / Libre	Parfois réservé aux contrôleurs SCSI pour des transferts rapides.
DMA7	Disponible	Canal libre pour des périphériques ou usages spécifiques.