Cours de systèmes d'exploitation centralisés

Chapitre 2

Séance 3

Gestion de la mémoire virtuelle

N.TEMGLIT, M.RAHMANI

3. La segmentation

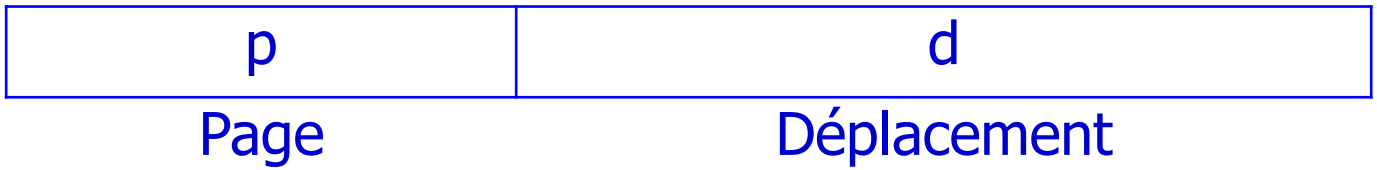
- **Pagination (Rappel)**

➢ L'utilisateur travaille avec des adresses linéaires :
Adresses comprises entre 0 et n-1.

Adresse virtuelle ou logique

➢ C'est le MMU qui divise l'adresse virtuelle en page et déplacement

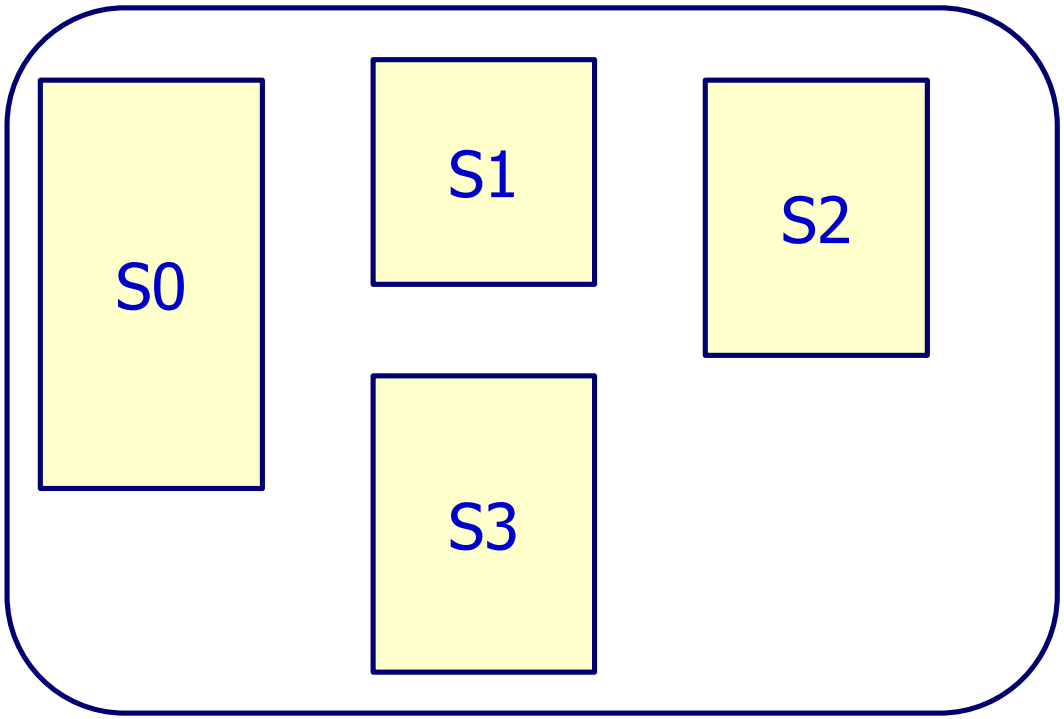
Adresse Virtuelle



➢ L'utilisateur ignore ce découpage.

3.1 Définitions

- Dans la segmentation l'espace virtuel est composé d'un ensemble de segments de tailles variables (ou tailles différentes).
- L'utilisateur voit un programme comme un ensemble de modules de tailles variables.



- Un module peut être :
  - Un module de code.
  - Un module de données.

La segmentation est une technique de gestion mémoire permettant de supporter cette vue de l'utilisateur.

- L'espace virtuel est constitué d'un ensemble de segments.
- Chaque segment est un espace d'adressage linéaire indépendant des autres segments.
- Un programme utilisateur est composé de un ou plusieurs segments.

- Chaque segment est défini par :
  - Un nom,
  - Une longueur (taille).

- Une adresse virtuelle est composée du nom du segment et d'un déplacement à l'intérieur du segment. ➔
- L'utilisateur désigne un objet dans son programme à l'aide de deux informations :

Nom du segment	Déplacement

Remarque:

- Pour simplifier l'implémentation, les segments sont numérotés et donc référencés par un numéro plutôt que par un nom : (numéro de segment, déplacement).
- La numérotation des segments est réalisée à l'assemblage ou à la compilation (complétée à l'édition de liens si le programme fait appel à des segments externes).

3.1 Représentation de l'espace virtuel d'un processus

- L'espace virtuel d'un processus est décrit à l'aide d'une table de segments.
- Chaque entrée de la table de segments contient :
  - La base (ou l'adresse début) du segment en mémoire centrale.
  - La limite (ou la longueur) du segment.

- La table de segments peut décrire :
  - ➢ tout l'espace virtuel ou
  - ➢ l'espace virtuel réellement utilisé.
- Elle contient autant d'entrées que de segments (maximum ou utilisés) de l'espace virtuel .
- Le bloc de contrôle du processus (PCB) contient un pointeur vers la table de segments.

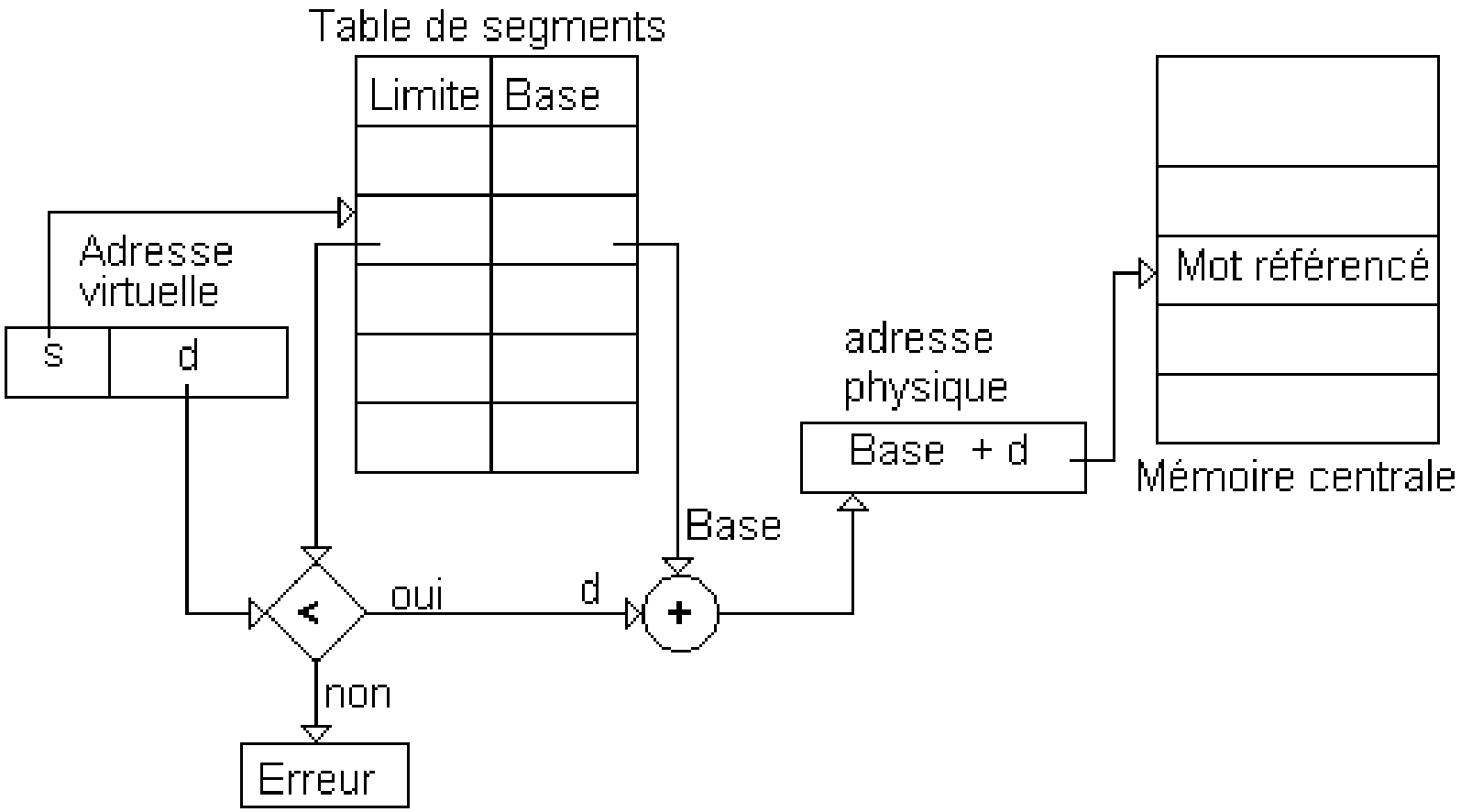
3.2 Allocation de la mémoire centrale aux segments

- Les segments sont de longueurs variables : **partitions de tailles variables** ➔
- L'allocation de la mémoire principale est donc réalisée à l'aide d'un algorithme first-fit, best-fit ou autres algorithmes.

Inconvénients : Fragmentation externe et perte de temps pour la recherche d'un bloc libre.

3.3 Traduction d'une adresse virtuelle en adresse réelle

- Une entrée de la table de segments contient :
  La taille(Limite) et l'adresses début(Base) du segment



3.4 Implémentation de la table de segments

- La table de segments peut être chargée dans un ensemble de registres rapides (base, limite) ou en mémoire centrale.
- Dans le cas où la table de segments est en mémoire centrale, deux accès sont nécessaires pour atteindre le mot référencé.

3.5 Protection

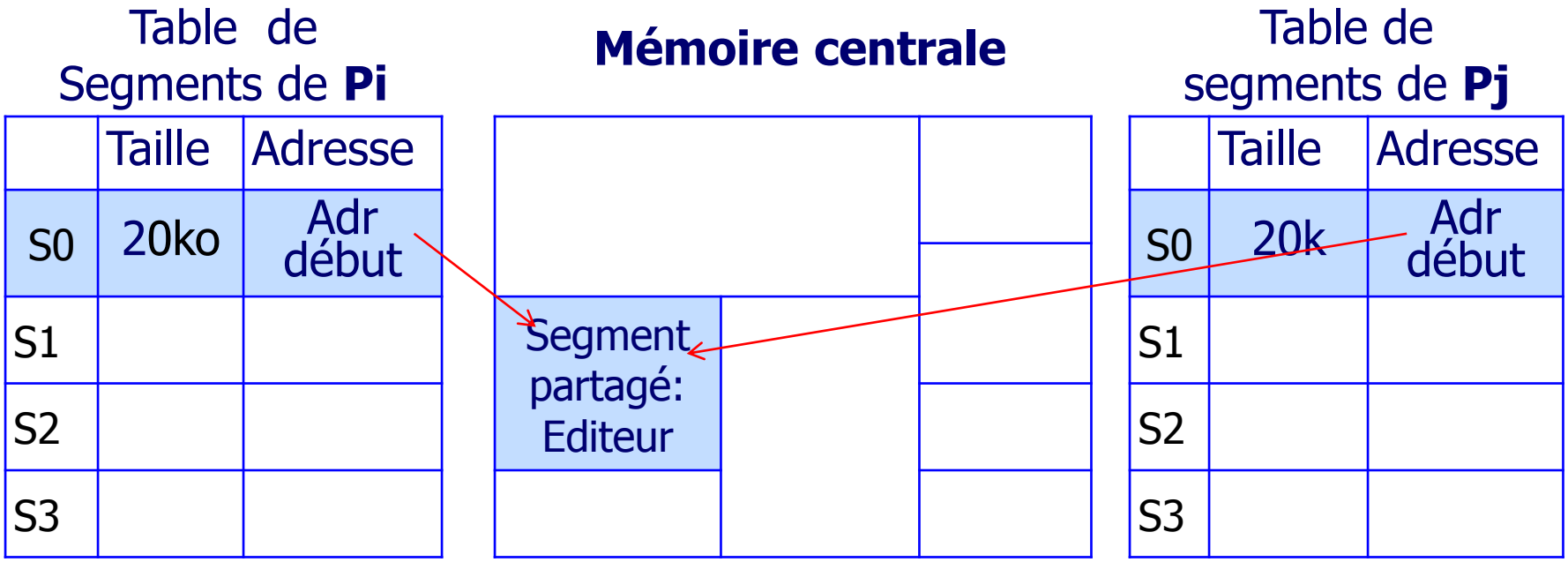
- L'un des avantages de la segmentation est le fait de pouvoir associer à chaque segment une "**protection particulière**".

  Exemple :

  - Un segment de code peut être protégé contre l'écriture mais lecture et exécution autorisées.
  - Un segment de données peut être : en lecture seule ou en lecture/écriture.

3.6 Partage de segments

- Le partage de segments est plus facile que dans la pagination.
- Pour partager un segment, chaque processus utilise, dans sa table de segments, une entrée qui pointe vers le même segment physique.



4. Segmentation avec pagination

- Principaux inconvénients de la segmentation sans pagination :
  - Fragmentation externe,
  - Temps de recherche d'un bloc mémoire libre pour un segment
- **Comment éviter la fragmentation externe ?**
  - **Paginer les segments.**
  - ✓ L'espace d'adressage virtuel est constitué d'un ensemble de segments,
  - ✓ Un segment composé d'une ou de plusieurs pages.
  - ✓ Les segments sont de tailles variables ➔ ils n'ont pas le même nombre de pages.
  - ✓ L'adresse virtuelle est composée de trois champs:(N° segment, N°page, Déplacement dans la page).

- Pagination de segment ➔ consiste à découper la partie déplacement dans le segment en deux informations:
  - Numéro de page,
  - Déplacement à l'intérieur de la page.

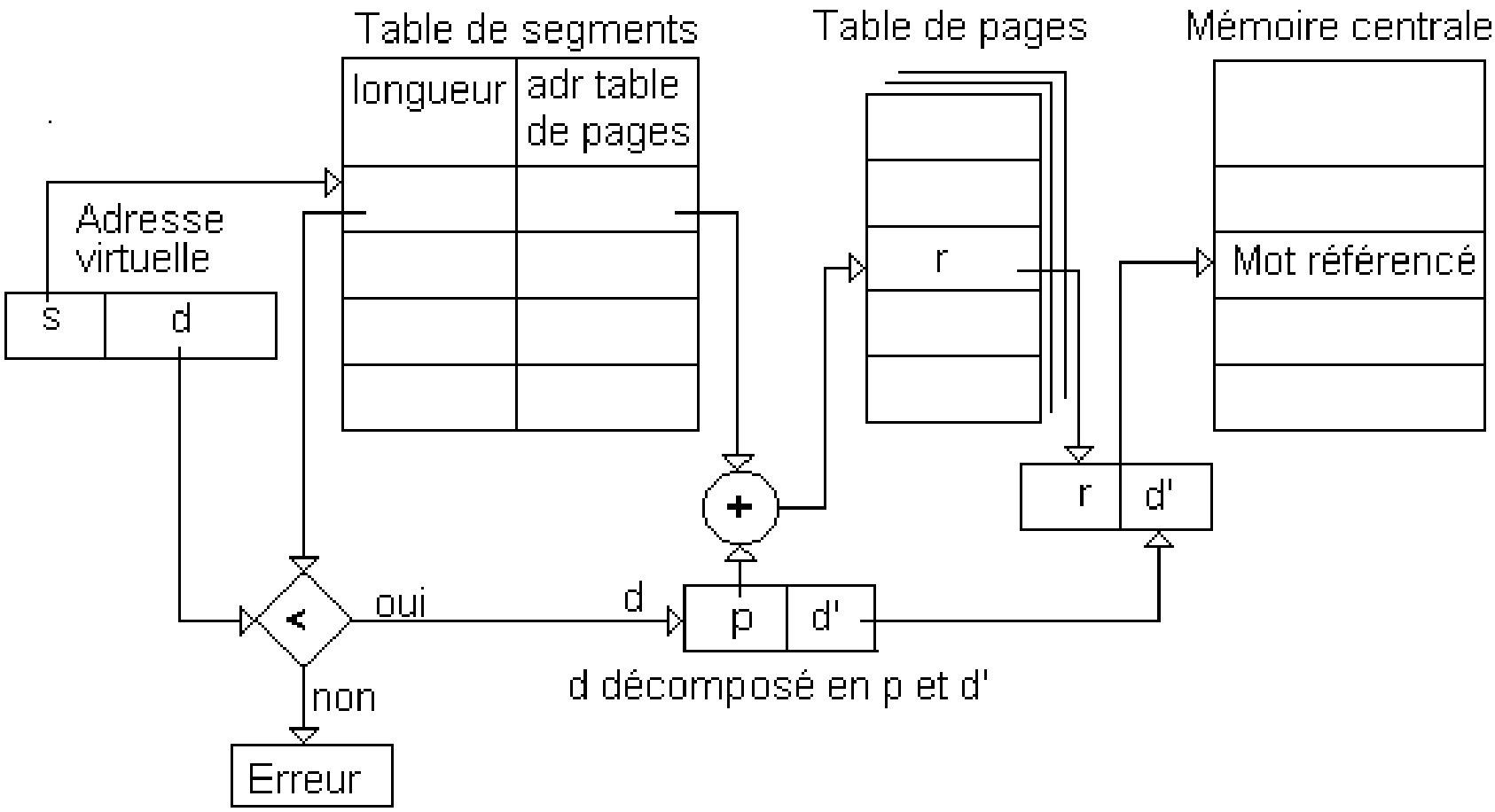
Adresse virtuelle		
Numéro de segment	Déplacement dans le segment	
	Page	déplacement

- L'espace virtuel d'un processus est décrit à l'aide de :
  - Une table de segment et
  - Une ou plusieurs tables de pages.

- La table de pages ne décrit que l'espace virtuel du segment.
- Les tables de pages des segments sont de tailles variables.
- La dernière page d'un segment est utilisée en moyenne à 50%
  ➔ En moyenne une demi-page de fragmentation interne par segment.

- Une entrée de la table de segments contient :
  - Le nombre d'entrées de la table de pages(nombre de pages du segment) ou taille du segment en octets.
  - L'adresse(N° de case ou N° de page physique) de la table de pages du segment.

4.1 Traduction d'une adresse virtuelle en adresse physique



4.2 Exemples

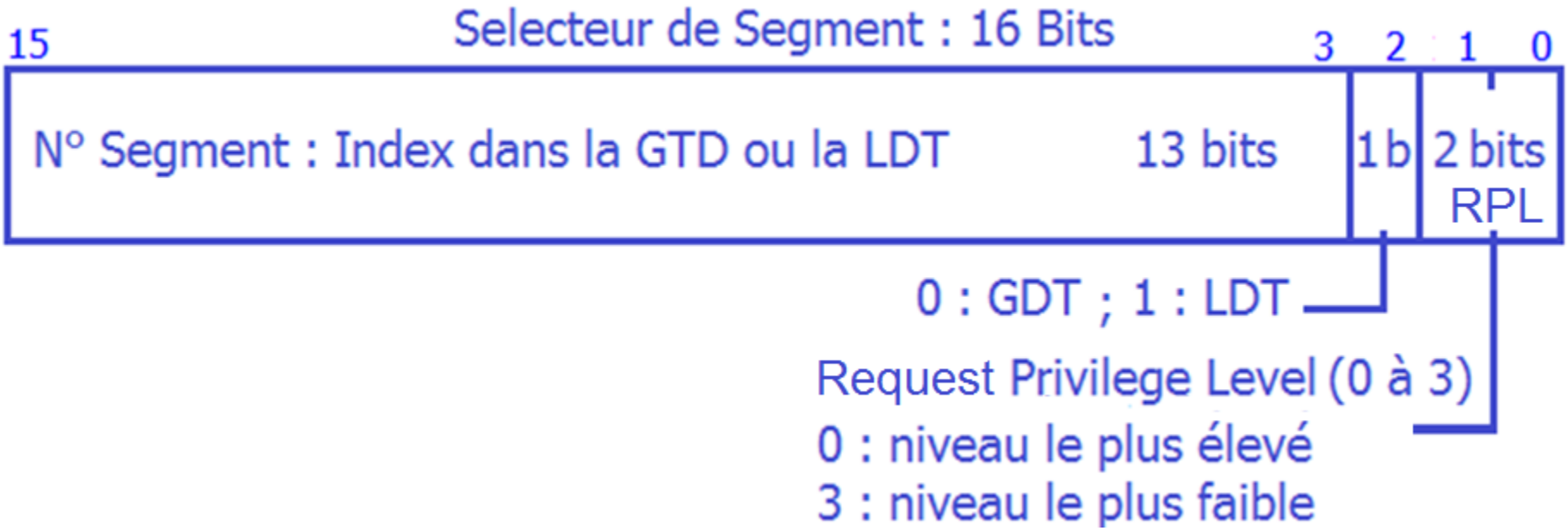
4.2.1 Segmentation et pagination de l'Intel

- Une adresse logique est composée
  - d'un sélecteur de segment sur 16 bits et
  - d'un déplacement (offset) sur 32 bits.

Sélecteur : 16 bits	Déplacement ou offset : 32 bits

- Les registres (CS, SS, DS, ES, FS et GS) contiennent des sélecteurs de segments.
  - ✓ **CS** : segment de code;
  - ✓ **SS** : Segment de pile.
  - ✓ **DS, ES, FS et GS** : segments de données.

- Le sélecteur du segment contient :
  - Niveau de privilège requis (droits d'accès): 2 bits (0-1),
  - Type de table de descripteur de segment(GDT ou LDT):1 bit
  - N° du segment : 13 bits (bits 3-15).



- L'espace virtuel d'un processus est décrit à l'aide d'une table locale de descripteurs de segments que l'on appelle LDT(Local Descriptor Table) : **Il y a une LDT par processus**.

- La table globale des descripteurs que l'on appelle GDT(Global Descriptor Table) peut être utilisée par tous les processus : **Il y a une seule GDT**.
- Chaque entrée de la table des descripteurs de segments (GDT ou LDT) contient un descripteur de segment (voir l'annexe).
- Un descripteur est composé de 8 octets:
- Toutes les LDT's ont leur descripteur de segment dans la GDT.
- Un processus peut référencer(adresser) jusqu'à $2*2^{13}$ segments de 32 bits.

Adresses et tailles des Tables GDT et LDT

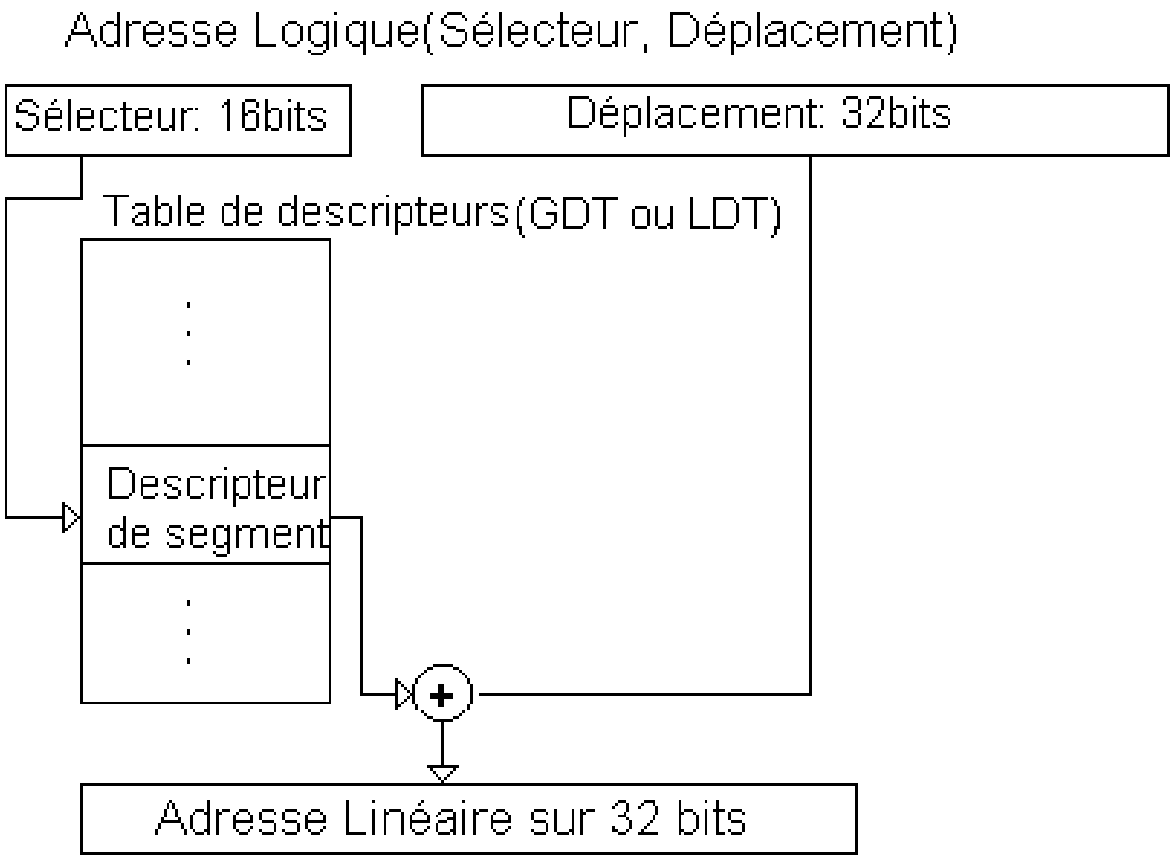
- Le registre **GDTR** : Contient l'adresse et la taille de la **GDT**.
- Le registre **LDTR** : Contient l'adresse et la taille de **LDT**.

- Puisque les registres GDTR et LDTR contiennent les tailles des tables de descripteurs ➔
  - La LDT ne décrit que l'espace réellement utilisé par un processus.
  - La GDT ne décrit que l'espace utilisé par le système et les processus utilisateurs.

- Toutes les LDT's ont leur descripteur de segment dans la GDT.

Traduction d'une adresse logique en adresse physique

1) Segmentation sans pagination :



- Adresse contenu dans le descripteur = **adresse physique**.
- Adresse linéaire = **adresse physique(ou réelle)**.

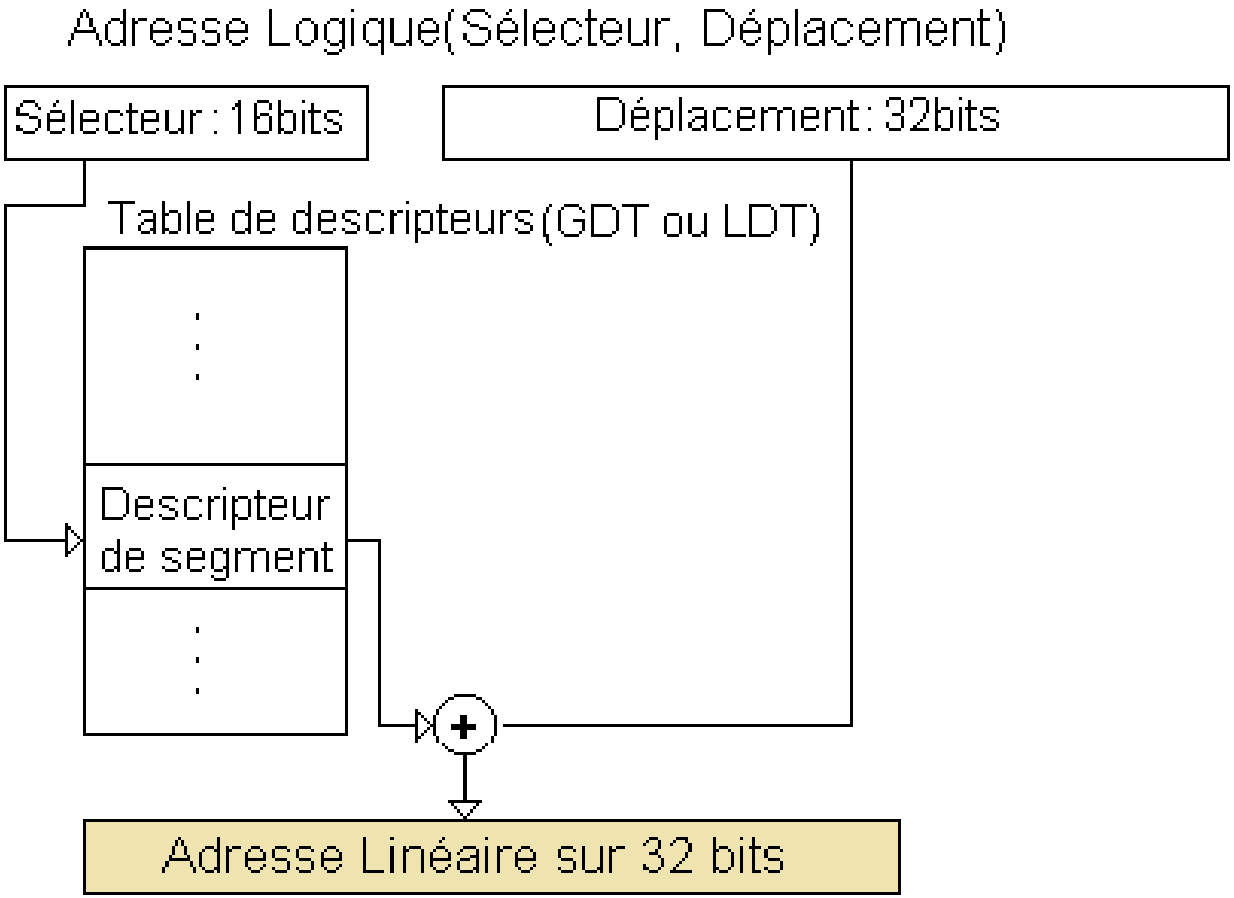
2) Segmentation et pagination

- **L'Intel 80386** utilise la pagination à deux niveaux.
  La traduction d'une adresse est réalisée en deux étapes:

  1. L'adresse logique (sélecteur de segment, déplacement) est traduite en une adresse linéaire (ou adresse virtuelle) sur 32 bits qui considèrent que l'espace virtuel est linéairement paginé.
  2. L'adresse linéaire est traduite en adresse physique.

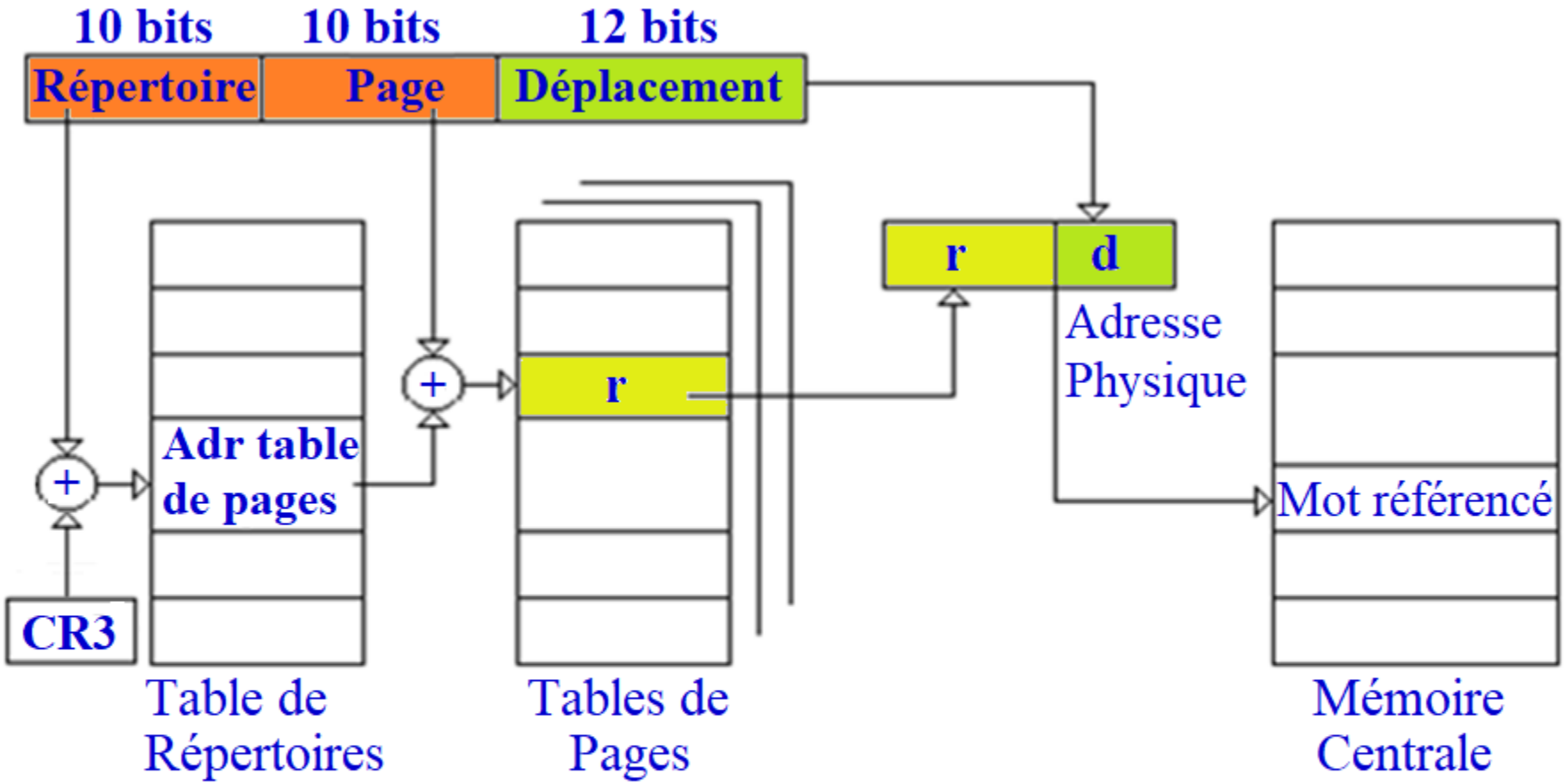
- **Pagination à deux niveaux de l'espace linéaire (32 bits)**
  - Adresse linéaire (virtuelle) sur 32 bits,
  - Page : 4096 octets,
  - Adresse virtuelle composée de trois champs : Répertoire, page et déplacement.
  - Adresse physique ou réelle : 32 bits.

Etape1 : Traduction de l'adresse logique (sélecteur de segment, déplacement) en une adresse linéaire sur 32 bits.



- Adresse contenu dans le descripteur = **adresse virtuelle**.
- Adresse linéaire = **adresse virtuelle**.

Etape2 : Traduction de l'adresse linéaire en adresse physique

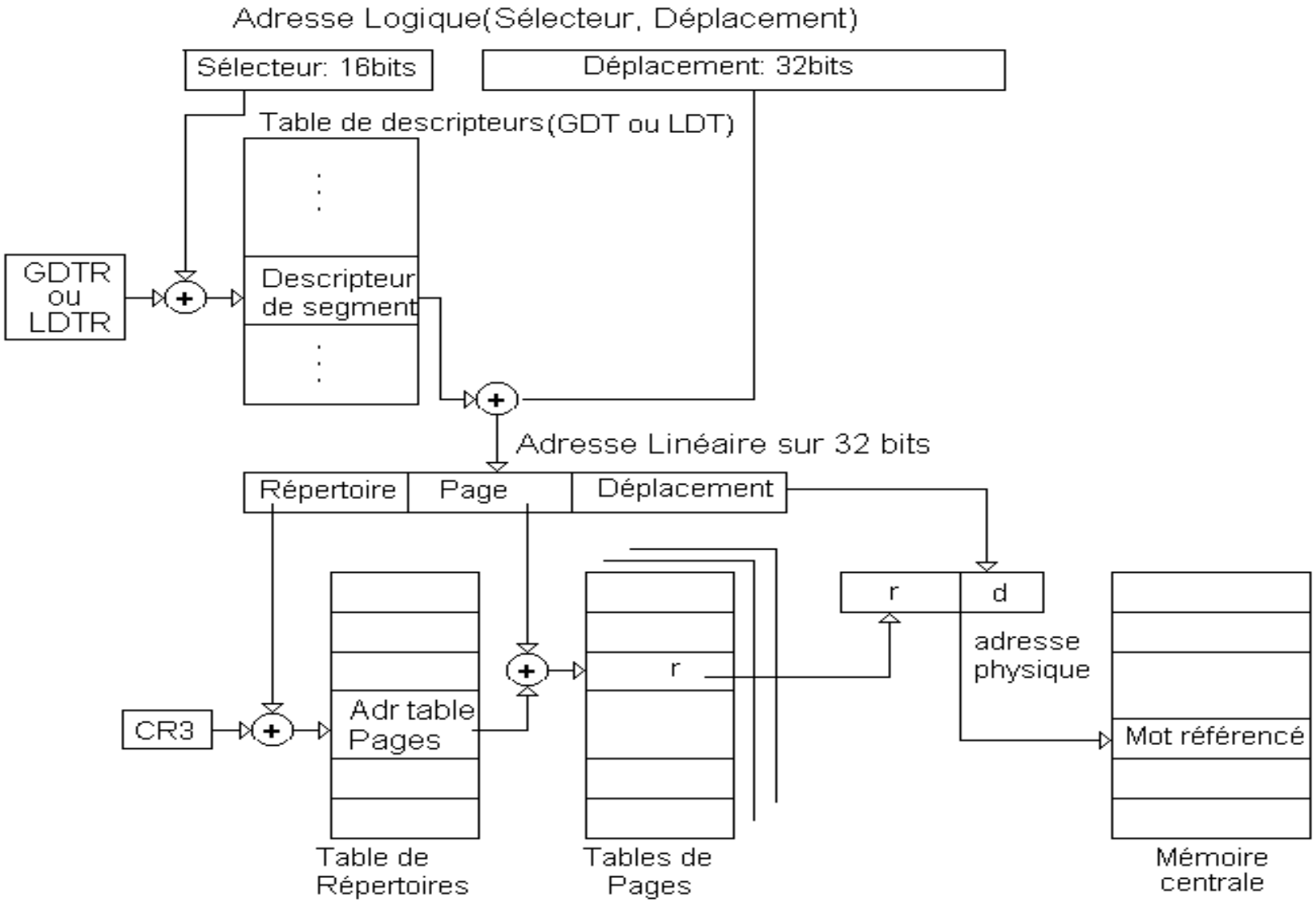


Toutes les tables doivent commencer sur une frontière de page physique

Adr table de pages = N° de la page physique de la table de pages

CR3 = N° de la page physique de la table des répertoires

Traduction d'une adresse logique en adresse physique



Résumé de la pagination des processeurs Intel

a) Pagination des processeurs Intel 32bits (Intel IA32)

Adresse logique : Sélecteur de segment sur 16 bits et
Déplacement à l'intérieur du segment sur 32 bits.

Sélecteur : 16 bits	Déplacement ou offset : 32 bits

	Taille de l'adresse linéaire	**Taille de la page**	**Taille de l'adresse physique**	**Niveaux de pagination**
32 bits sans PAE	32 bits	4Ko	32 bits	2
32 bits avec PAE	32 bits	4Ko	36 bits	3
À partir du Pentium Pro	32 bits	2Mo	36 bits	2
	32 bits	4Mo	36 bits	1

PAE: Physical Address Extension
IA-32 Intel® Architecture Software Developer's Manual
Volume 3: System Programming Guide (Order Number 245472-012)

b) Pagination des processeurs Intel 64 et 32bits (Intel IA32e)

Adresse logique : Sélecteur de segment sur 16 bits et
Déplacement à l'intérieur du segment sur 64 bits.

Sélecteur : 16 bits

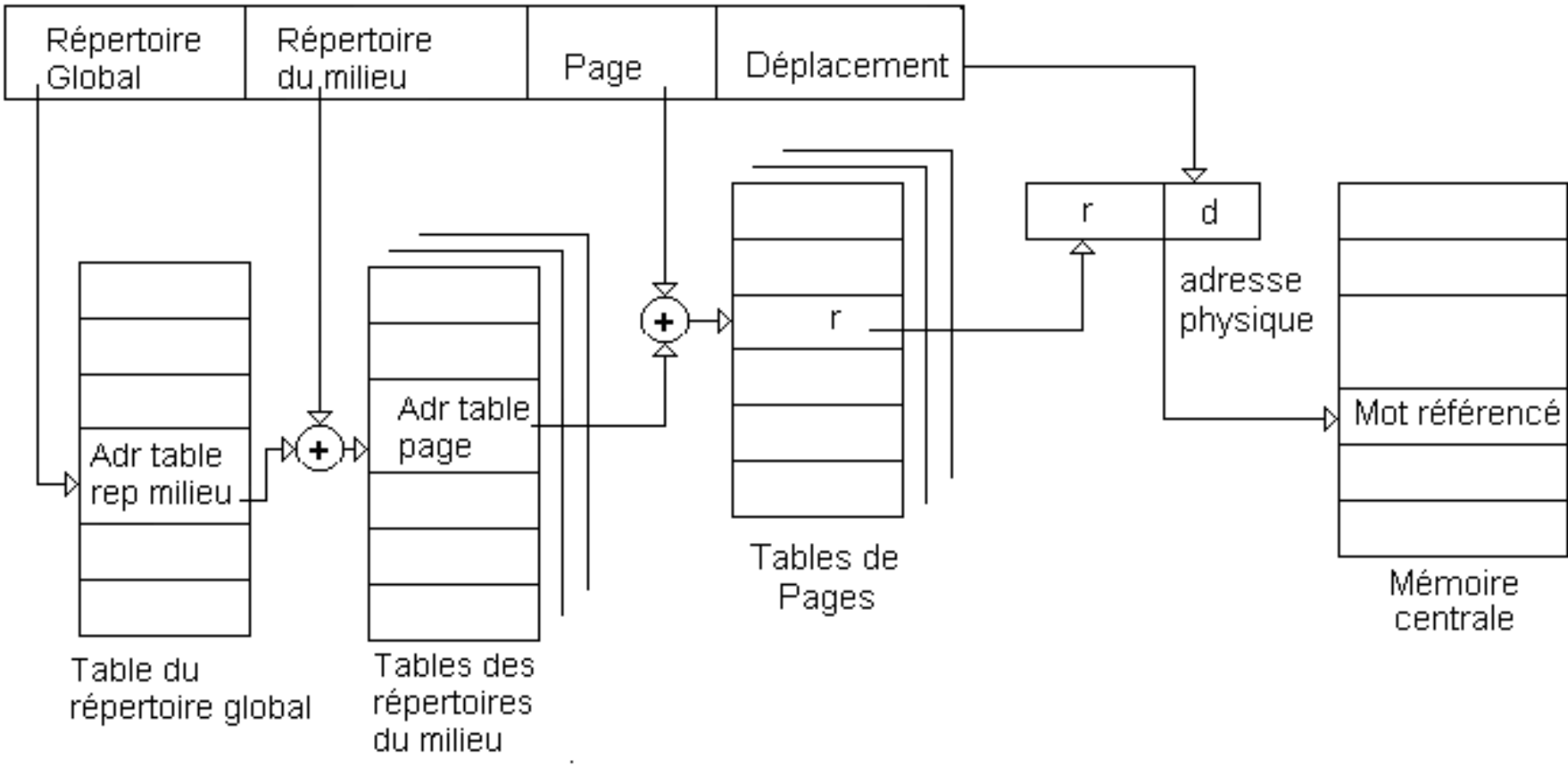
Déplacement ou offset : 64 bits

	Taille de l'adresse linéaire	Taille de la page	Taille de l'adresse physique	Niveaux de pagination
32 bits sans PAE	32 bits	4 Ko	32 bits	2
	32 bits	4 Mo	40 bits si PSE-36 supporté	1
32 bits avec PAE	32 bits	4 Ko	Jusqu'à 52 bits	3
	32 bits	2 Mo	Jusqu'à 52 bits	2
IA-32e (64 bits)	48 bits	4 Ko	Jusqu'à 52 bits	4
	48 bits	2 Mo	Jusqu'à 52 bits	3
	48 bits	1 Go	Jusqu'à 52 bits	2

Intel® 64 and IA-32 Architectures Software Developer's Manual
Volume 3A: System Programming Guide, Part 1 (Order Number 253668)

4.3.2 Pagination dans le système Linux jusqu'à la version 2.6.10

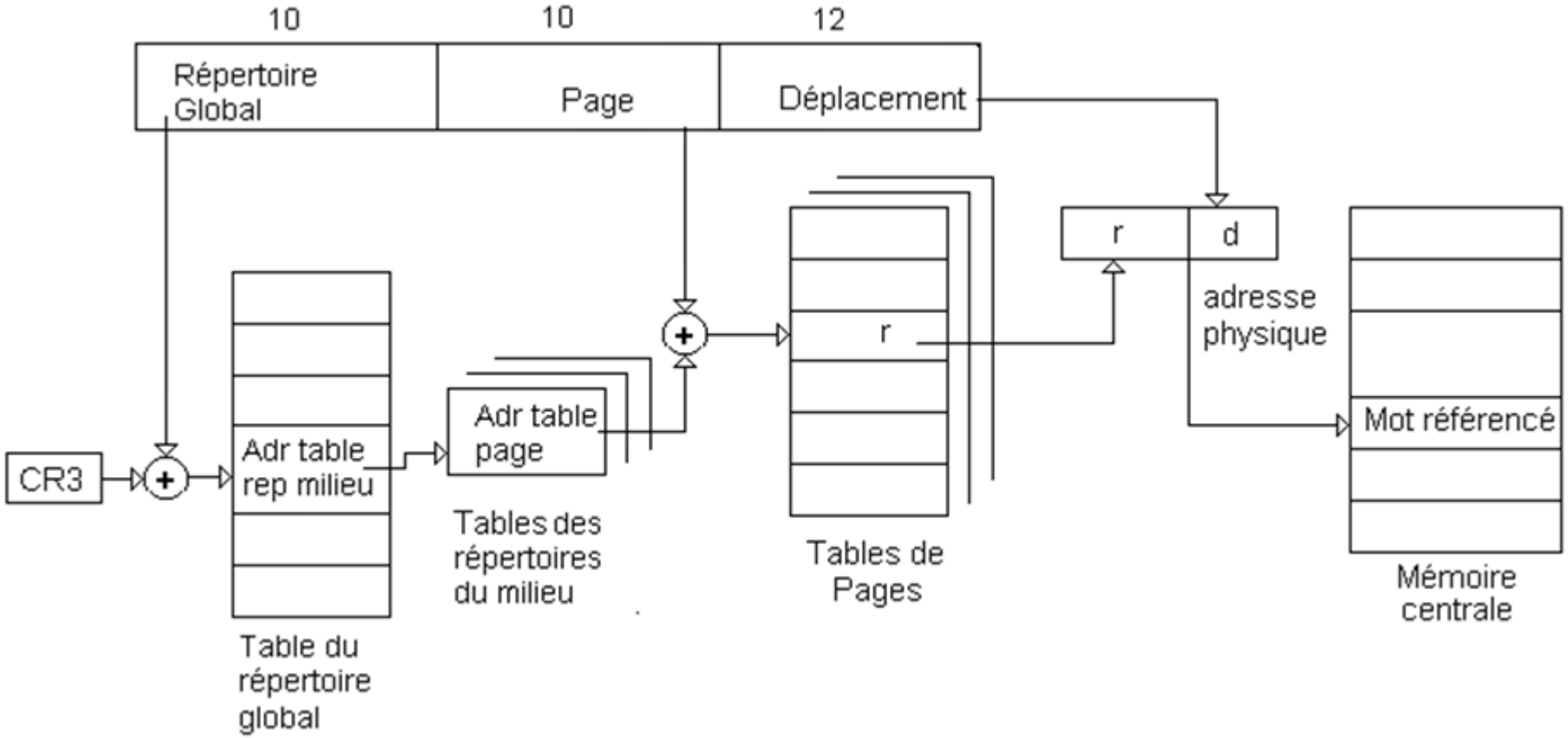
- Le système Linux utilise la pagination à **3 niveaux**.



Implantation de linux(version ≤ 2.6.10) sur une machine Intel 80x86

- Les architectures Intel 32 bits utilisent la pagination à deux niveaux.
- Pour adapter la pagination à 3 niveaux du système Linux à celle des processeurs Intel, le gestionnaire de la mémoire Linux considère que la taille du répertoire du milieu est égale à zéro :

Implantation de linux sur une machine Intel 80x86 (32bits)



4.3.2 Pagination 4 niveaux de Linux depuis la version 2.6.11

- Une adresse virtuelle est composée de 5 champs:
  - Répertoire global (Global Directory),
  - Répertoire Supérieur(Upper Directory),
  - Répertoire du milieu (Middle Directory) ,
  - Page,
  - Déplacement à l'intérieur de la page.

Adresse virtuelle

Répertoire global	Répertoire supérieur	Répertoire du milieu	Page	Déplacement

2.6 Mémoire associative ou registres associatifs

(**T**ranslation **L**ook-aside **B**uffers : **TLB**)

- Les tables de pages sont, en général, stockées en mémoire centrale. Pour accéder à un mot mémoire, on doit faire:
  - ❖ 2 accès mémoires : Pagination à un niveau.
  - ❖ 3 accès mémoires : Pagination à deux niveaux.
  - ❖ n+1 accès mémoires : Pagination à n niveaux.
- En observant des processus en exécution, On a constaté que:
  - ➢ Pendant un intervalle de temps, la plupart des processus effectuent un grand nombre de références à un petit nombre de pages **: propriété de localité**. ➔
  - ✓ Seule une petite fraction, des entrées des tables de pages, est souvent lue(utilisée);
  - ✓ Les autres entrées sont rarement utilisées.

Comment réduire le temps d'accès à la mémoire principale ?

Solution :

- Sauvegarder les numéros de cases correspondant aux pages fréquemment utilisées:
  - Première référence à un mot : 1 accès à la table de pages
  - Accès suivant(s) : Eviter l'accès à la table de pages.

- Pour éviter l'accès à la table de pages ➔ **Utiliser une mémoire cache** qui permet de traduire les adresses virtuelles en adresses physiques sans passer par la table de pages.
- Ce composant, appelé mémoire associative ou registres associatifs (Translation Look-aside Buffers : TLB), est un ensemble de registres à accès rapide.

- Une entrée de la mémoire associative contient :
  - Le numéro de la page : **p**,
  - Le numéro de la case **r** correspondant à la page **p**.
  - Indicateurs (bits de validité, de modification, de référence,…).

- La mémoire associative est plus rapide que la mémoire principale et permet un accès parallèle (simultané) à l'ensemble des registres.

1. Traduction d'une adresse virtuelle en adresse réelle en utilisant la mémoire associative

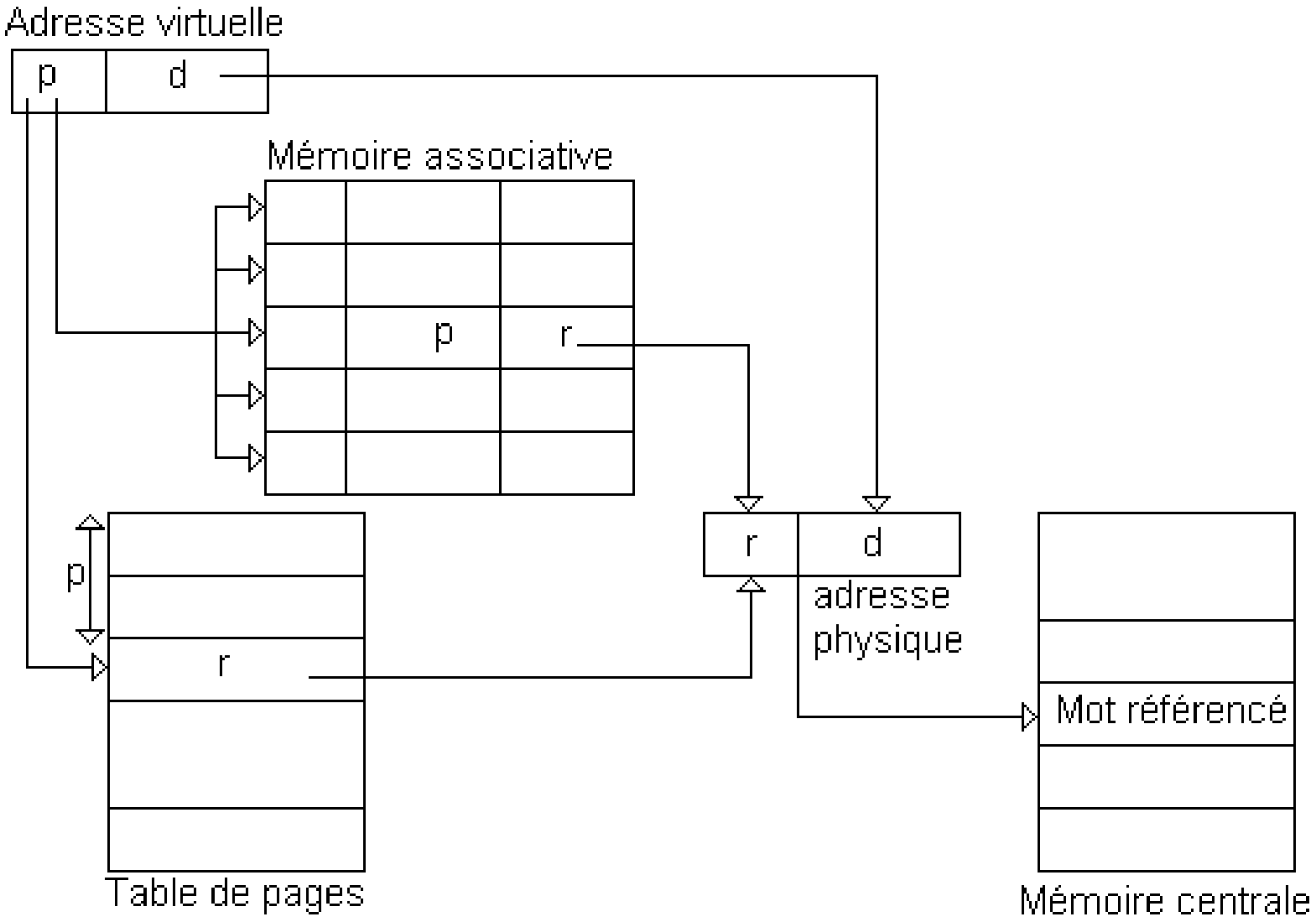
- Le numéro de la page est comparé simultanément à toutes les entrées de la mémoire associative.
- Si le numéro de page **p** existe dans la mémoire associative ➔
  - ✓ Prendre le numéro de case **r** correspondant directement de la mémoire associative sans passer par la table de pages.
- Si le numéro de page **p** ne se trouve pas dans la mémoire associative ➔
  - ✓ Effectuer un accès à la table de pages pour prendre le numéro de case **r** correspondant.
  - ✓ Le numéro de page **p** et le numéro de case **r** correspondant (**p, r**) sont ajoutés à la mémoire associative.

- Si la mémoire associative est pleine, le système choisit une entrée, selon une stratégie de remplacement(ex: LRU), et la remplace par l'entrée nouvellement trouvée.

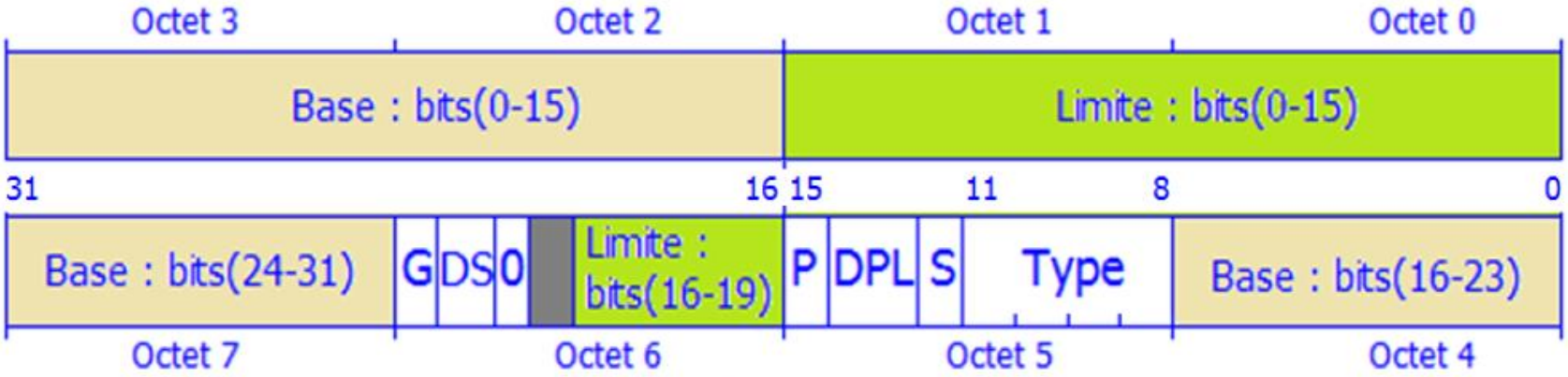
Remarque :

- A chaque commutation de contexte, la mémoire associative doit être remise à zéro car on doit charger la table de pages du processus élu.

- Nombre de registres associatifs : 8 à 128.


Adresse virtuelle
p
d
Mémoire associative
p
r
p
r
r
d
adresse
physique
Mot référencé
Table de pages
Mémoire centrale

Annexe: Descripteur de segment de la machine Intel



Descripteur de segment

- **Base** : Adresse du segment sur 32 bits.
  Composée de 3 champs: octets 2 et 3, octet4, octet 7.
- **Limite** : Taille du segment sur 20 bits.
  Composée de 2 champs: octets 0 et 1 et les bits[0..3] de l'octet6.

  Elle peut donc varier :
  de 1 à 1 Mo si G=0 (segment sans pagination) ou
  de 4K à 4 Go si G=1 (segmentation avec pagination).

- **Bit de Granularité (G)** :

  G=0 ➔ Taille du segment en octets (max = 2^{20} octets).

  G=1 ➔ Taille du segment en page de 4ko (max=$2^{20}*2^{12}=2^{32}$ o).

- **Taille par défaut du segment (DS)** :

  DS=0 ➔ Taille maxi d'un segment sur 16 bits

  DS=1 ➔ Taille maxi d'un segment sur 32 bits.

- **Niveau de privilège du segment** (**DPL** : Descriptor Privilege Level) : Indique le niveau de protection du segment.

  Si Max(**CPL, RPL**) **≤ DPL** ➔

  accéder au segment

  Sinon

  pas d'accès (erreur).

- **Bit de Présence du Segment (P) :**

  P=0 ➔ le segment n'est chargé en mémoire centrale;

  P=1 ➔ le segment est en mémoire centrale.

- **Type de descripteur (S) :**

  S= 0 ➔ descripteur de segment système

  S=1 ➔ Descripteur de segment de code ou de données (application utilisateur).

- **Type de Segment (Type) :**

  Si « **S=1** » ➔ Type d'accès au segment de code ou segment de données (pour plus de détail voir tableau ci-dessous).

Annexe: Descripteur de segment de la machine Intel

Code and Data Segment Types

Bit 11	Bit 10 E	Bit 9 W	Bit 8 A		Description
0	0	0	0	**Data**	Read-Only
0	0	0	1		Read-Only, accessed
0	0	1	0		Read/Write
0	0	1	1		Read/Write, accessed
0	1	0	0		Read-Only, expand-down
0	1	0	1		Read-Only, expand-down, accessed
0	1	1	0		Read/Write, expand-down
0	1	1	1		Read/Write, expand-down, accessed
	C	**R**	**A**		
1	0	0	0	**Code**	Execute-Only
1	0	0	1		Execute-Only, accessed
1	0	1	0		Execute/Read
1	0	1	1		Execute/Read, accessed
1	1	0	0		Execute-Only, conforming
1	1	0	1		Execute-Only, conforming, accessed
1	1	1	0		Execute/Read-Only, conforming
1	1	1	1		Execute/Read-Only, conforming, accessed

The Intel Architecture Software Developer's Manual, Volume 3:
System Programing Guide (Order Number 243192).