ARCHITECTURE DES SYSTEMES A BASE DE MICROPROCESSEUR

PLAN

- Introduction
- Architecture programmable
- Processeur : modèle et caractéristiques
- Machine câblée versus machine micro programmée
- Présentation du système de mémorisation

PLAN

- Introduction
- Architecture programmable
- Processeur : modèle et caractéristiques
- Machine câblée versus machine micro programmée
- Présentation du système de mémorisation

Introduction

- 3 Grandes classes de dispositifs de traitement peuvent être identifier:
  - **Systèmes combinatoire** : les sorties à l'instant t ne dépendent que des entrées à ce même instant (exemple un système de calcul)

    St = F(Et)

  - **Systèmes séquentiels** : les sorties à l'instant t dépendent des entrées à ce même instant mais, aussi de l'historique de la machine (état interne Xt). Exemple: un automate gérant un carrefour.

    St = F(Et,Xt)

    Xt+1 = G(Et,Xt)

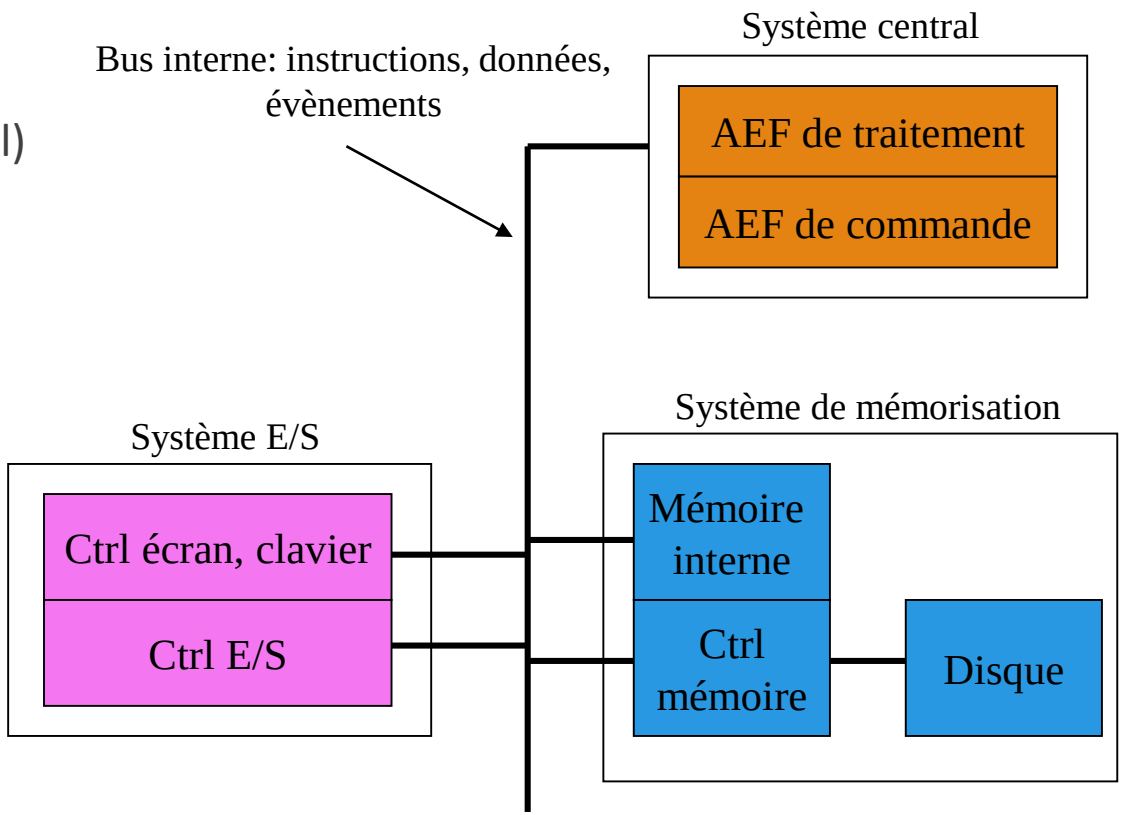
Introduction

- **Systèmes à microprocesseur** : le système peut être modélisé par une double équation vu précédemment (système séquentiel)
  - ***Automate de traitement (chemin de données, data path):*** il est chargé du calcul et il émet des comptes rendus vers l'automate de commande.
    - St = F(Et,Xt) les résultats après calculs dépendent des données d'entrées et des instructions à exécuter.
    - Xt+1=G(Et,Xt) certains résultats sont mémorisés pour une utilisation ultérieure.
  - ***Automate de commande (IP):*** il est chargé du contrôle et de la commande de l'automate de traitement et de l'environnement
    - S(t)=(Et,Xt); Xt+1=G(Et,Xt) les sorties et la prochaine instruction à exécuter dépendent des entrées (compte rendus et instruction venant de l'environnement) et des instructions précédemment exécutées.

Introduction

Architecture d'un ordinateur:

- Microprocesseur (système central)
- Mémoire (système de mémorisation)
- Entrées/Sorties (système E/S).



PLAN

Introduction

Architecture programmable

Processeur : modèle et caractéristiques

Machine câblée versus machine micro programmée

Présentation du système de mémorisation

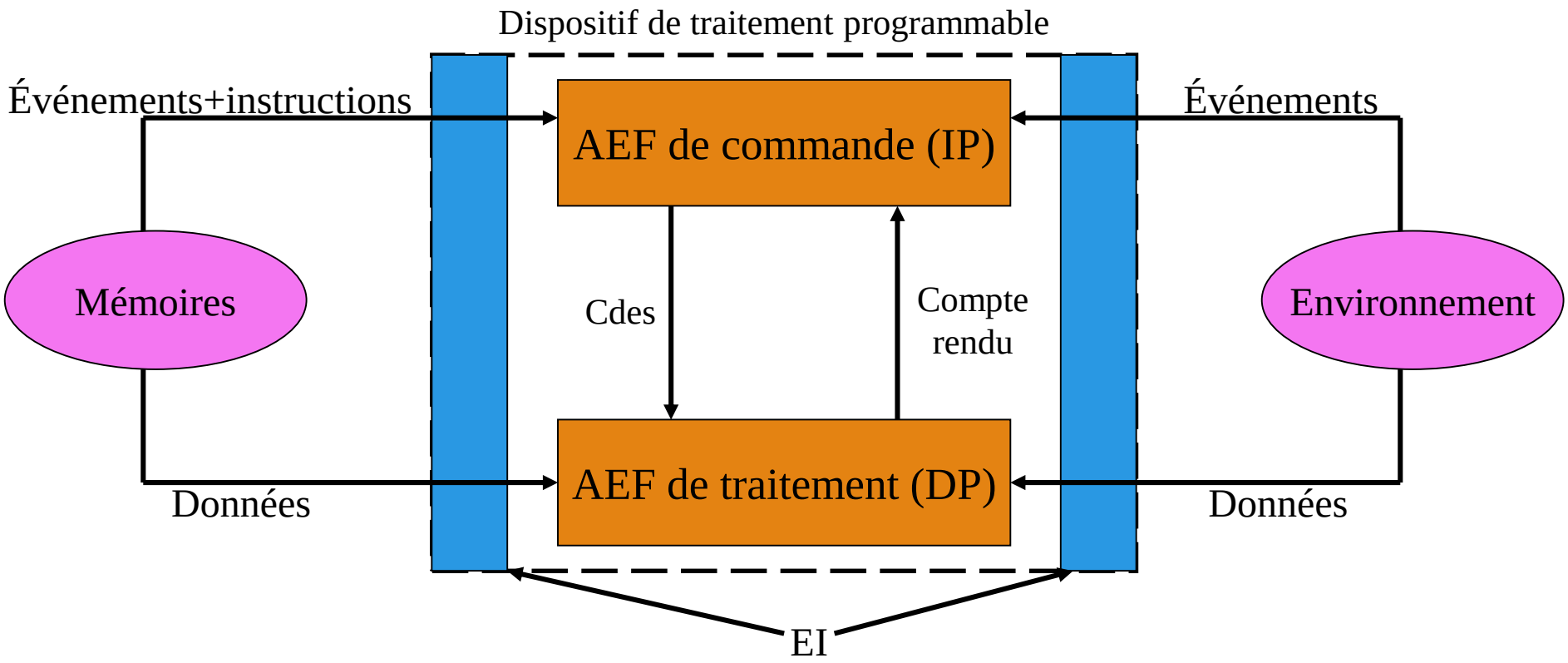
Architecture programmable

Modélisation: Les unités fonctionnelles

- Processeur d'instructions: ***IP (ou AEF de commande)***
  - Automate qui interprète les instructions et les passe au DP
- Processeur de données: ***DP (ou AEF de traitement)***
  - Automate qui modifie ou transforme les données et qui peut mémoriser dans des registres des données provenant ou devant aller vers la mémoire.
- Mémoire d'instructions: ***IM***
  - Stockage des instructions (programme)
- Mémoire de données:
  - Stockage des données à traiter.
- Interface externe: ***EI***
  - Contrôle les accès aux périphériques (environnement) aussi bien de type mémoire que de type E/S (disque, imprimante…) à travers des protocoles de communication.

Architecture programmable

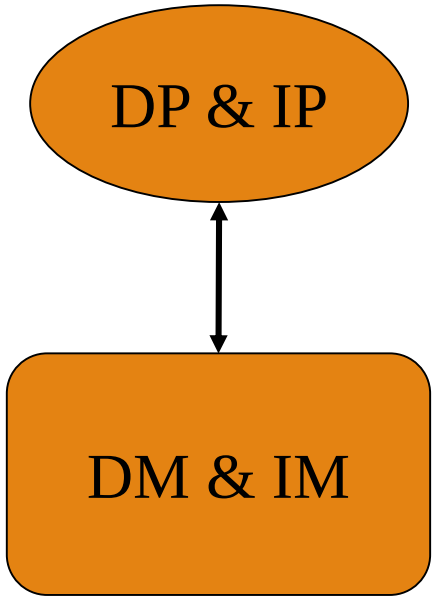
Modélisation: les unités fonctionnelles (suite)



Architecture programmable

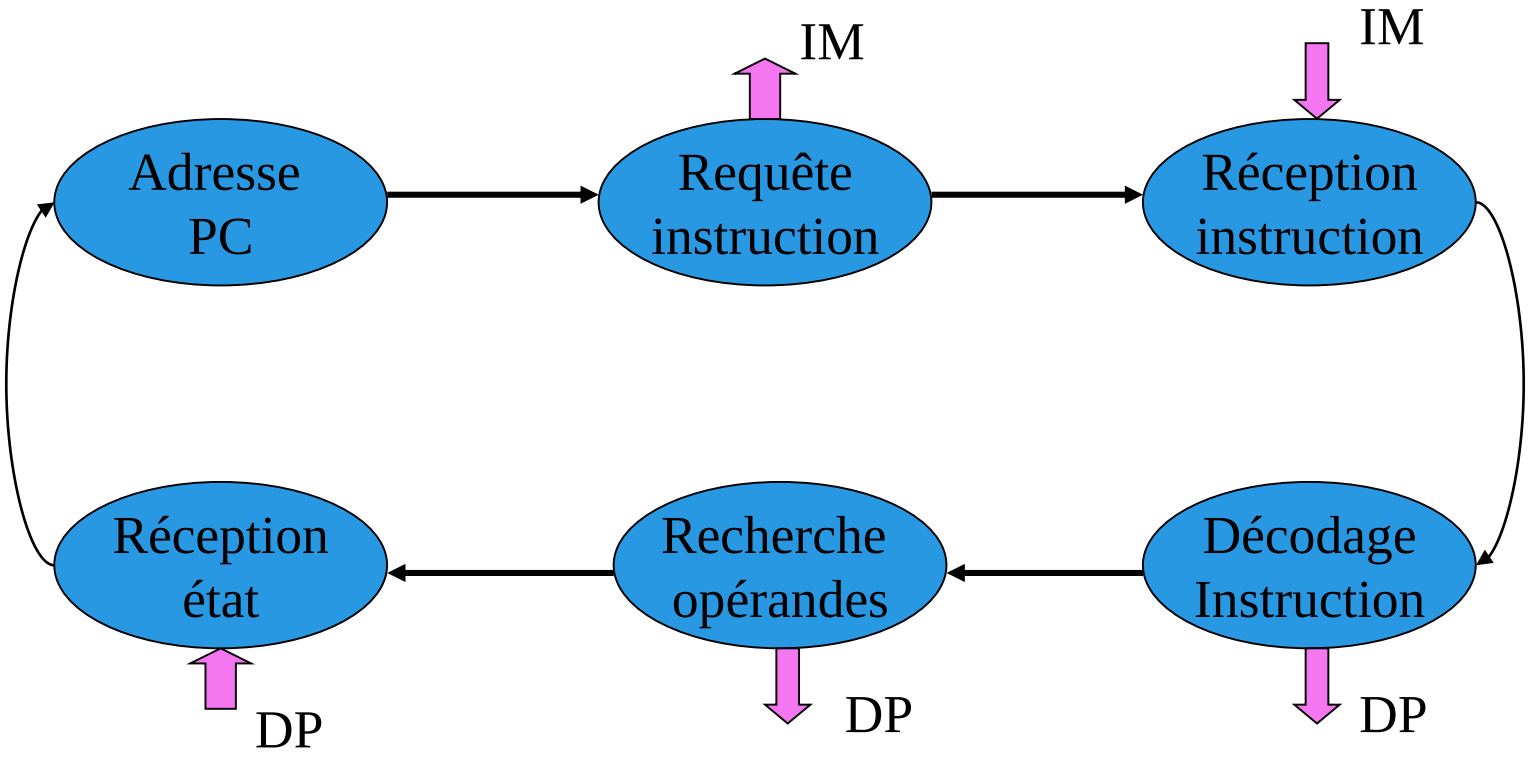
Modélisation: le processeur ***Von Neumann***

- Les calculs sur les adresses et les données sont réalisés sur les mêmes opérateurs:
  - Pas d'opérateur dédié au calcul du registre PC
- Il n'existe qu'une seule mémoire qui contient à la fois le programme (instructions) et les données à traiter.
- Fonctionnement séquentiel.



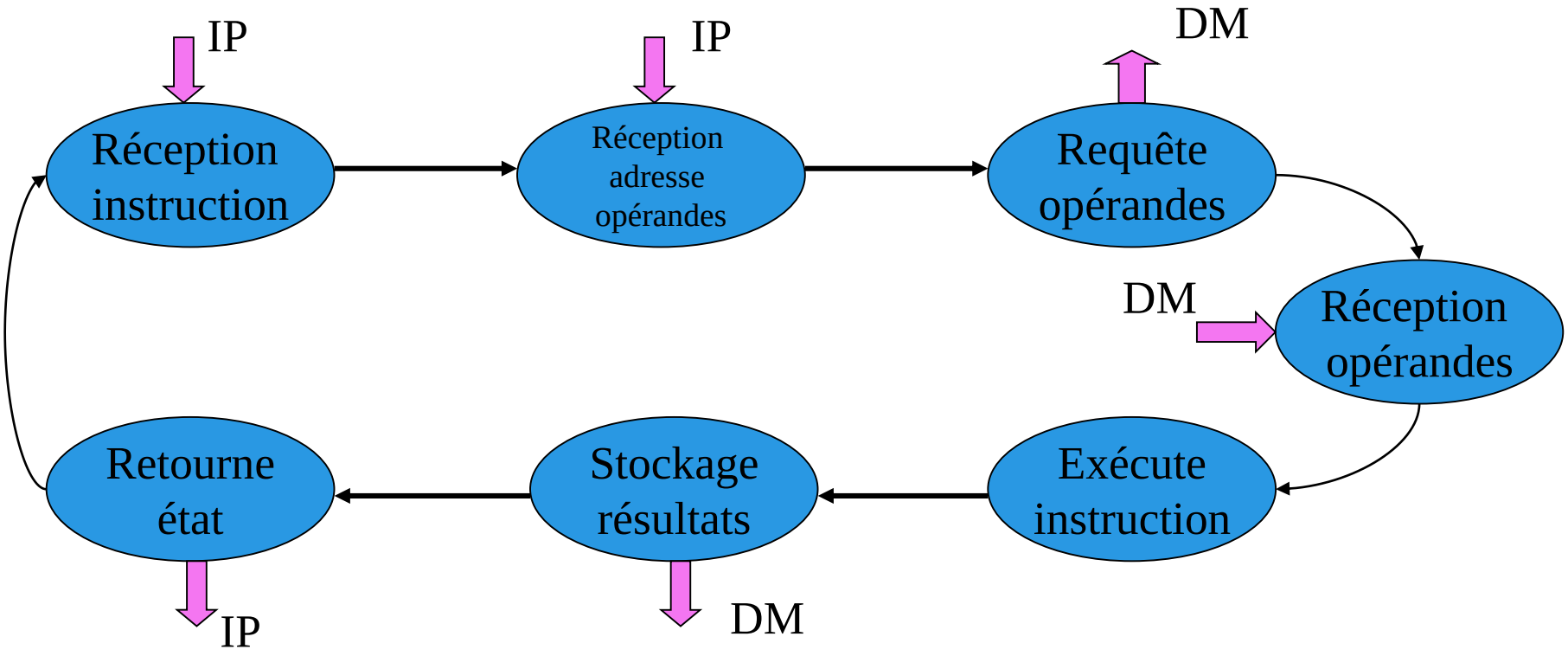
Architecture programmable

Diagramme d'états de l'AEF de commande (IP)



Architecture programmable

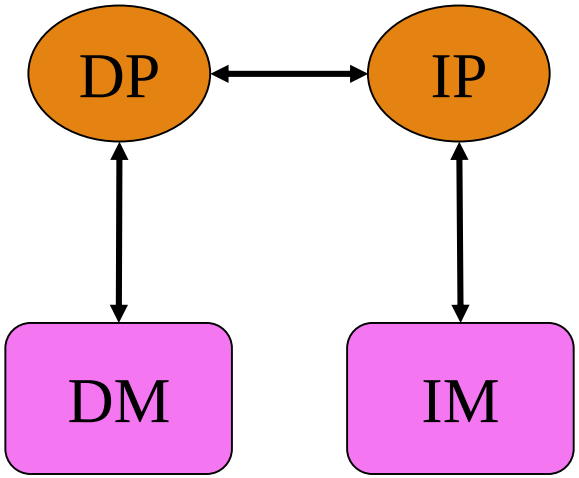
Diagramme d'états de l'AEF de traitement (DP)



Architecture programmable

Modélisation: le modèle ***Harvard***

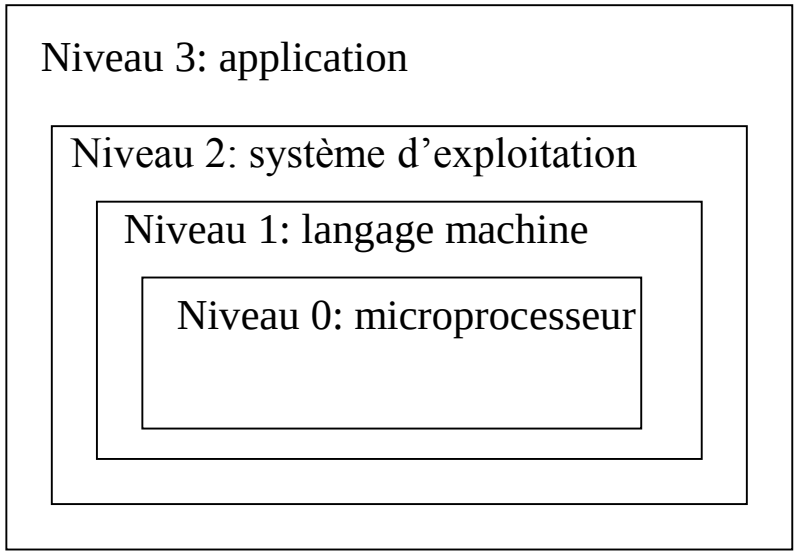
- Mise en place d'une unité spécifique de calcul des adresses.
- Séparation des mémoires de données et de programme (instructions).
- Séparation des bus de données et d'adresses
- Permet:
  - l'accès parallèle à une données et à une instruction
  - Le travail en parallèle des unités



Architecture programmable

Programmation d'un microprocesseur

- Nous avons une représentation hiérarchique des ordinateurs
- Avoir des langages applicatifs les plus abstraits possibles



Le passage de N1 à N0 => ***cycle Von Neumann***

-Li: Lecture Instruction

-DI: Décodage instruction et recherche des opérandes

-EX: Exécution de l'instruction

-RR: Rangement des résultats

-Rq: le cycle Von Neumann peut être décomposé différemment

Architecture programmable

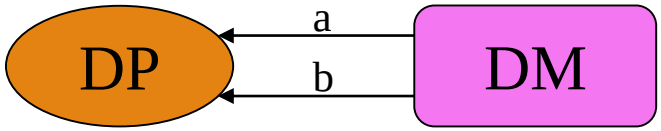
La mémoire

- Accès à la mémoire pour y lire des instructions
- Accès à la mémoire pour y lire des données
- Utilisation de différents formats: 8/16/32/64.. Bits
- Utilisation de différents modes d'adressage: direct, indirect, indexé, post/pré incrémenté etc..
- Utilisation de différents types de mémoire: SRAM, SDRAM, DDR-RAM, Flash etc..

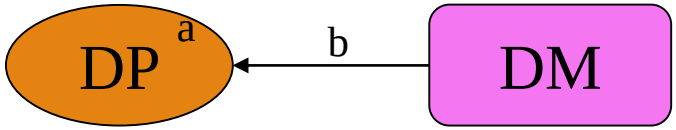
Architecture programmable

Accès à la mémoire de données: c = add (a,b)

- Architecture mémoire/mémoire
  - Add *a,*b,@c



- Architecture mémoire/registre: les architectures CISC (Complex Instruction Set Computer)
  - LD *a,r1
  - Add r1,*b,@c



- Architecture registre/registre ou LOAD/STORE : les architectures RISC (Reduced Instruction Set Computer
  - LD *a,r1
  - LD *b,r2
  - Add r1,r2,r3
  - ST r3,@c

Architecture programmable

Exemple:

- Soit le programme applicatif écrit en langage L2 (pas d'OS).
- Soit le langage L1 composé des instructions suivantes
  - Load Ri,*adrj: chargement dans le registre Ri de la valeur contenue à l'adresse mémoire adrj
  - Add Ri,*adrj: additionne le contenu du registre Ri et la valeur contenue à l'adresse mémoire adrj. Le résultat est dans Ri
  - Store adrj,Ri: stockage du contenu du registre Ri à l'adresse mémoire adrj.
- Rq: Les registres sont ceux contenus dans le DP et la mémoire est la mémoire DM.

Début

A = B + c;

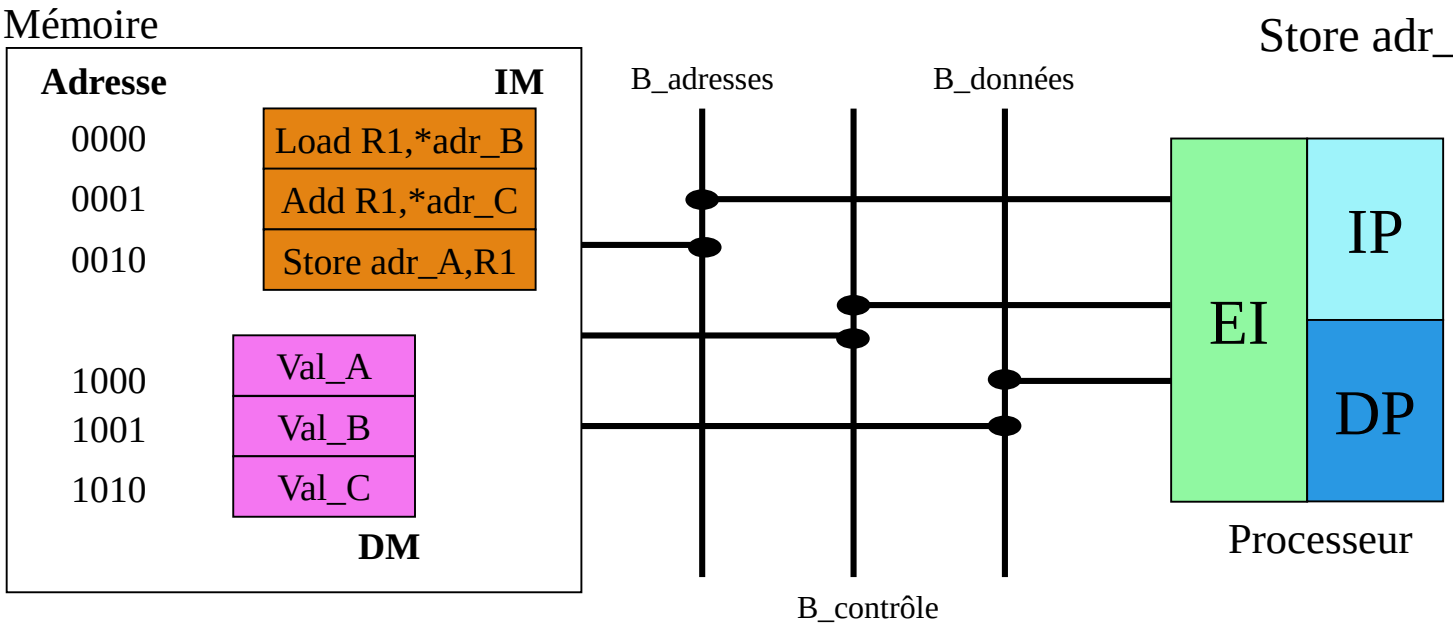
Fin

Architecture programmable

Exemple (suite)

- Le programme en langage L1 peut s'écrire:

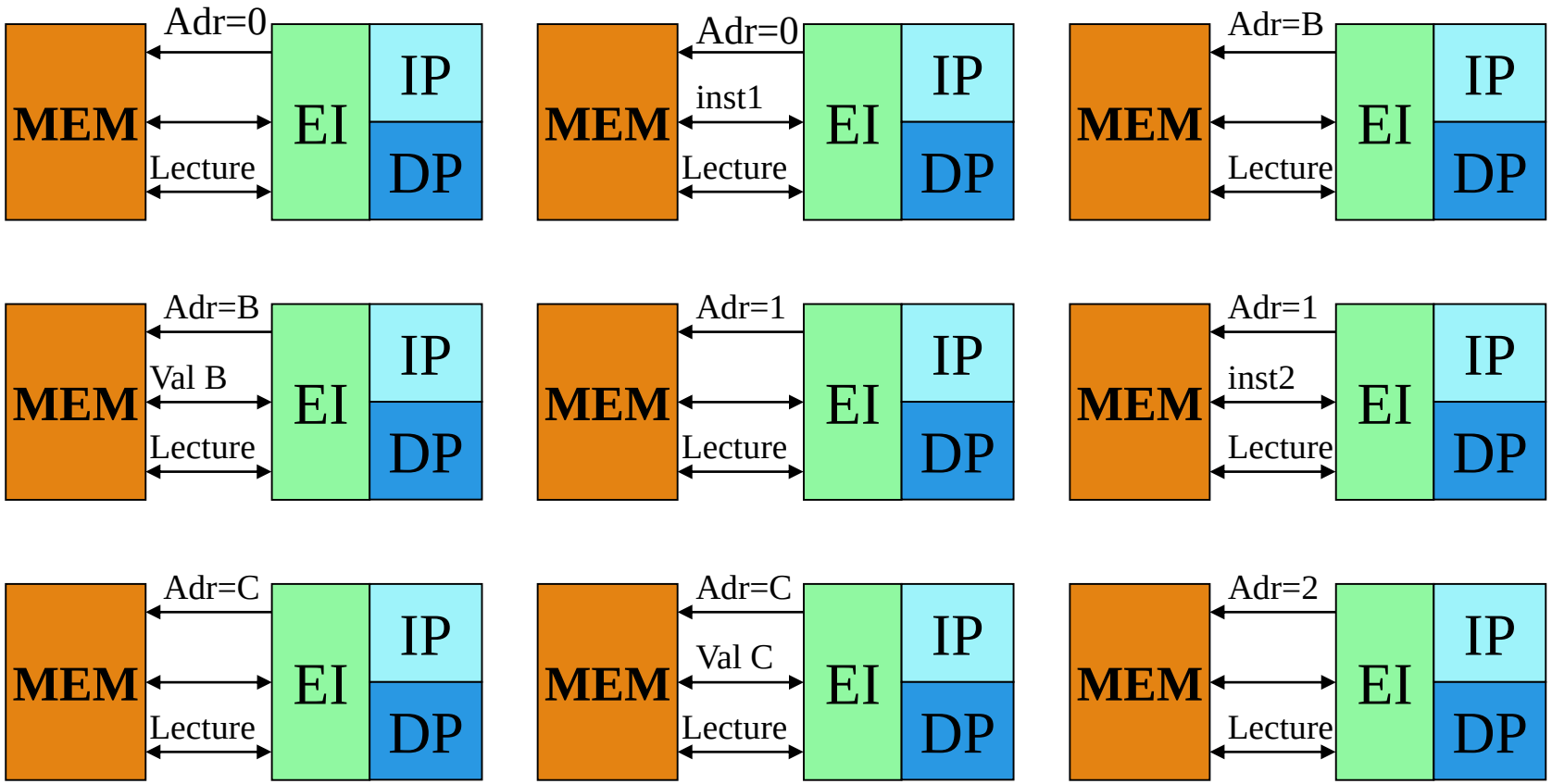
Load R1,*adr_B
Add R1,*adr_C
Store adr_A,R1



Rq: la mémoire contient une représentation binaire de ces informations et non une représentation symbolique

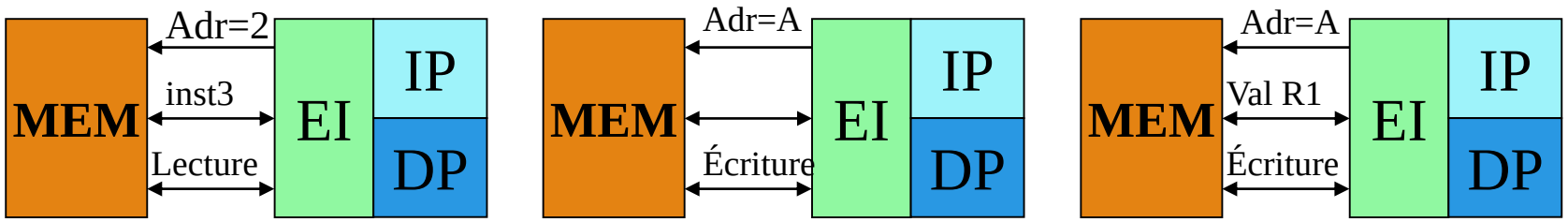
Architecture programmable

Exemple (suite): Exécution du cycle Von Neumann



Architecture programmable

Exemple (suite): Exécution du cycle Von Neumann



Architecture programmable

Exemple (suite): Visualisation du chargement d'une instruction

- Utilisation d'un simulateur (J. Garton)

PLAN

Introduction

Architecture programmable

Processeur : modèle et caractéristiques

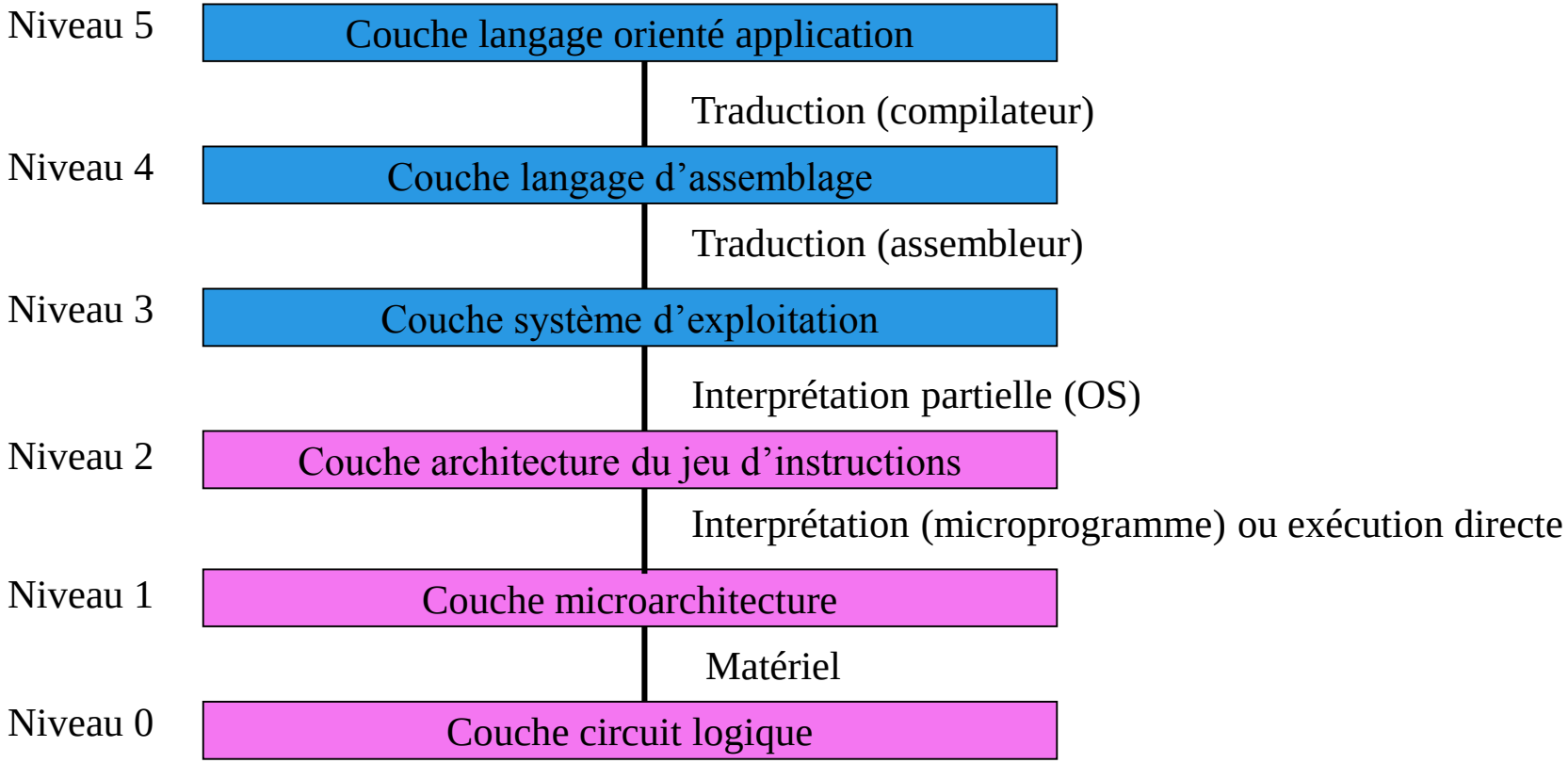
Machine câblée versus machine micro programmée

Présentation du système de mémorisation

Processeur : modèle et caractéristiques

Un système à microprocesseur peut être représenté par un modèle dit modèle en couche (A. Tannenbaum)

- Un système possède au minimum 2 couches et peut en compter jusqu'à 6.



Processeur : modèle et caractéristiques

Caractérisation des processeurs:

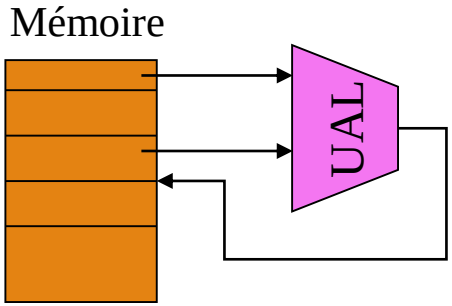
- Largeur des mots manipulés
- La fréquence d'horloge
- Jeu d'instructions
  - Instructions complexes ou non (CISC vs RISC)
- Modes d'adressage
  - Adressage complexe ou non (CISC vs RISC)
  - Adressage spécialisé ou non (DSP vs GPP)
- Mesure de performances
  - Le GIPS : Milliard d'instructions par seconde
  - Le GFLOP : Milliard d'opérations flottantes par seconde
  - Le Whestone est un calcul mettant en œuvre une MPY et une ADD (tps de traitement + retards dus aux transferts de données)

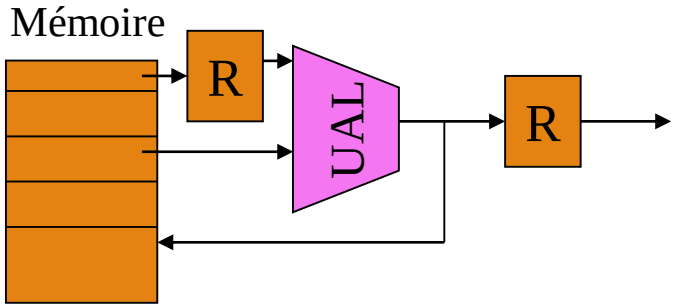
Processeur : modèle et caractéristiques

Caractérisation des processeurs:

- Nature de l'architecture

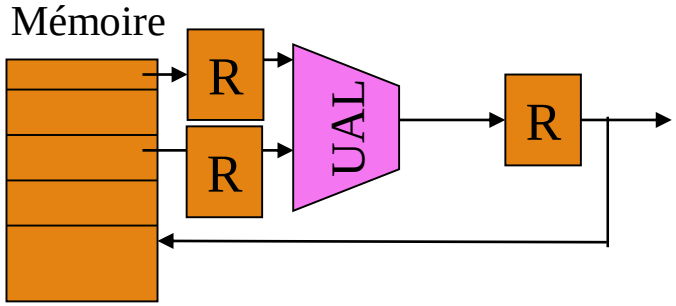
Architecture mémoire/mémoire





Architecture mémoire/registre (CISC/DSP)

Architecture registre/registre (RISC/DSP)



Processeur : modèle et caractéristiques

Architecture du jeu d'instructions (ISA)

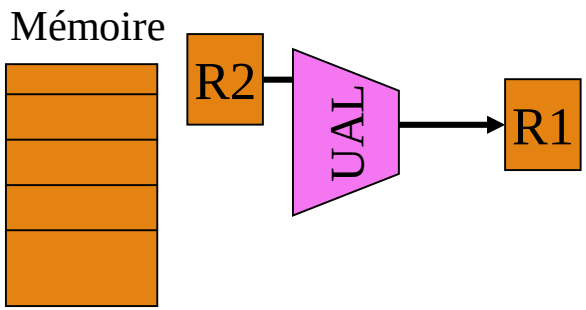
- **Instruction de traitement**: permettent de réaliser des opérations arithmétiques et logiques. Certaines positionnent des comptes rendus (drapeaux ou flags) tels que nul, overflow, carry…
- **Instructions de contrôle**: permettent de modifier le déroulement du programme en fonction d'évènements internes (flags) ou externes (IT). Elles affectent le compteur de programme (PC) et sont classées suivant 3 critères:
  - Débranchement conditionnel ou inconditionnel (flags)
  - Débranchement explicite ou implicite (label ou adresse)
  - Débranchement avec ou sans restitution de contexte (saut, appel de sous programmes).
- **Instruction de transfert**: permettent d'accéder à la mémoire depuis ou vers un registre du microprocesseur (différents modes d'adressage sont possibles).

Processeur : modèle et caractéristiques

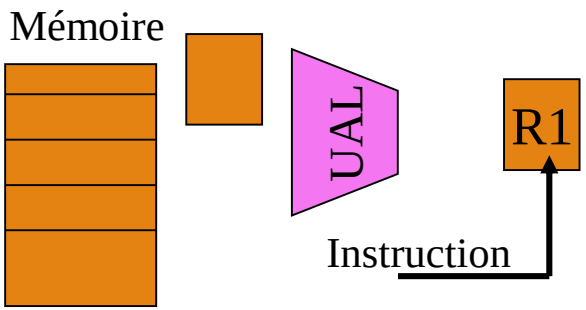
Architecture du jeu d'instructions (ISA)

- Instructions de transfert: les modes d'adressage

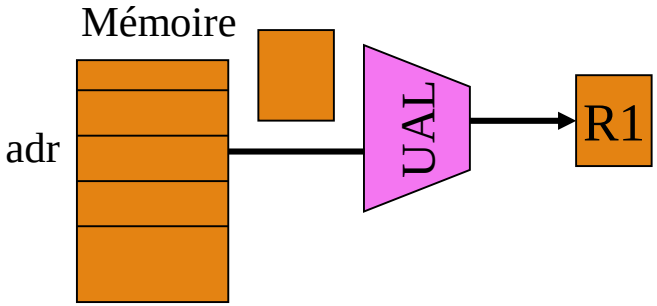
Transfert registre/registre (move R1,R2)



Chargement immédiat (move R1,10)



Chargement absolu (move R1,(adr))

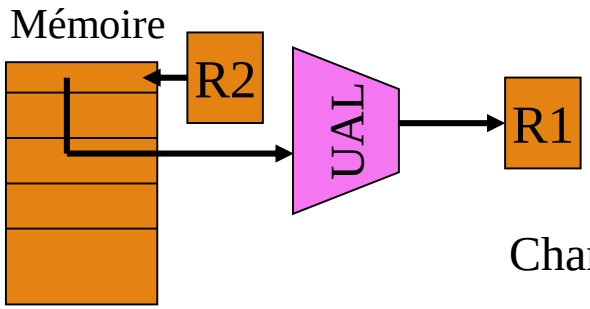


Processeur : modèle et caractéristiques

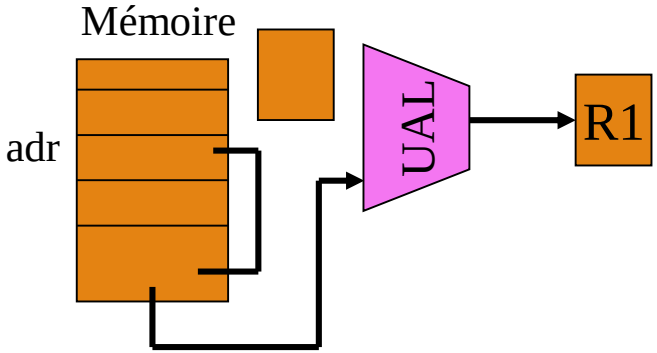
Architecture du jeu d'instructions (ISA)

- Instructions de transfert: les modes d'adressage

Chargement indirect (move R1,(R2))



Chargement indirect référencé à la mémoire (move R1,@(adr))



Processeur : modèle et caractéristiques

Architecture du jeu d'instructions (ISA)

- Instructions de transfert: les modes d'adressage
  - Autres mécanismes d'adressage:
    - Adressage indirect post incrémenté: après l'accès le contenu du registre d'indirection est incrémenté.
    - Adressage indirect pré incrémenté: avant l'accès le contenu du registre d'indirection est décrémenté.
    - Adressage indexé: Ri contient l'adresse de la base et Rj celle du déplacement -> adresse = (Ri) + (Rj)
    - Adressage indexé avec déplacement: adresse = (Ri) + (Rj) + dep
    - Adressage relatif (au PC) indexé: Le PC est un registre particulier qui pointe sur l'instruction en cours d'exécution -> adresse = (PC) + (Ri)

PLAN

Introduction

Architecture programmable

Processeur : modèle et caractéristiques

Machine câblée versus machine micro programmée

Présentation du système de mémorisation

Machine câblée versus micro programmée

Machine câblée:

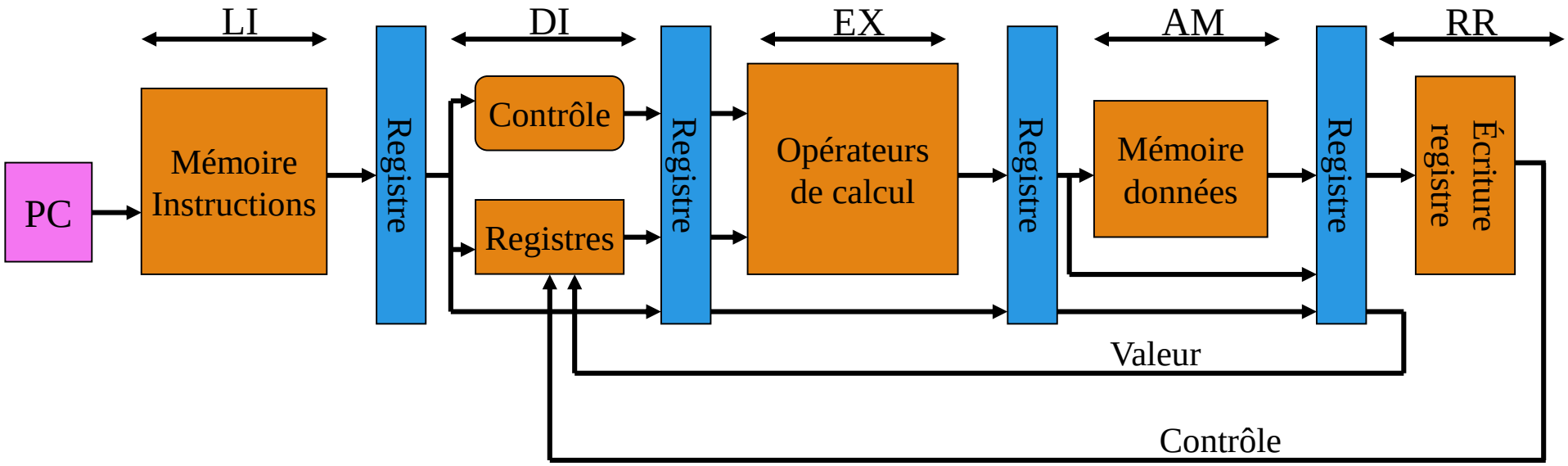
- Ce type d'architecture réalise toujours le même cycle d'interprétation pour exécuter une instruction (généralement en 5 étapes élémentaires). Les instructions sont de taille fixe et en nombre réduit (RISC: cf. cours de Master1).

Machine micro programmée:

- Ce type d'architecture réalise différents cycles d'interprétation pour exécuter une instruction (la taille du cycle est variable). Les instructions sont de taille variable et en grand nombre (CISC).

Machine câblée versus micro programmée

Machine câblée (RISC): Exemple



5 étage: Lecture Instruction, Décodage, Exécution, Accès Mémoire, Rangement Résultat.

Machine câblée versus micro programmée

Machine micro programmée (CISC): Exemple

- Considérons un langage L1 comprenant les instructions suivantes (***X signifie bit non utilisé***):
  - MOVE Ri,Rj (Ri:=Rj), codée 00XXXXXX
  - SUB Ri,Rj,Rk (Ri:=Rj – Rk), codée 01XXXXXX
  - JZ @ (saut à l'adresse précisée si le compte rendu Z vaut 1, sinon passage en séquence), codée 10XXXXXX
  - JMP @ (saut à l'adresse précisée) codée 11XXXXXX
- Considérons que ces instructions sont codées de la façon suivante:

Code opératoire	1er octet
Opérande 1	2nd octet
Opérande 2	3ième octet

Pour les instructions de transfert et calcul

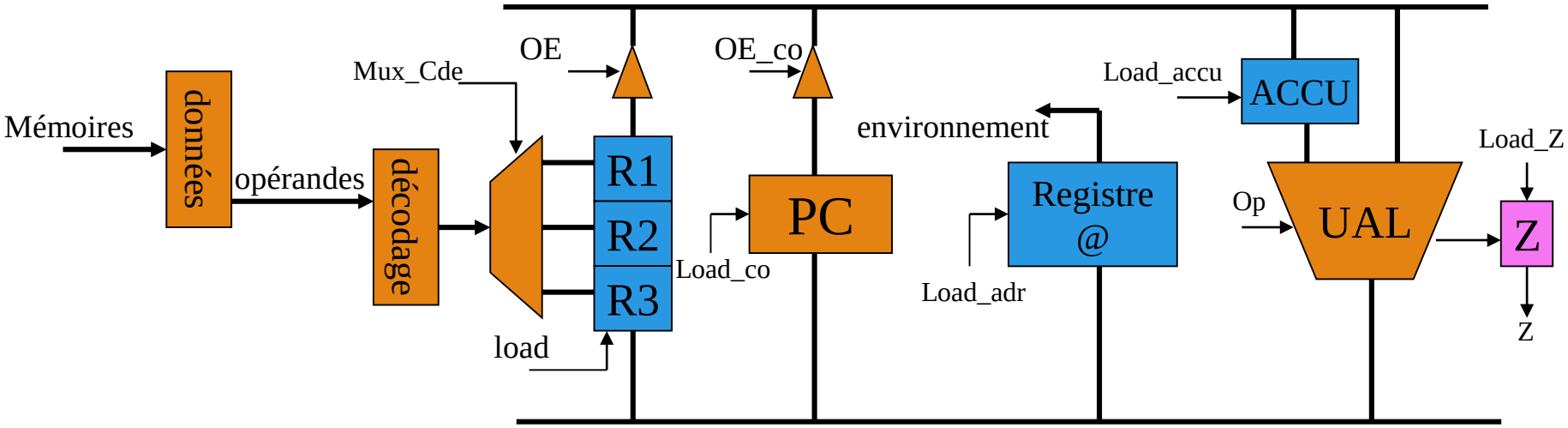
Code opératoire	1er octet
Opérande	2nd octet
XXXX	3ième octet

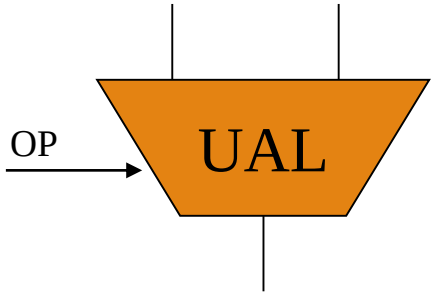
Pour les instructions de saut

Machine câblée versus micro programmée

Exemple (suite)

- Schéma de l'automate de traitement de niveau 0





Op = 00 fonction incrémentation

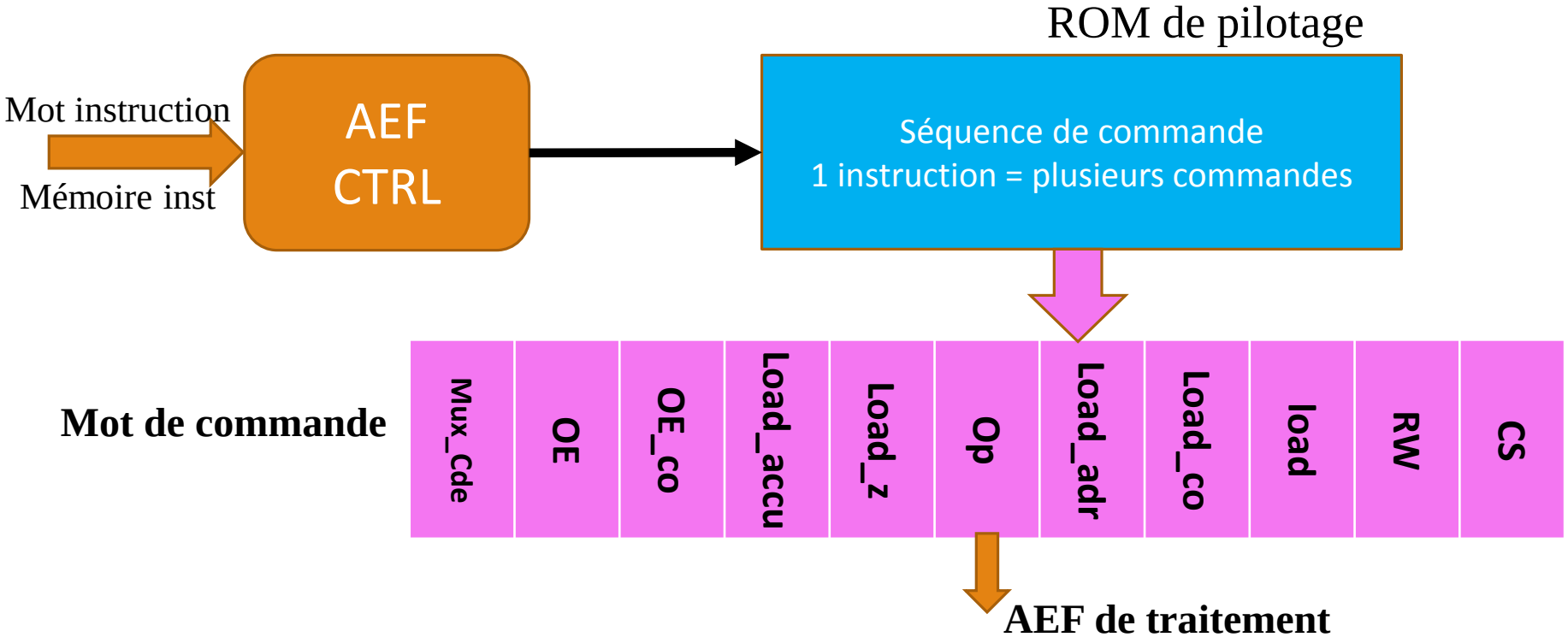
Op = 01 fonction transparence

Op = 10 fonction soustraction

Op = 11 configuration non utilisée

Machine câblée versus micro programmée

- Schéma de l'automate de commande de niveau 0



Rq: Les signaux CS et RW permettent de gérer l'environnement (mémoire: CS signal de validation, RD signal de lecture)

Machine câblée versus micro programmée

Exemple (suite)

Attention certaines opérations ne peuvent pas s'exécuter simultanément

@ Mem	Mux_ Cde	OE	OE_co	Load_ accu	Load_ z	Op	Load_ adr	Load_ co	load	RW	CS
0000	x	x	1	x	x	x	1	x	x	1	1
0001	1er bit testé est à 1, on saute à une autre partie de la mémoire de décodage										
0100	Adresse début du programme d'interprétation du JMP										
	x	x	1	x	x	00	x	x	x	x	X
0101	x	x	x	x	x	x	1	x	x	1	1
0110	x	x	x	x	x	X	1	x	x	1	1
0111	x	x	x	x	x	01	x	1	x	x	X
1000	Retour à la première ligne de la mémoire de décodage										

Microprogramme d'interprétation de l'instruction JMP @

Machine câblée versus micro programmée

Exemple (suite): Simulation de l’exécution d’un code sur une machine de Von Neumann

PLAN

Introduction

Architecture programmable

Processeur : modèle et caractéristiques

Machine câblée versus machine micro programmée

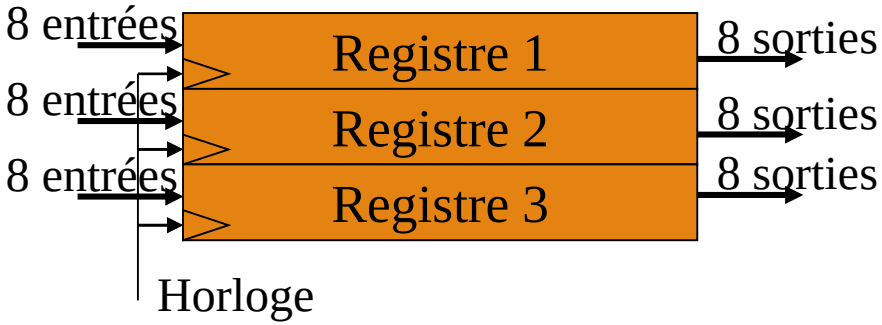
Présentation du système de mémorisation

Organisation et gestion de la mémoire

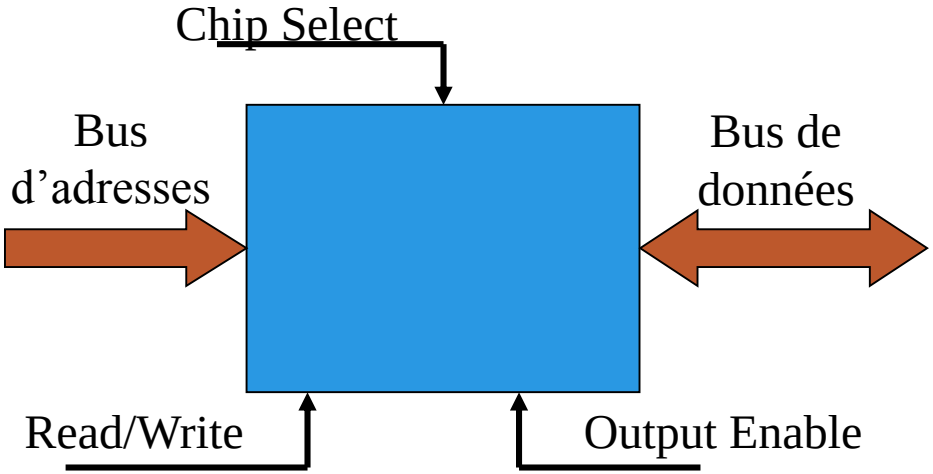
Réalisation de la mémoire: 2 solutions possibles

-A base de registres (association de bascules)

- coûteux, pb d'entrées/sorties



-Solution privilégiée: organisation avec 1 bus de donnée et 1 bus d'adressage



Organisation et gestion de la mémoire

Caractéristiques des mémoires:

- Taille et organisation: On compte généralement en « kilo » et on précise la largeur
  - Exemple: 1 mémoire 2K x 8 peut contenir 2048 mots de 8 bits; elle est adressée par 11 bits.
- Type:
  - RAM=> Random Access Memory: mémoire vive qui permet la lecture et l'écriture.
    - Ram statiques: basées sur une structure semblable à celle des bascules (4 à 6 transistors/bit)
    - Ram dynamiques: basées sur des points mémoires type condensateur (1 transistor/bit); nécessite un rafraîchissement régulier.

Organisation et gestion de la mémoire

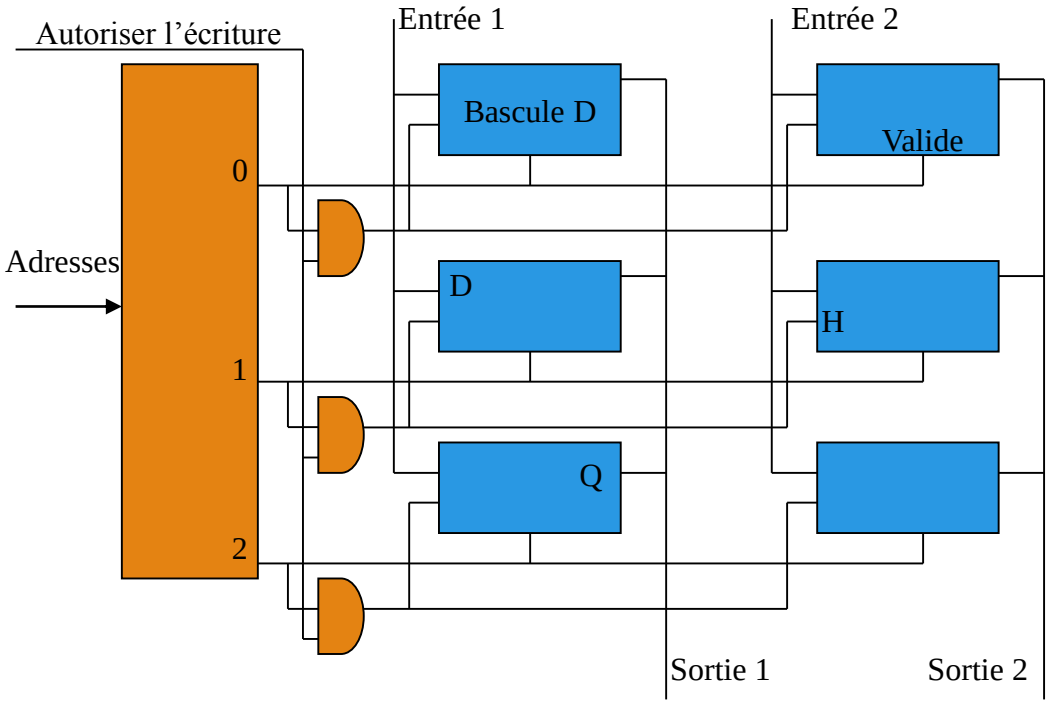
Caractéristiques des mémoires:

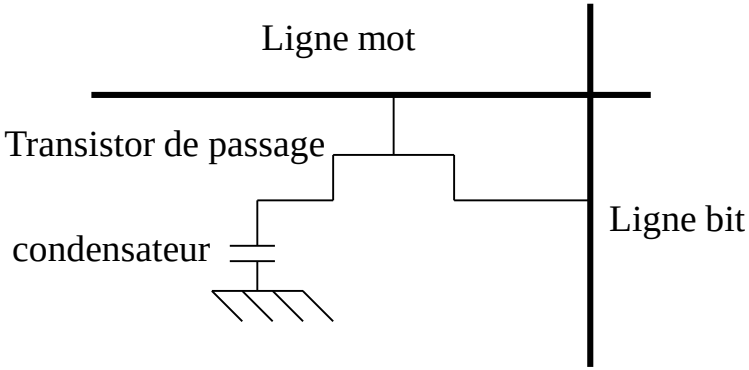
- Type:
  - ROM=> Read Only Memory: mémoire morte écrite à la construction (ROM) ou par programmation (PROM).
    - Rom non effaçable (OTP)
    - Rom effaçable électriquement (EEPROM/FLASH)
    - Rom effaçable par ultra violet (UVPROM)
- Outre la capacité et le type, d'autres paramètres guident le choix d'une mémoire:
  - Vitesse d'accès
  - Consommation électrique
  - Type d'accès (aléatoire ou séquentiel)

Organisation et gestion de la mémoire

Caractéristiques des mémoires (suite)

Architecture d'une SRAM 3x2; les signaux CE et OE ne sont pas représentés



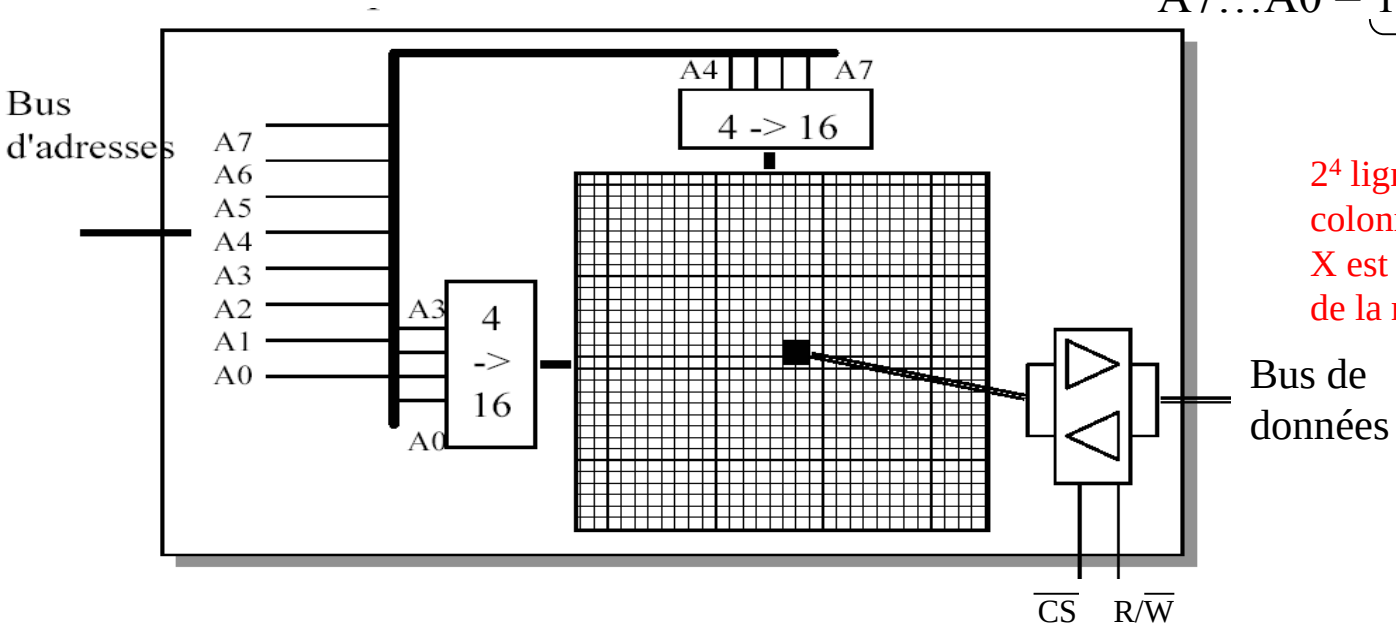


Constitution du point mémoire élémentaire pour une DRAM

Organisation et gestion de la mémoire

Caractéristiques des mémoires (suite)

Mots accessibles de A7…A0 = $\overbrace{00000000}^{00h}$ à A7…A0 = $\underbrace{1111111}_{11h}$



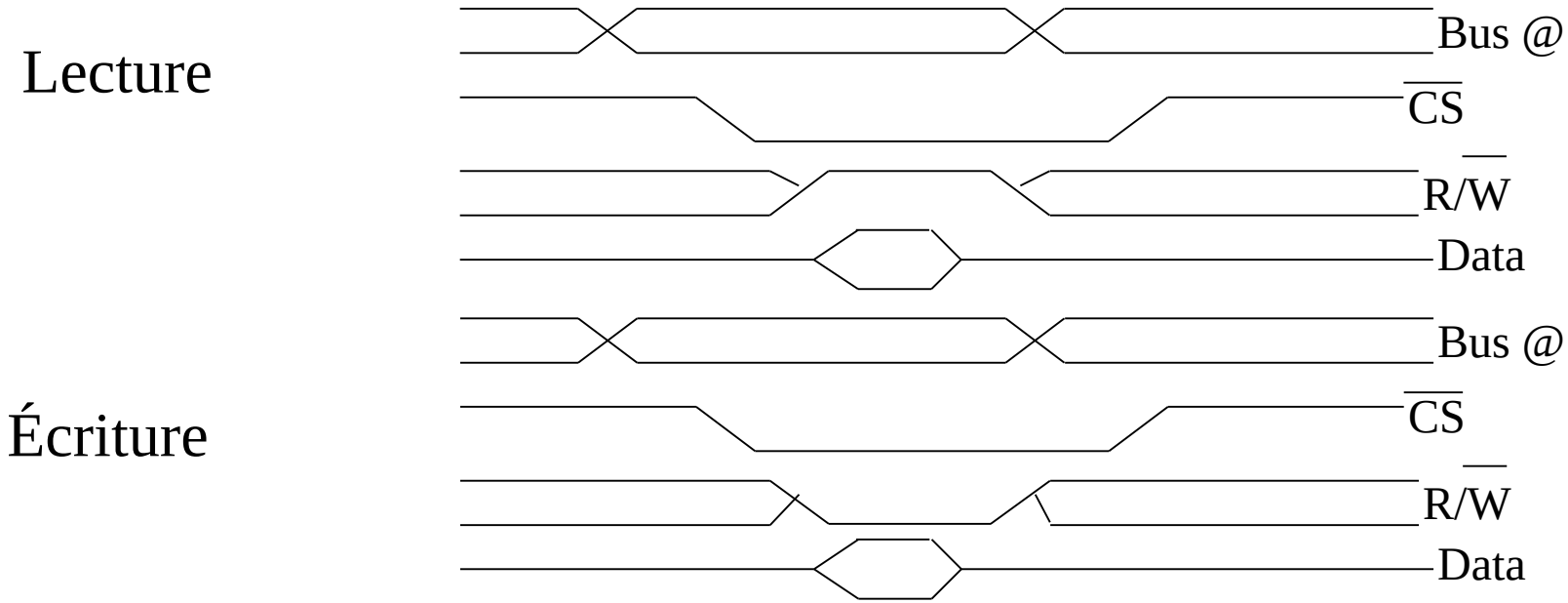
2^4 lignes soit 16 lignes et 2^4 colonnes soit 16 colonnes de x bits. X est la largeur du bus de données de la mémoire.

$\overline{CS}$: Sélection de boîtier; R/$\overline{W}$ sens lecture ou écriture

Organisation et gestion de la mémoire

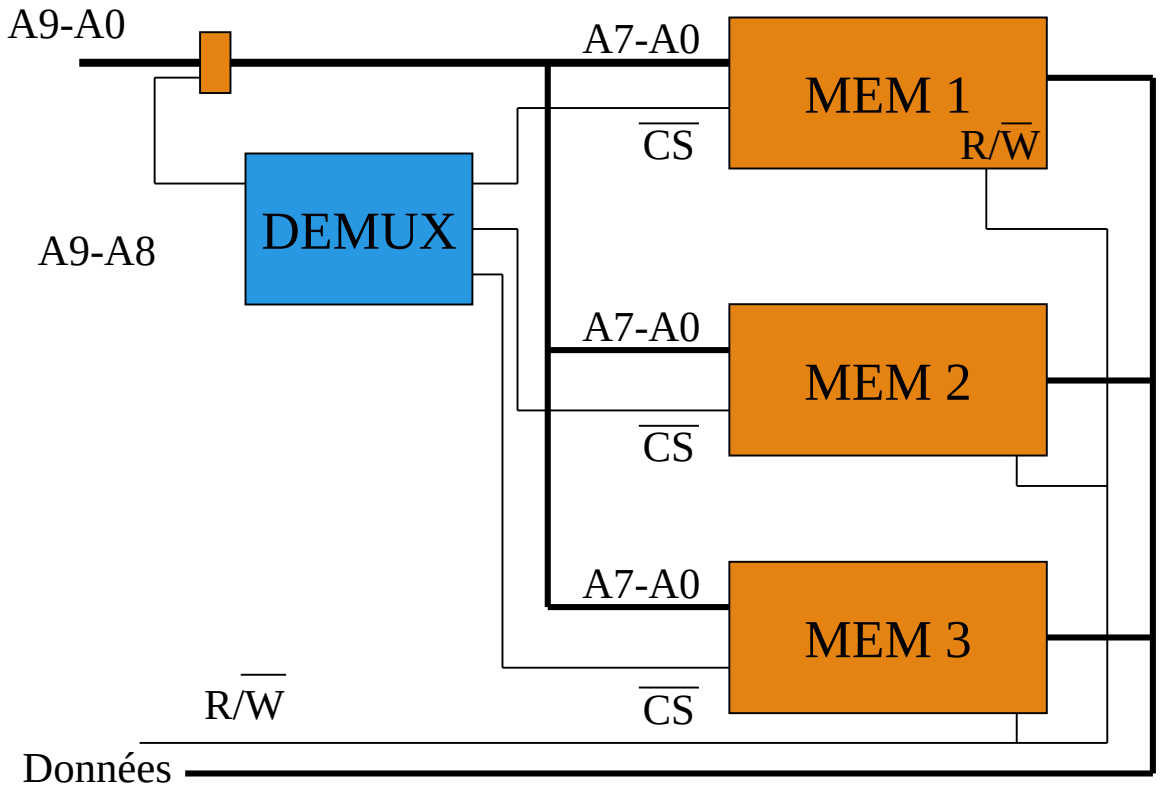
Caractéristiques des mémoires (suite)

- La gestion des échanges se fait en respectant divers protocoles spécifiés souvent à l'aide de chronogrammes.
  - Échange synchrone



Organisation et gestion de la mémoire

Association de boîtiers mémoire:



Organisation et gestion de la mémoire

Utilisation particulière: La pile

- Utilisée par les instructions conditionnelles
  - Implicite ou non: gère la sauvegarde et la restitution de contexte (appel de sous programme, interruption, trappes).
  - La pile est une zone mémoire « dernier entrée – premier sorti » ou LIFO.
  - Elle utilise un mécanisme d'adressage implicite.
  - On accède à la pile en utilisant un pointeur de pile (SP). On peut avoir plusieurs pointeurs (1 pour les variables locales, 1 pour les paramètres d'appel, 1 pour le contexte…).
  - Les instructions de pile sont PUSH (empile) et POP (dépile).

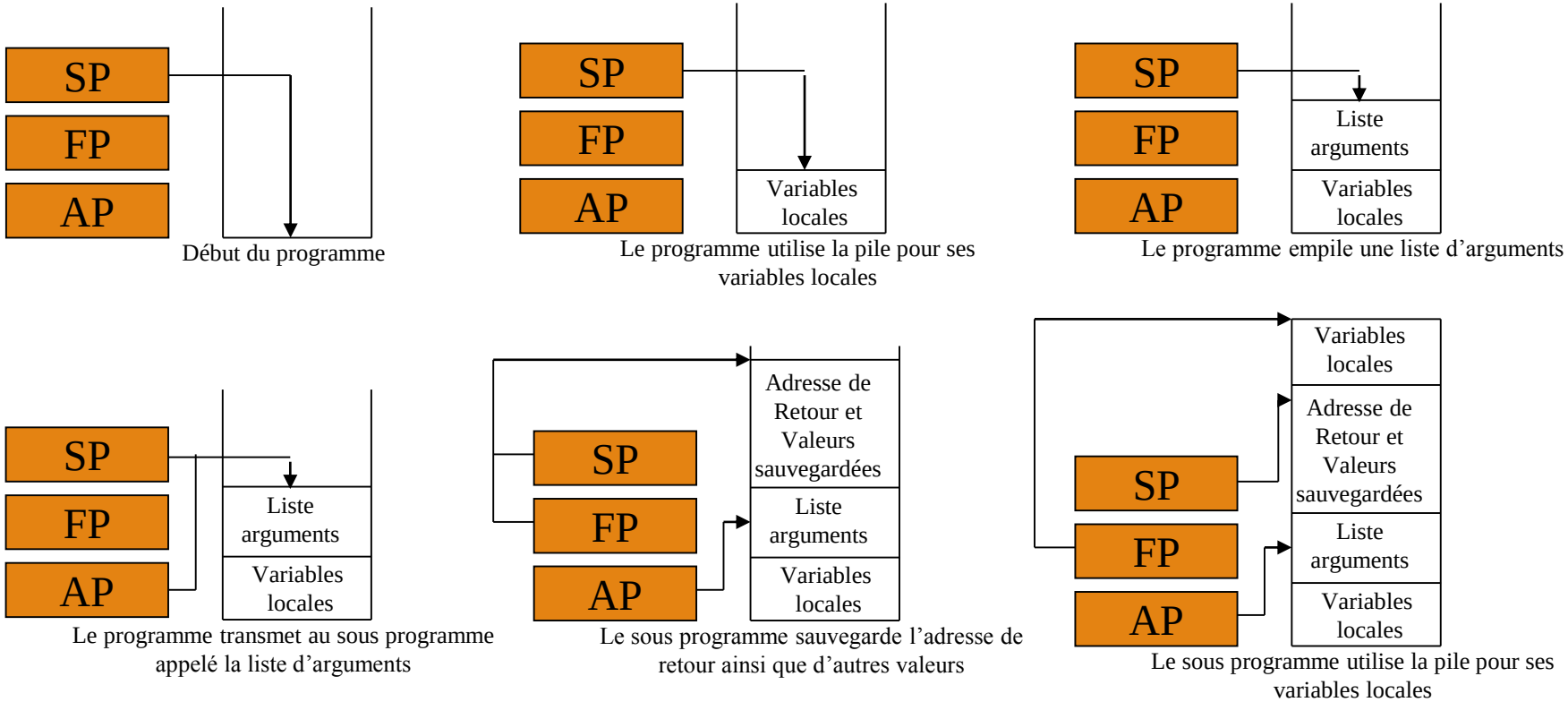
Organisation et gestion de la mémoire

Utilisation particulière: La pile

- L'appel à un sous programme (ex: CALL)
  - Sauvegarde du contexte d'exécution du programme appelant (PUSH). L'exécution de l'instruction de retour (ex: RTS) provoque la restitution (POP) du contexte depuis la pile.
  - Le passage de paramètres (entre le programme appelant et l'appelé) utilise la pile.
  - Les variables locales d'une procédure sont résidentes dans la pile.
- Interruption et trappes
  - Utilise une sauvegarde du contexte d'exécution du programme interrompu dans la pile.

Organisation et gestion de la mémoire

Exemple: Programme procédant à l'appel d'un sous programme: on utilise 3 pointeurs (SP, FP & AP)



Organisation et gestion de la mémoire

Exemple d'utilisation de la pile:

- 1 sous programme récursif ayant comme arguments h,i,j
- K: variable locale du sous programme
- Les données sont codées sur 16 bits
- Le contexte est constitué de l'adresse de retour et du FP

	Contenu de la pile	Adresse
PP →	K=2	990
FP →	Ancien FP	992
	Adresse de retour	994
	J=3	996
	I=1	998
	H=3	1000

1ère itération du sous programme

2ème itération

	Contenu de la pile	Adresse	
PP →	K=3	978	Variable locale
FP →	992(FP)	980	contexte
	Adresse de retour (994)	982	
	J=2	984	arguments
	I=0	986	
	H=2	988	
	K=2	990	
	Ancien FP	992	
	Adresse de retour	994	
	J=3	996	
	I=1	998	
	H=3	1000	

Organisation et gestion de la mémoire

3ème exécution

	Contenu de la pile	Adresse
PP →	K=4	966
FP →	980(FP)	968
	Adresse de retour (982)	970
	J=1	972
	I=5	974
	H=1	976
	K=3	978
	992(FP)	980
	Adresse de retour (994)	982
	J=2	984
	I=0	986
	H=2	988
	K=2	990
	Ancien FP	992
	Adresse de retour	994
	J=3	996
	I=1	998
	H=3	1000

Cette exécution ne provoque pas de nouvel appel et le sous programme se termine

	Contenu de la pile	Adresse
PP →	992(FP)	980
FP →	Adresse de retour (994)	982
	J=2	984
	I=0	986
	H=2	988
	K=2	990
	Ancien FP	992
	Adresse de retour	994
	J=3	996
	I=1	998
	H=3	1000

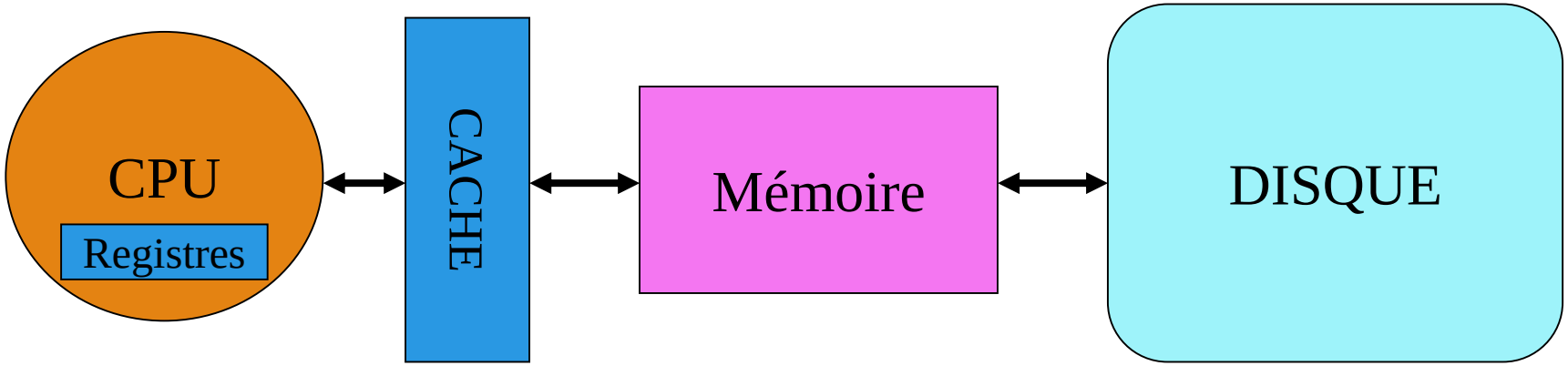
Principe de la mémoire hiérarchisée

- Le processeur accède au niveau le plus bas de la hiérarchie

La hiérarchie est composée de mémoires rapides (tps accès court) mais petites (qq Ko à qq Mo) et de mémoires lentes mais grosses (qq Go/To).

- La mémoire la plus rapide de la hiérarchie doit avoir un temps d'accès proche du temps de cycle processeur.
- La mémoire la plus lente doit être la plus grande possible

Principe de la mémoire hiérarchisée



Définition du temps d'accès moyen à un niveau n

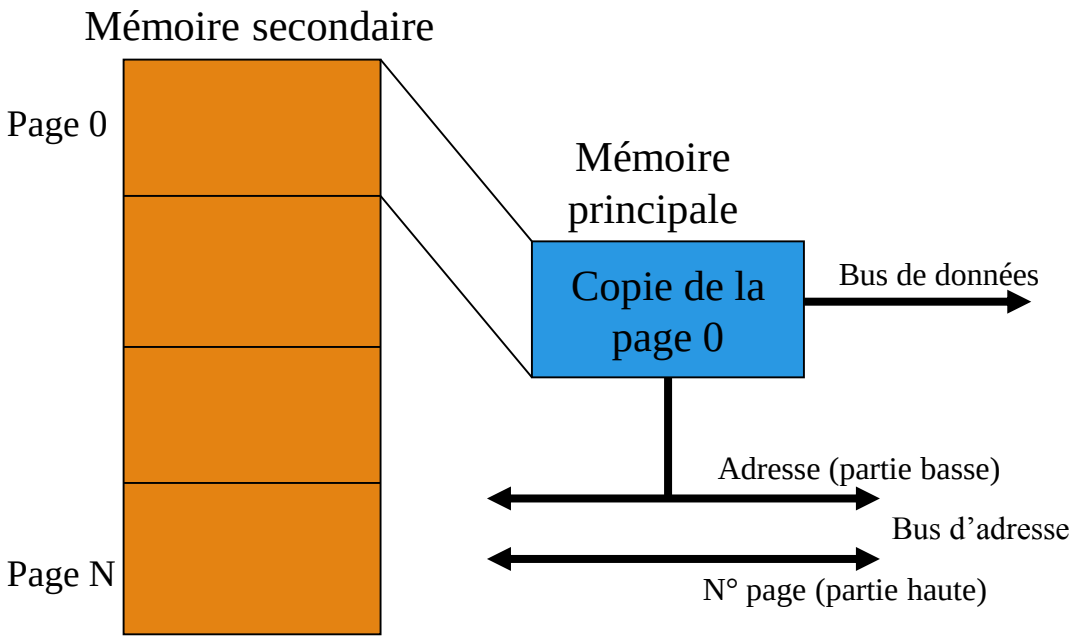
- Tmoy = P Tn + (1 – P) Tsup
  - P: facteur de succès, 1-P facteur d'échec
  - Tn: temps d'accès de la mémoire de niveau n
  - Tsup: temps moyen d'accès aux mémoire de rang < n
- But: Avoir P le plus proche possible de 1 avec Tn le plus proche possible du T souhaité (temps cycle processeur)

Principe de la mémoire hiérarchisée

Une mémoire de niveau n peut être considérée comme une fenêtre ouverte sur une mémoire de niveau n+1 de taille plus grande mais, moins rapide.

-Généralement 3 Niveaux

- -Mémoire cache
  - -Cache primaire
  - -Cache secondaire
- -Mémoire centrale (RAM)
- -Mémoire secondaire (disque)



Principe de la mémoire hiérarchisée

Tous les objets stockés à un niveau n+1 ne peuvent pas être simultanément présents au niveau n. Il en va de même entre le niveau n+2 et n+1.

Les grands mécanismes:

- Gestion des défauts: on fait référence à un objet non présent dans le niveau n mais, présent dans un niveau supérieur.
  - Le processeur va chercher la donnée au niveau le plus bas si il y a un défaut alors la demande est propagée vers la couche supérieure et ainsi de suite jusqu'à la réussite de l'accès.
- Gestion de la translation: après un deuxième chargement au niveau n, un bloc de niveau n+1 n'est pas forcément à la même adresse qu'après son premier chargement au niveau n.
- Gestion de l'anticipation: le chargement au niveau n depuis n+1 n'est pas uniquement consécutif à un défaut mais, on peut anticiper les besoins futurs de telle sorte qu'une donnée soit chargée avant d'être utilisée par le programme.

Principe de la mémoire hiérarchisée

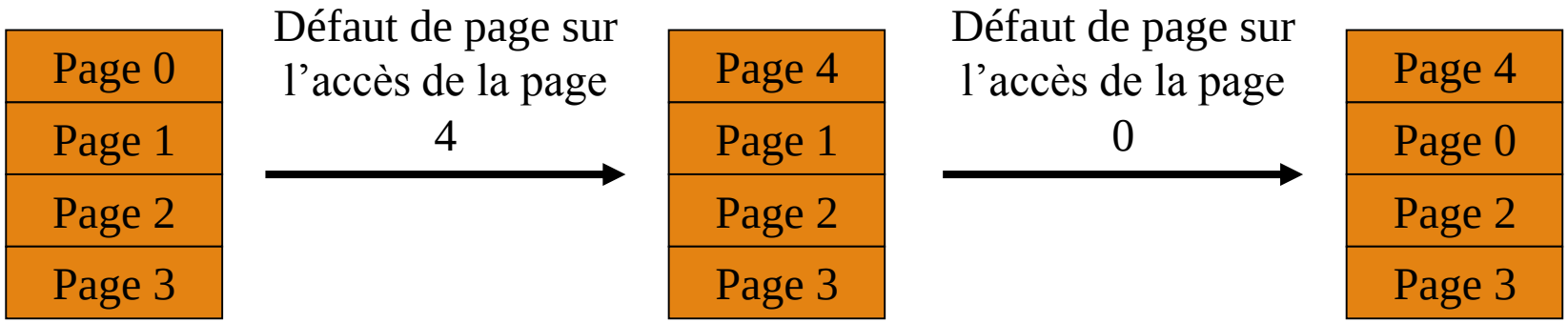
Gestion des défauts:

- Si la page référencée n'est pas en mémoire centrale: défaut de pages (exceptions).
- Si la mémoire centrale est totalement occupée, élection de la page qui va être supprimée de la mémoire principale (LRU, FIFO…)
- Sauvegarde de la mémoire centrale en mémoire secondaire
- Localisation de la page manquante en mémoire secondaire grâce à la table des pages
- Chargement de la page en mémoire principale
- Modification de la table des pages
- Remplacement des pages: différents algorithmes (LRU, FIFO…)

Principe de la mémoire hiérarchisée

Remplacement de pages: exemple

- Mise en échec de l'algorithme LRU.
  - Hypothèse: le programme boucle sur 5 pages alors que la mémoire physique ne peut en contenir que 4.

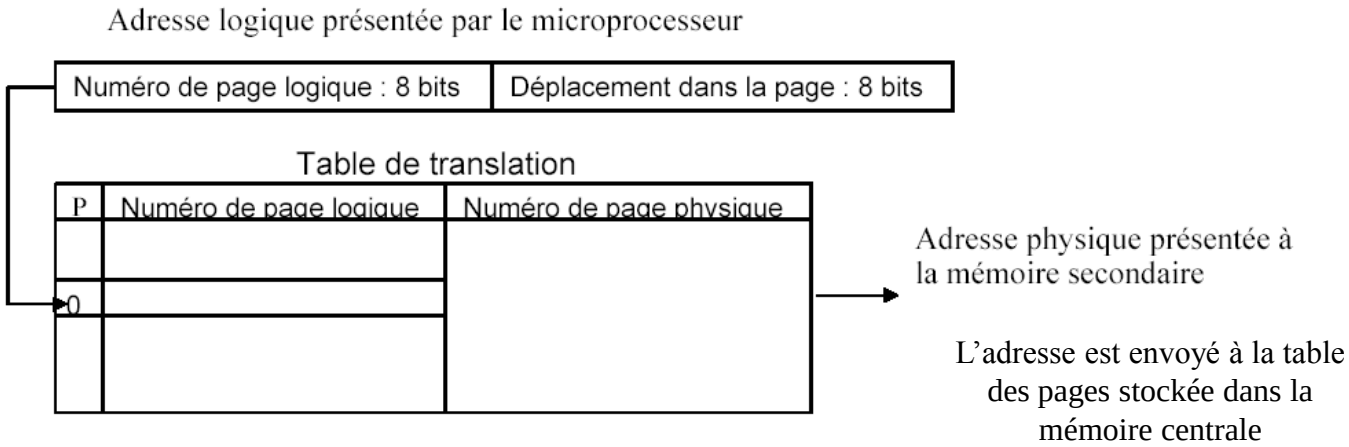


On enlève de la mémoire la page dont on aura besoin juste après

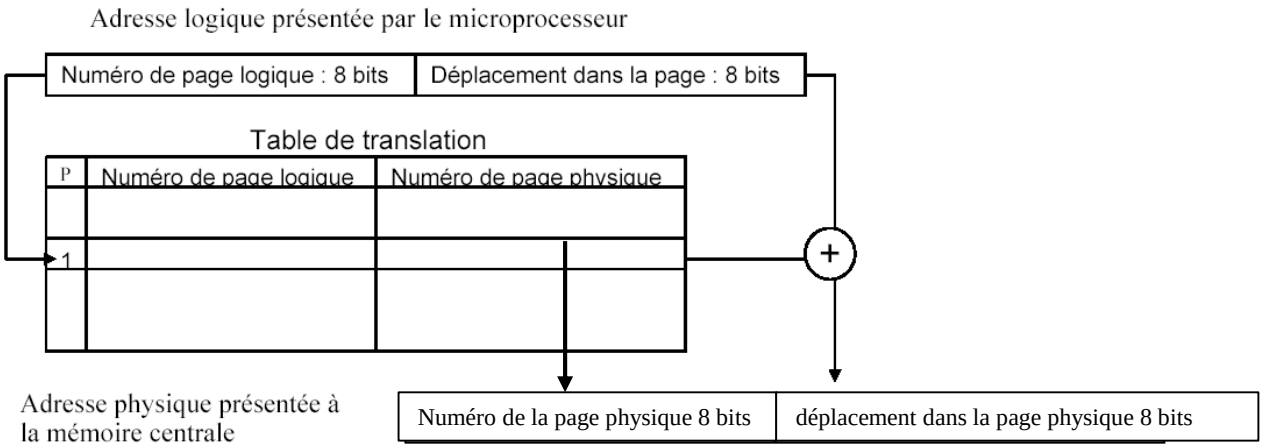
Principe de la mémoire hiérarchisée

Gestion de la translation

Défaut



Pas de défaut



Principe de la mémoire hiérarchisée

Gestion de l'anticipation

- Hypothèse de régularité
  - Au niveau du programme
  - Au niveau des données
- Pré chargement
- Si défaut: chargement depuis le niveau n+1 de la structure à laquelle appartient la donnée ayant provoquée le défaut et pas uniquement la donnée.

Principe de la mémoire hiérarchisée

Exemple:

- Adresse logique sur 16 bits: 2^{16}-1 objets adressables
- Adresse physique sur 12 bits: 2^{12}-1 objets adressables
- Système paginé, 1 page: 2^{8}-1 objets adressables

- Espace d'adressage logique: 256 pages
- Espace d'adressage physique: 16 pages

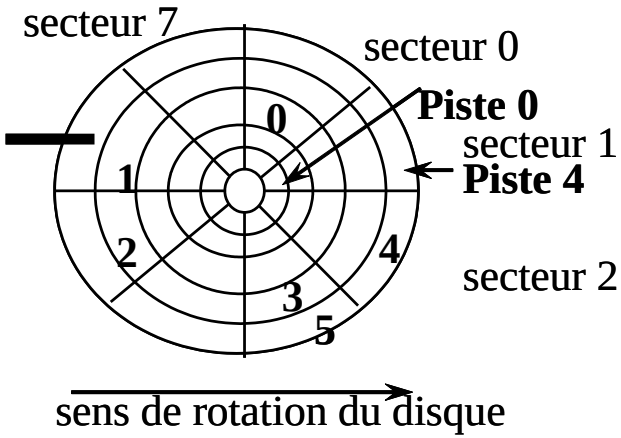
– Accès aux informations en mémoire secondaire (disque)

Secteurs

Pistes

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

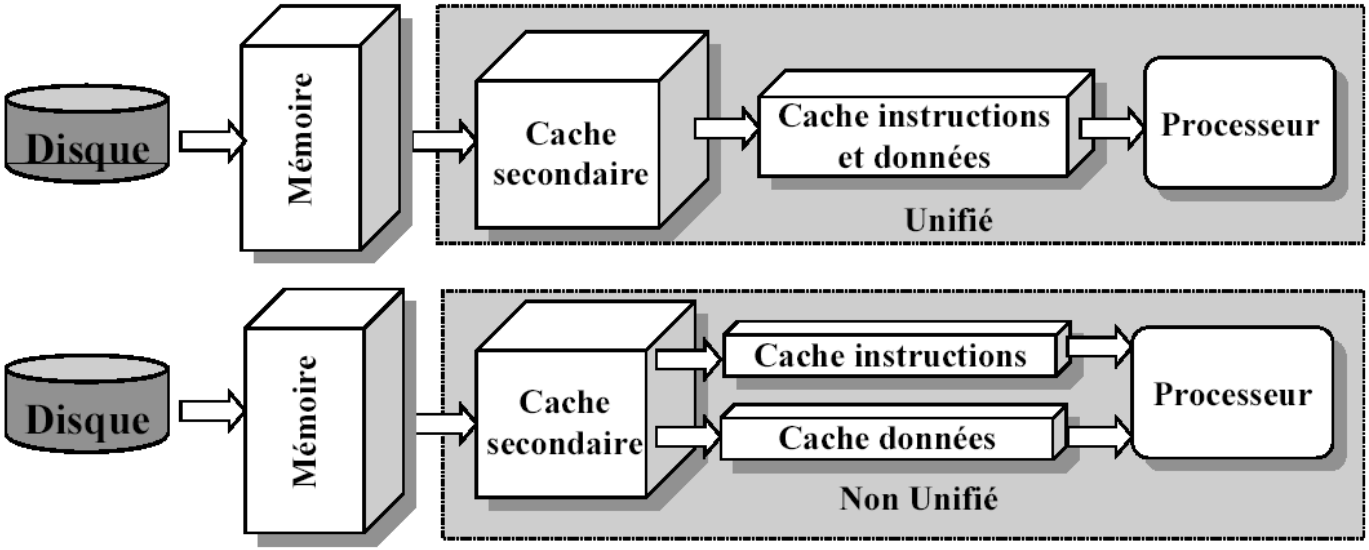
secteur 0 piste 2
secteur 6 piste 3
secteur 5 piste 3
secteur 3 piste 3
secteur 2 piste 4
secteur 3 piste4



Les mémoires caches

Localisation:

- Intégrée
- Pouvant être elles mêmes hiérarchisées (L1, L2, L3)
- Unifiées ou non (contient à la fois les données et les instructions).



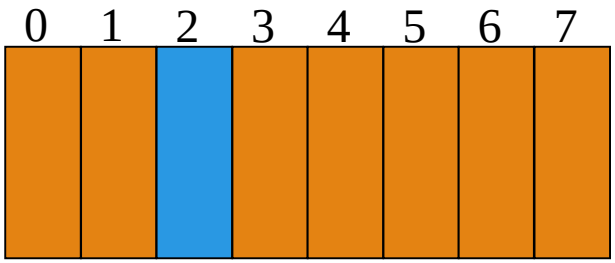
Les mémoires caches

Mécanismes d'organisation des caches:

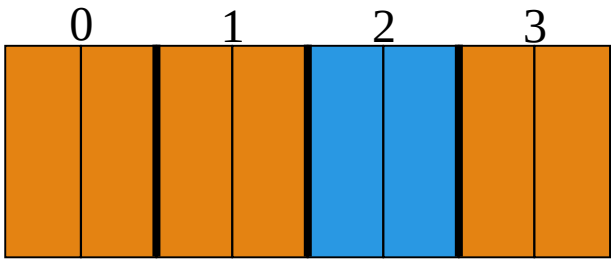
- Cache à accès direct:
  - 1 bloc n ne peut se trouver que dans le bloc n modulo l du cache (avec l représentant la taille du cache).
- Cache associatif par ensemble:
  - 1 bloc n ne peut se trouver que dans l'ensemble m modulo p du cache (avec p représentant le nombre d'ensemble du cache). Le bloc peut être dans n'importe quel élément de l'ensemble.
- Cache totalement associatif:
  - 1 bloc n peut se trouver n'importe où dans le cache.

Les mémoires caches

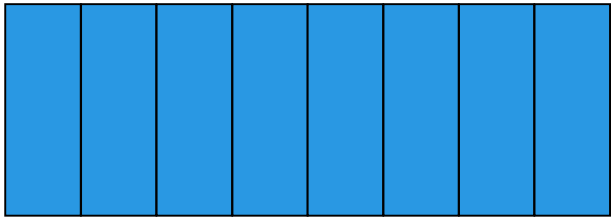
Exemple: Placement du bloc 10 pour un cache comprenant 8 blocs



Cache à accès direct (direct mapped)



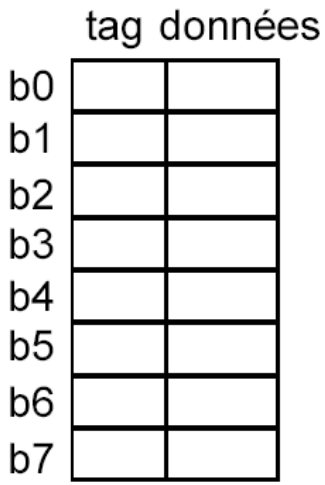
Cache associatif par ensemble de 2 (2 ways associative)



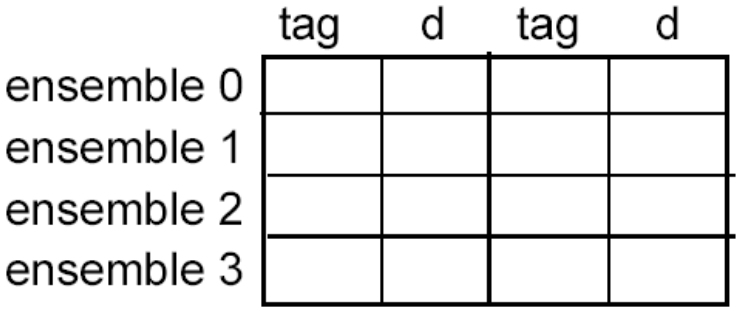
Cache totalement associatif (full associative)

Les mémoires caches

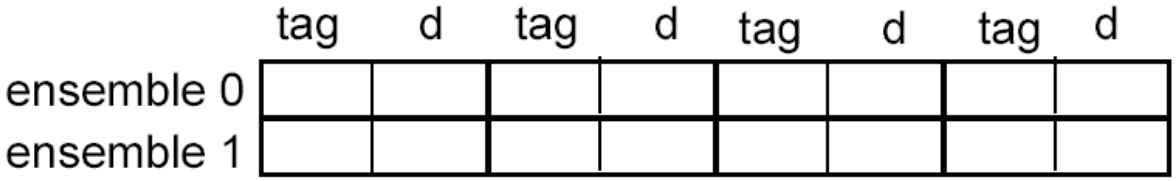
Mécanismes d'organisation des caches (suite):



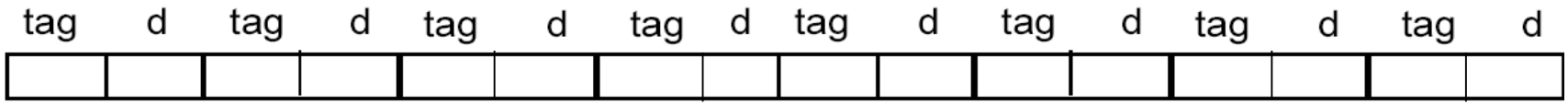
Associatif par ensemble de 1 bloc (à correspondance directe)



Associatif par ensemble de 2 blocs



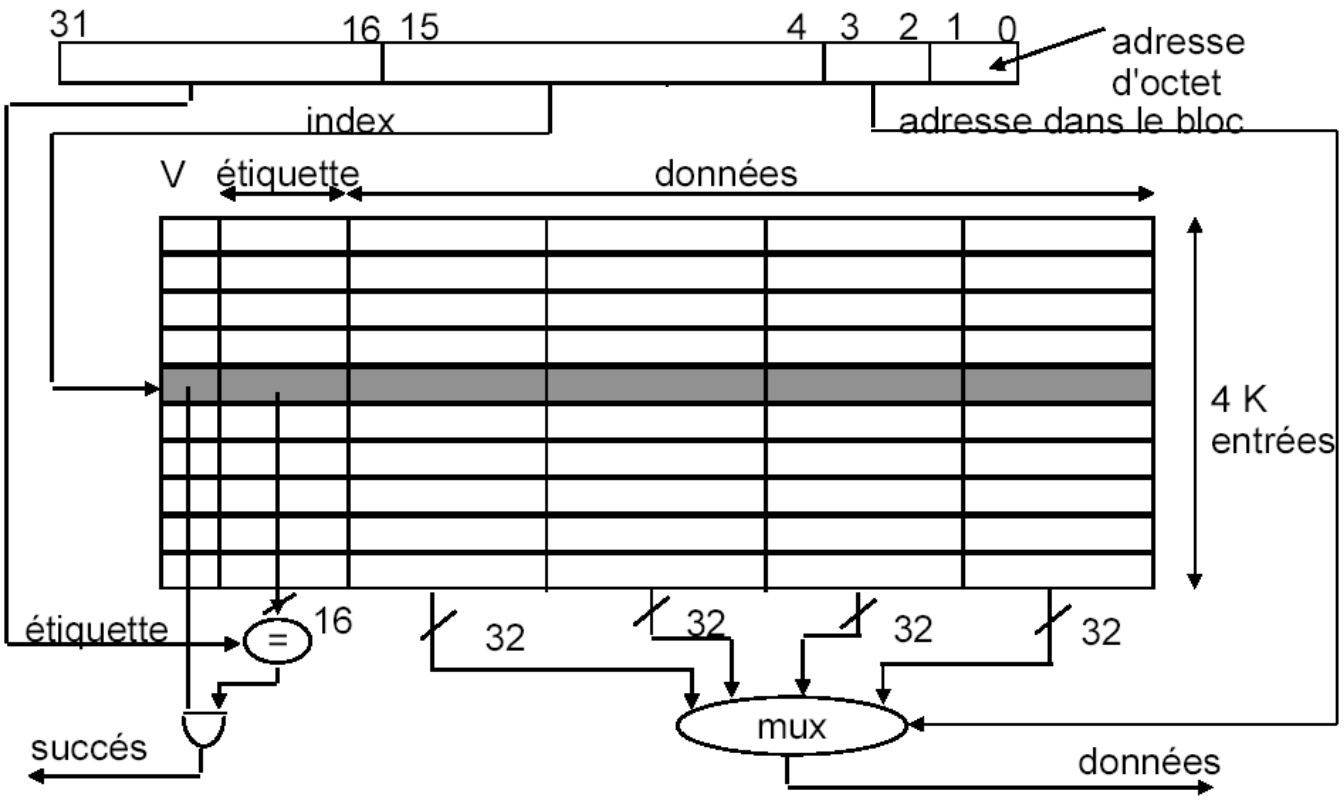
Associatif par ensemble de 4 blocs



Associatif par ensemble de 8 blocs (totalement associatif)

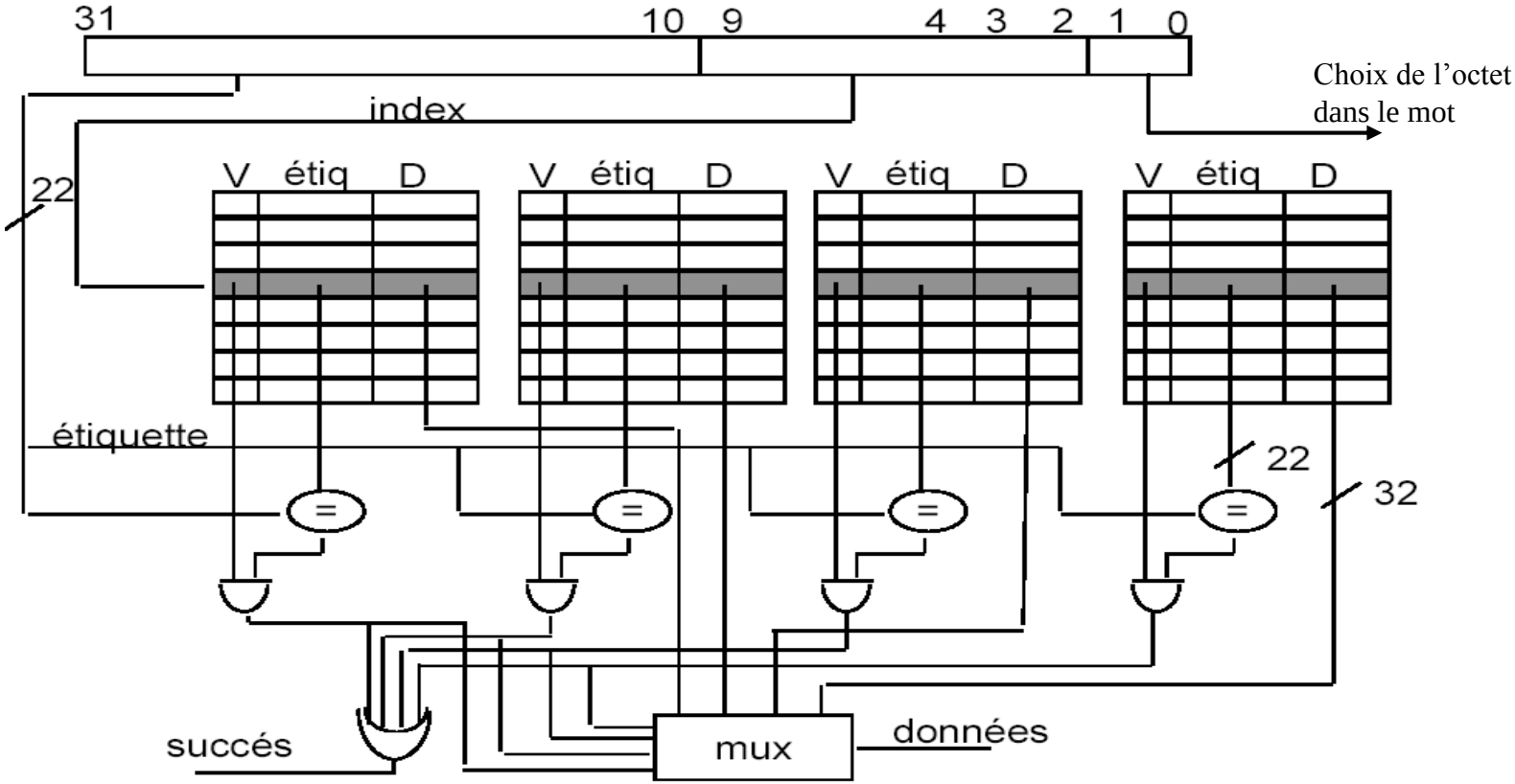
Les mémoires caches

Cache de 64Koctets avec des blocs de 4 mots



Les mémoires caches

Cache 4 ways associative



Les mémoires caches

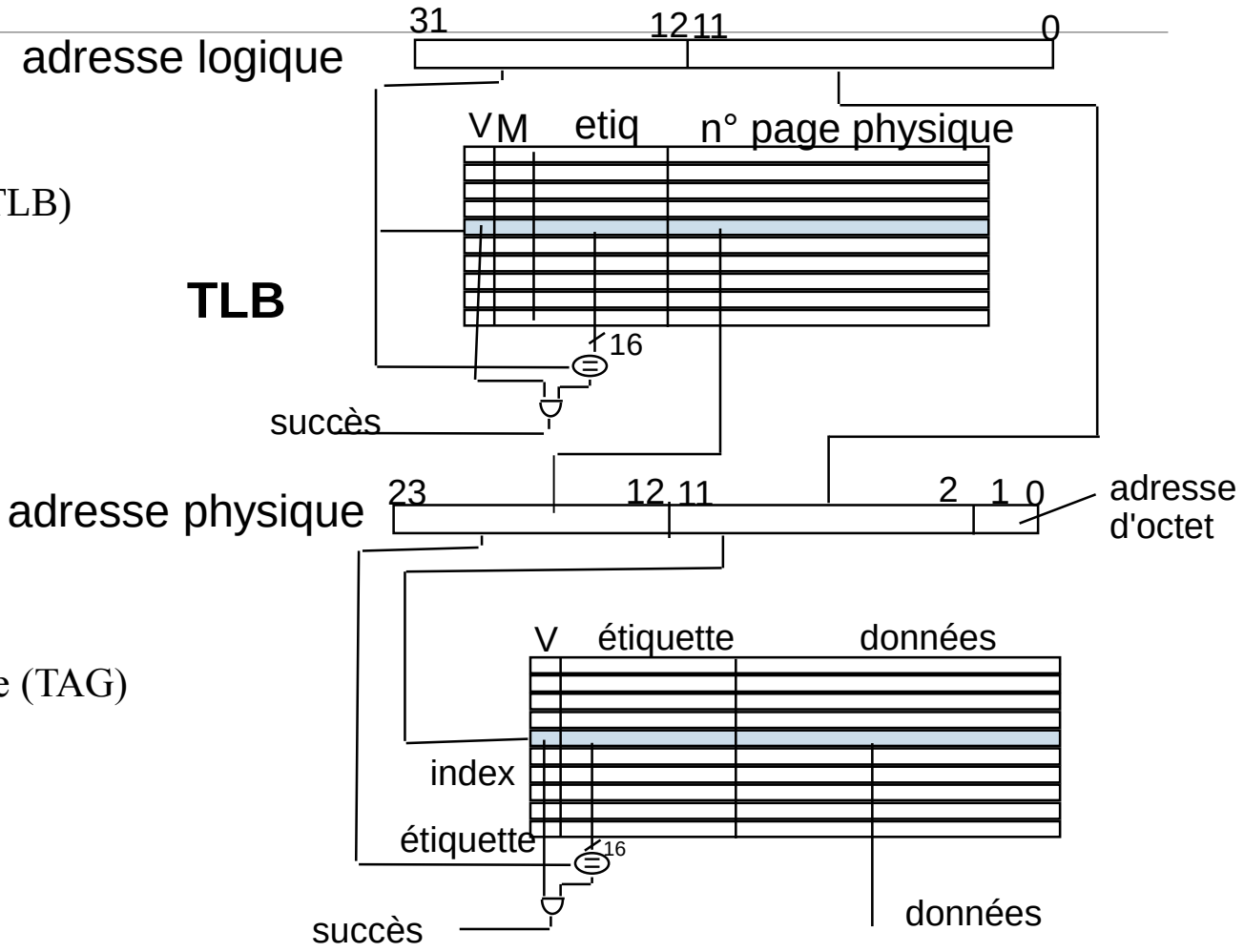
Politiques d'écriture:

- Écriture simultanée: write-through
  - La donnée est écrite à la fois dans le cache et dans la mémoire principale
- Réécriture (recopie ou rangement): write-back
  - La donnée est uniquement écrite dans le cache
  - Le bloc ainsi modifié est recopié en mémoire principale lorsqu'il est remplacé.

Les mémoires cache

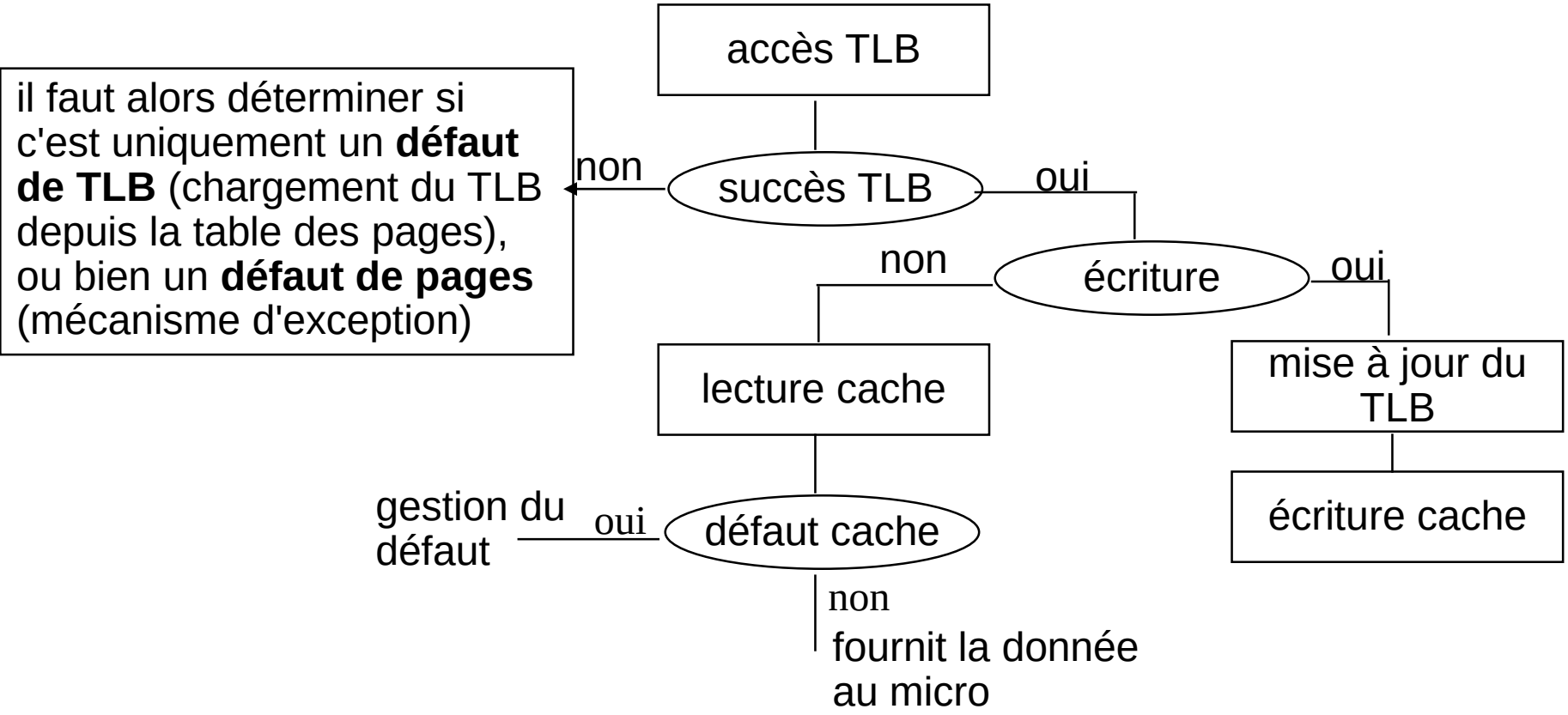
Le buffer de translation (TLB)

Définition d'une étiquette (TAG)



Les mémoires caches

Le buffer de translation (suite)



Les mémoires caches

Taille des étiquettes (TAG):

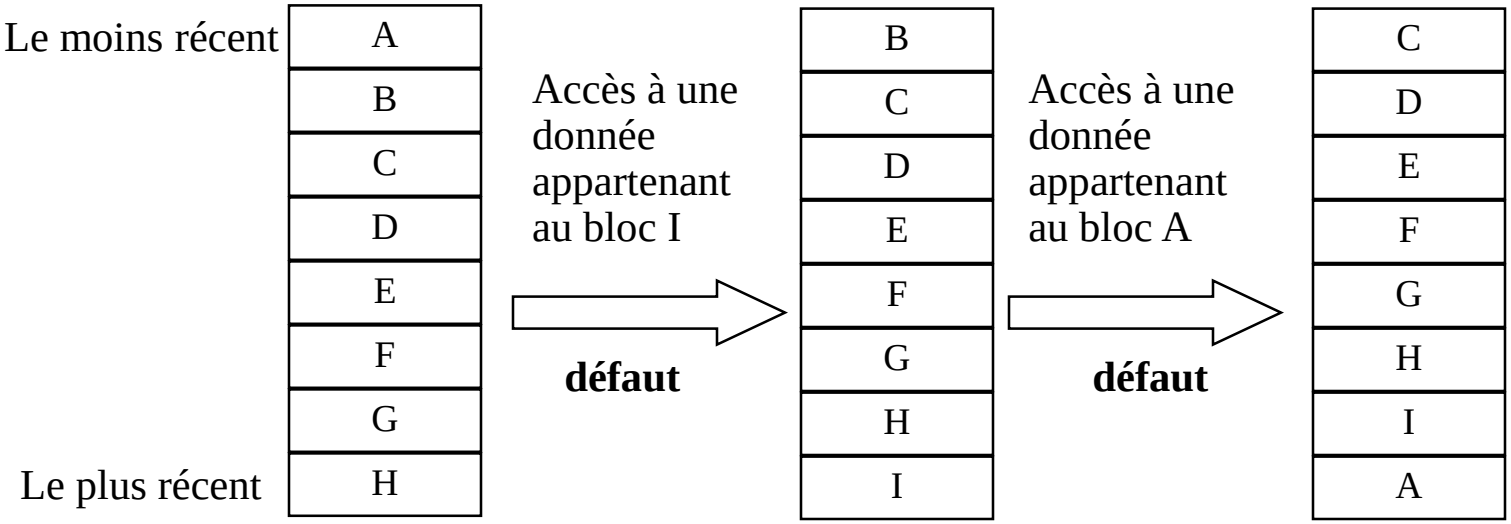
- Plus un bloc est petit et plus la gestion du cache est fine mais,
  - On augmente la taille du TAG
  - Pour une même taille de cache, on augmente le nombre de TAG à mémoriser.
- Exemple pour un cache de 1024 octets, le processeur accédant à 2^{24} octets

Taille du bloc	Nb de TAG	Taille du TAG	Mémoire TAG	Mémoire données	Rendement
1024	1	14	14 bits	8 Kbits	0.999
128	8	17	136 bits	8 Kbits	0.98
16	64	20	1280 bits	8 Kbits	0.84
1	1024	24	24576 bits	8 Kbits	-2!!!

Les mémoires caches

La gestion des défauts: les algorithmes de remplacement

- LRU, FIFO, Random



Le LRU n 'est pas toujours le plus performant !

Taille	LRU	Aléatoire
16 Ko	4,75 %	5,31 %
64 Ko	1,60 %	1,73 %
256 Ko	1,13 %	1,14 %

Taux d'échec des accès au cache dépend de la gestion des défauts

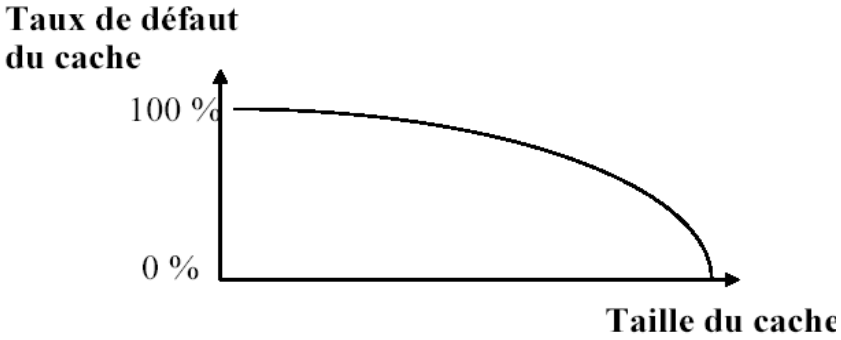
Les mémoires caches

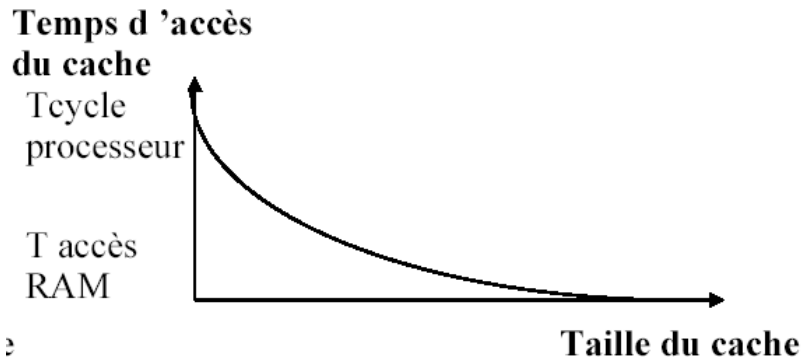
Conclusion: Tout est question de compromis

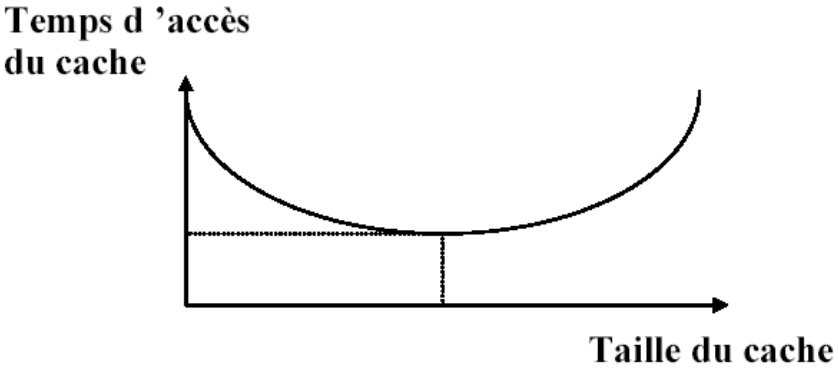
- Le nombre d'échecs diminue avec la taille du cache mais, le temps d'accès aussi
- Les performances de l'algorithme de remplacement dépendent du programme exécuté
- Une taille de bloc importante améliore l'anticipation mais, diminue les performances si le programme fait des accès désordonnés
- Une taille de bloc faible est adaptée à des accès désordonnés mais, ne l'est pas pour un programme régulier et diminue le « rendement » de la surface du cache
- Un cache associatif a, a priori, de meilleures performances mais, son temps d'accès est plus grand par rapport au Direct Mapped.

Les mémoires caches

Calcul de performances: $T_{moy} = P\ T_{cache} + (1-P)T_{sup}$







Trouver un compromis

Les mémoires caches

Calcul de performances

- Texe= (Nb cyles exécution x Tcycle + Nb cycle attente mémoire x Tattente)
- Exemple:
  - Cache1: 32K, taux de défaut 5%, temps d'accès 17ns
  - Cache2: 64K, taux de défaut 4%, temps d'accès 25ns
  - 200 instructions à exécuter; Tcycle = 5ns

  - Coût cache1 = 0.05 x 200 x 17 =170ns
  - Texe = (200 X5) + 170 = 1170ns

  - Coût cache2 = 0.04 x 200 x 25 = 200 ns
  - Texe = (200 x 5) + 200 = 1200ns

Le cache le plus gros n'est pas le plus performant : le taux de défaut de cache n'est pas tout, attention au temps d'accès

To be continued