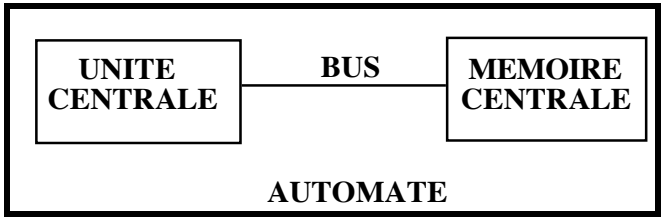
OBJETS CONTENUS EN MÉMOIRE CENTRALE

- OBJETS PASSIFS : répertoires, fichiers, tampons de données d'entrées-sorties ou de transfert réseau,

- OBJETS ACTIFS : processus.

• contraintes de placement ou de mobilité si : références internes sous forme d'adresse, processus

LA MÉMOIRE DANS LE SCHÉMA D'UN ORDINATEUR (VON NEUMAN)



1ère idée : PROGRAMME ENREGISTRÉ

2ème idée : MÉMOIRE ADRESSABLE

On ne désigne pas une valeur mais l'adresse mémoire où elle est rangée.

```
/* en C */
main()
{
  int i, j, k ;
  i = 10 ;
  j = i + 15 ;
  k = j*i + 3 ;
}
```

```
/* en ADA*/
procedure MAIN is
    i, j, k : INTEGER;
begin
    i := 10;
    j := i + 15;
    k := j * i + 3;
end MAIN;
```

```
/* instructions machine (assembleur)
On réserve trois adresses (virtuelles)
pour les résultats nommés i, j et k.
Range 10 à l'adresse réservée pour i.
Lit le contenu de i, y ajoute 15 et range le
résultat à l'adresse réservée pour j.
Lit les contenus de i et de j, fait le calcul et
range le résultat à l'adresse réservée pour k
```

• les instructions et données que traite l'unité centrale doivent être placés en mémoire centrale au moment où elle les référence.

• Deux rôles pour la mémoire : repérer les objets (espace d'adressage), stocker les valeurs (mémoire physique)

RÔLES DE LA FONCTION MÉMOIRE POUR L'EXÉCUTION D'UN PROGRAMME

1. FAIRE L'INVENTAIRE = CLASSER ET ÉNUMÉRER TOUS LES ÉLÉMENTS QUE POURRAIT RÉFÉRENCER LE PROGRAMME

• instructions, pointeurs et données du programme, bibliothèques appelées statiquement ou dynamiquement, directement ou par fermeture transitive, environnement, objets externes

ESPACE D'ADRESSAGE VIRTUEL : associe un numéro unique (appelé adresse virtuelle) à chaque objet.

• espace d'adressage linéaire (on dit aussi que c'est un espace plat, "one level flat address space")

• espace d'adressage segmenté avec deux index, un numéro de segment et une adresse dans le segment.

FONCTION CALCUL D'ADRESSE : dans l'unité centrale, fournit l'adresse effective (virtuelle) de l'objet concerné dont la valeur est à stocker ou à rechercher en mémoire. Une unité centrale ne désigne jamais une valeur, mais donne son index de classement dans l'espace d'adressage virtuel.

Le classement des objets d'un processus est préparé tout au long de la chaîne de production de programme (compilation, édition de liens, chargement, lancement).

TAILLE DU PROCESSUS : taille (en octets) de l'espace d'adressage nécessaire pour classer tous les objets.

L'espace d'adressage virtuel et les conventions de son utilisation doivent être connus et respectés par tous les intervenants qui préparent ou exécutent le processus :

- l'architecture du processeur : adresses d'interruption, de récupération des appels systèmes, ...

- le système d'exploitation : frontière mode maître - mode utilisateur, protections, gestion mémoire,...

- la chaîne de production de programme : zone de code, zone de constantes, zone de données statiques, zone de données dynamiques, zone de la pile d'exécution.

RÔLES DE LA FONCTION MÉMOIRE POUR L'EXÉCUTION D'UN PROGRAMME

2. CONSERVER ET STOCKER LES VALEURS DES OBJETS

LA VALEUR d'un objet manipulé pendant l'exécution d'un programme doit être stockée en mémoire centrale physique pour pouvoir être relue ou modifiée par l'unité centrale. Ce lieu de stockage est divisé en contenants de la taille du composant élémentaire, en général l'octet (en fait des groupes d'octets consécutifs). Ce lieu de stockage est géré en utilisant les adresses de contenants physiques. La valeur d'un objet est fournie à l'unité centrale en allant lire le contenu d'un contenant dont on a l'adresse de stockage physique. De même l'unité centrale range la valeur d'un objet numéroté (ils le sont tous) dans un contenant d'adresse de stockage donné.

Certaines valeurs des objets d'un processus (code, constantes, variables initialisées) sont préparées par la chaîne de production de programme (compilation, édition de liens, chargement, lancement). Elles sont conservées dans des fichiers et servent à élaborer l'IMAGE EXÉCUTABLE INITIALE DU PROCESSUS.

Cette image est stockée en MÉMOIRE SECONDAIRE (en Unix c'est un fichier d'un répertoire /bin) et est CHARGÉE totalement ou partiellement en mémoire centrale avant le lancement du processus.

D'autres valeurs sont élaborées par le processus pendant son exécution et sont donc stockées en premier lieu en mémoire centrale.

RÔLES DE LA FONCTION MÉMOIRE POUR L'EXÉCUTION D'UN PROGRAMME

PLACEMENT DES OBJETS CLASSÉS POUR UN PROCESSUS

• Relation entre classement et stockage.
• Passage de la mémoire virtuelle à un rangement physique dans la mémoire centrale.

a) STOCKAGE ET CLASSEMENT CONFONDUS

Quand l'ordre de classement est aussi l'ordre de stockage en mémoire, cela introduit trois rigidités :
- Les valeurs doivent être rangées selon des adresses contiguës en mémoire physique (à cause du compteur ordinal)
- Les valeurs des objets d'un processus ne peuvent pas être déplacées sur leur support sans modifier aussi le classement de ces objets. Seule solution : tout déplacer en bloc, à condition d'avoir un registre de translation.
- L'espace d'adressage ne peut pas être plus grand que la mémoire centrale. On triche en faisant du recouvrement ("overlay") dans l'espace d'adressage virtuel.

adresse physique = adresse virtuelle

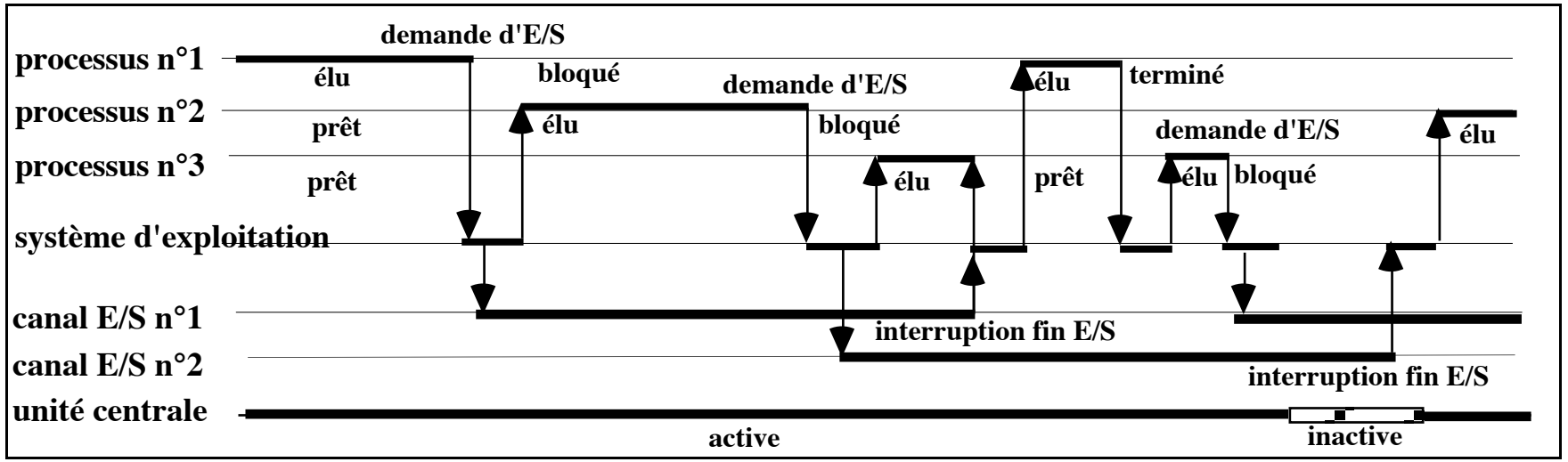
b) INDÉPENDANCE ENTRE STOCKAGE ET CLASSEMENT

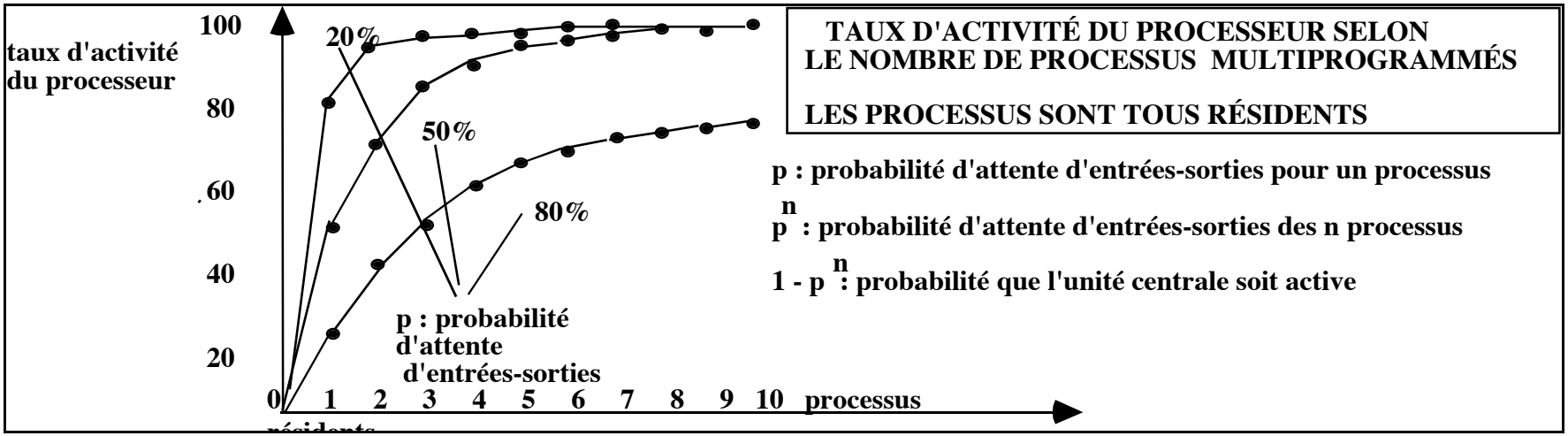
On introduit une fonction de placement de l'espace d'adressage virtuel dans l'espace de stockage en mémoire centrale (pagination et mappage défini avec l'aide d'une MMU). On établit aussi un placement de l'espace d'adressage virtuel dans un espace de stockage en mémoire secondaire qui contient l'image exécutable initiale du processus et éventuellement l'image courante quand tous les objets ne peuvent être stockés en mémoire centrale (disque de va et vient).

Adresse physique = f(adresse virtuelle)

MONOPROGRAMMATION OU MULTIPROGRAMMATION

but : augmenter le taux d'utilisation de l'unité centrale quand les programmes font des entrées-sorties





TAUX D'ACTIVITÉ DU PROCESSEUR SELON LE NOMBRE DE PROCESSUS MULTIPROGRAMMÉS

LES PROCESSUS SONT TOUS RÉSIDENTS

p : probabilité d'attente d'entrées-sorties pour un processus

p^n : probabilité d'attente d'entrées-sorties des n processus

$1 - p^n$: probabilité que l'unité centrale soit active

NATURE DU PLACEMENT EN MÉMOIRE CENTRALE

• Placement complet au démarrage du processus, donc localisation unique et définitive
1 processus seulement en mémoire centrale (monoprogrammation)
plusieurs processus en mémoire centrale (multiprogrammation)
>> placement statique réalisé par le chargeur

• Placement complet à chaque élection de processus, donc localisation provisoire et va et vient
>> placement dynamique complet qui modifie la localisation

• Placement partiel au démarrage du processus et placement de morceaux complémentaires (segments ou pages) à la demande du processus en exécution
>> placement dynamique par morceaux à chaque demande du processus

POLITIQUE DE PLACEMENT PAR ZONES.

• L'image exécutable est placée en un seul bloc d'adresses contiguës en mémoire centrale. Il faut alors translater toutes les adresses de l'image pour la placer dans le bloc libre.

>> translation dynamique d'adresse par registre de base, contrôle d'accès dynamique par registre limite

POLITIQUE DE PLACEMENT PAR PAGINATION

• L'espace d'adressage et l'image sont découpés en pages de taille fixe. Chaque page est rangée dans une case de mémoire centrale. Les cases ne sont pas nécessairement contiguës, mais une transformation topographique des adresses permet de conserver la contiguïté logique des adresses virtuelles.

>> placement dynamique par mappe mémoire cablée (MMU "Memory Management Unit")

• L'image peut être placée complètement (mais par pages non contiguës) en mémoire centrale. Mais elle peut aussi n'y être placée que partiellement, ce qui permet d'avoir des images beaucoup plus grandes que la mémoire physique installée. Il faut alors que l'unité centrale puisse détecter qu'une page n'est pas chargée dans une case de mémoire centrale, et que ce défaut de page alerte le système pour qu'il charge la page manquante. C'est le mécanisme de pagination à la demande avec va et vient. Le temps d'exécution d'un processus en pagination à la demande varie avec le nombre de défauts de pages. La politique de gestion de la mémoire centrale a aussi pour objectif d'en limiter le nombre.

GESTION DE LA MÉMOIRE SECONDAIRE

La mémoire secondaire doit contenir l'image exécutable initiale, l'image exécutable sauvegardée en cas de va et vient ("swapping"), puis l'image finale résultat ("dump").

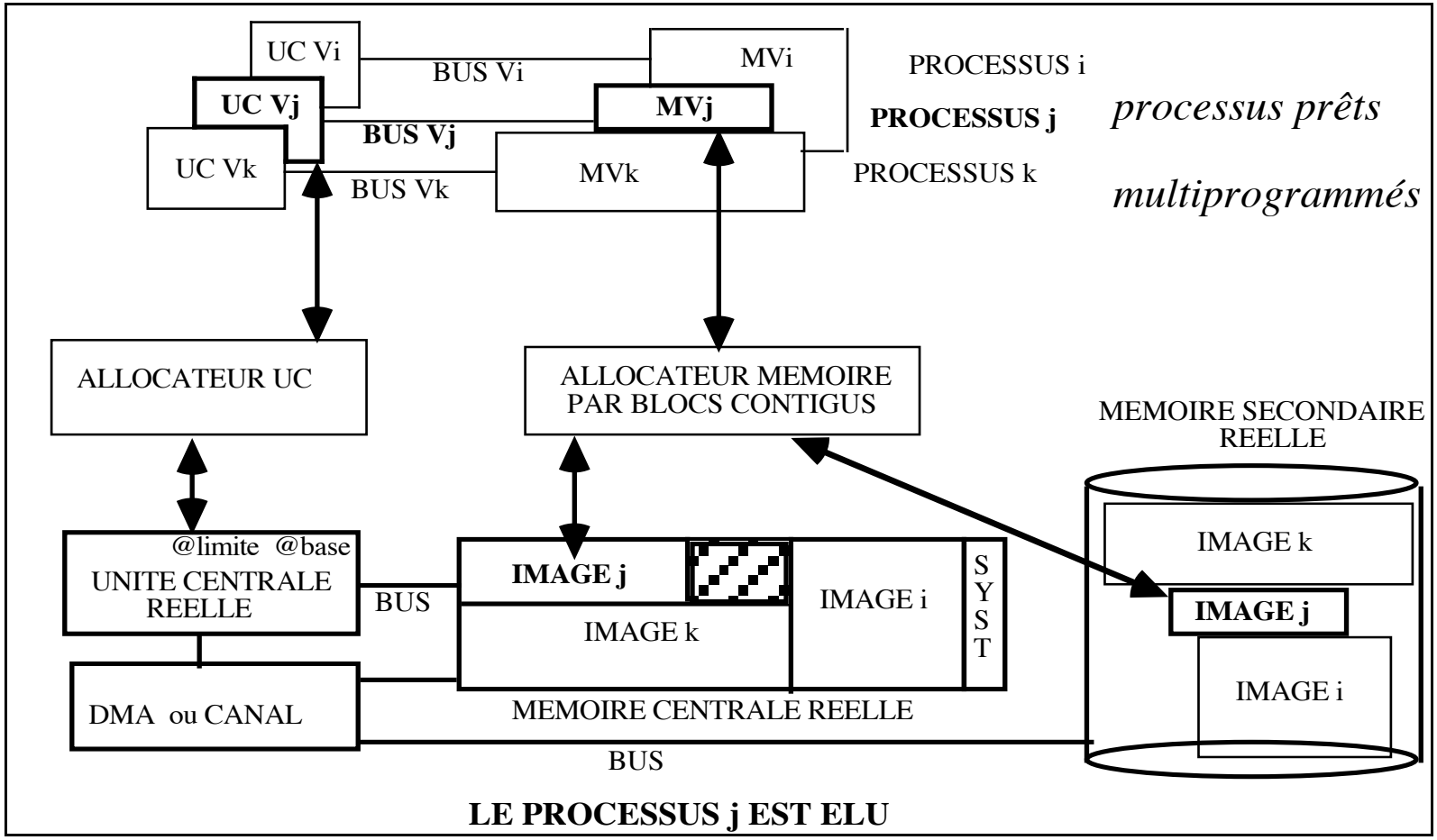
On utilise une table de localisation en mémoire secondaire pour repérer le placement de l'image. Le placement peut être statique (emplacements fixes) ou dynamique.

La mémoire secondaire est en général découpée en granules de même taille que les pages et le placement peut y être fait par granules contigus ou non. (la contiguïté permet de réduire la durée des accès disques).

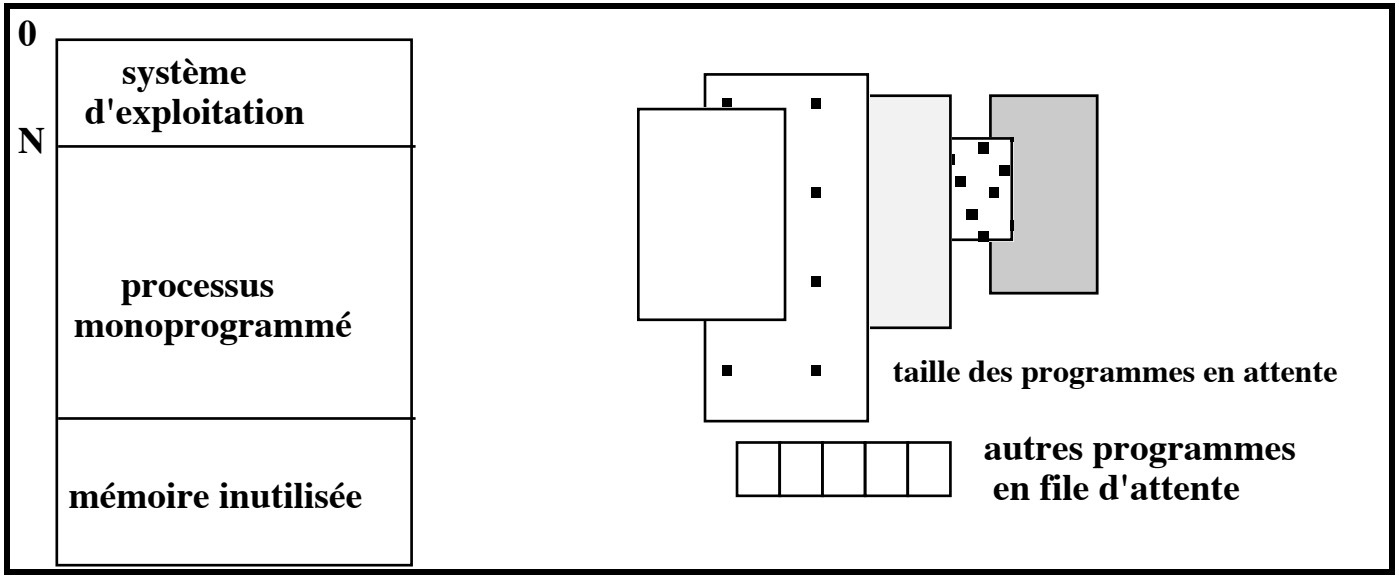
S'il n'y a qu'un niveau de mémoire secondaire (et donc pas de va et vient avec un autre niveau), l'allocation dynamique de granules de mémoire secondaire contient un risque d'interblocage quand on approche de la saturation.

PLACEMENT PAR BLOCS CONTIGUS

• processus = image (code + données + pile) en exécution sur processeur virtuel + mémoire virtuelle



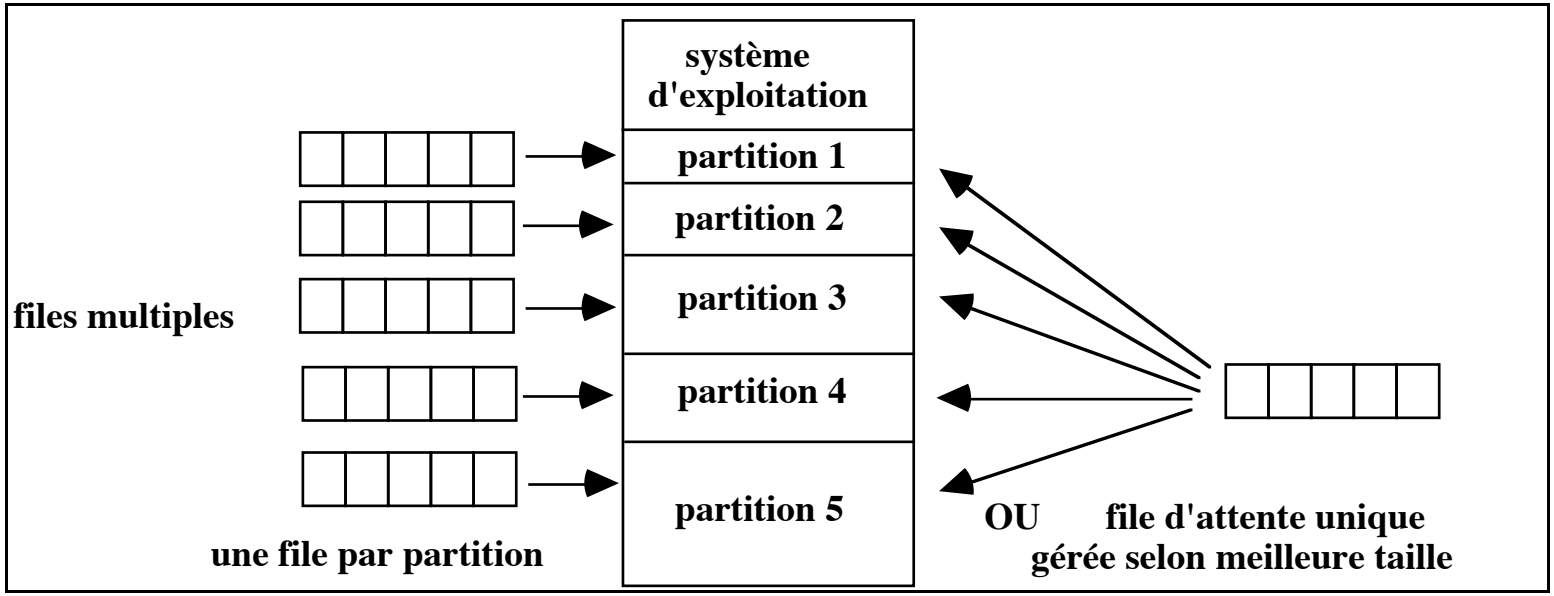
MONOPROGRAMMATION EN MÉMOIRE CENTRALE : le système d'exploitation et un seul processus géré par le système



problème de protection du système d'exploitation (appelé souvent moniteur).
Le moniteur gére l'enchaînement séquentiel des programmes en attente (politiques variées)
Taille maximale du programme limitée par la taille de la mémoire centrale (toute l'image doit être chargée).
Inactivité pendant les entrées-sorties

Un programme trop gros peut être découpé pour faire du recouvrement de certaines sections qui ne font pas de références mutuelles ("overlay").

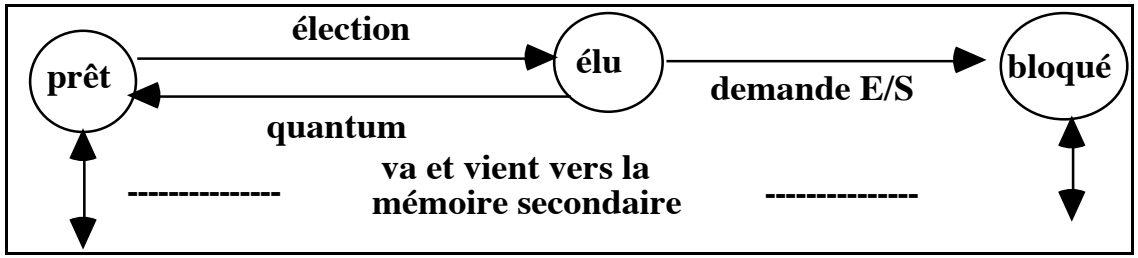
MULTIPROGRAMMATION DE PLUSIEURS PROCESSUS PRÊTS
GESTION AVEC DES PARTITIONS FIXES ET DE TAILLES DIFFÉRENTES (MFT)



PROBLÈMES

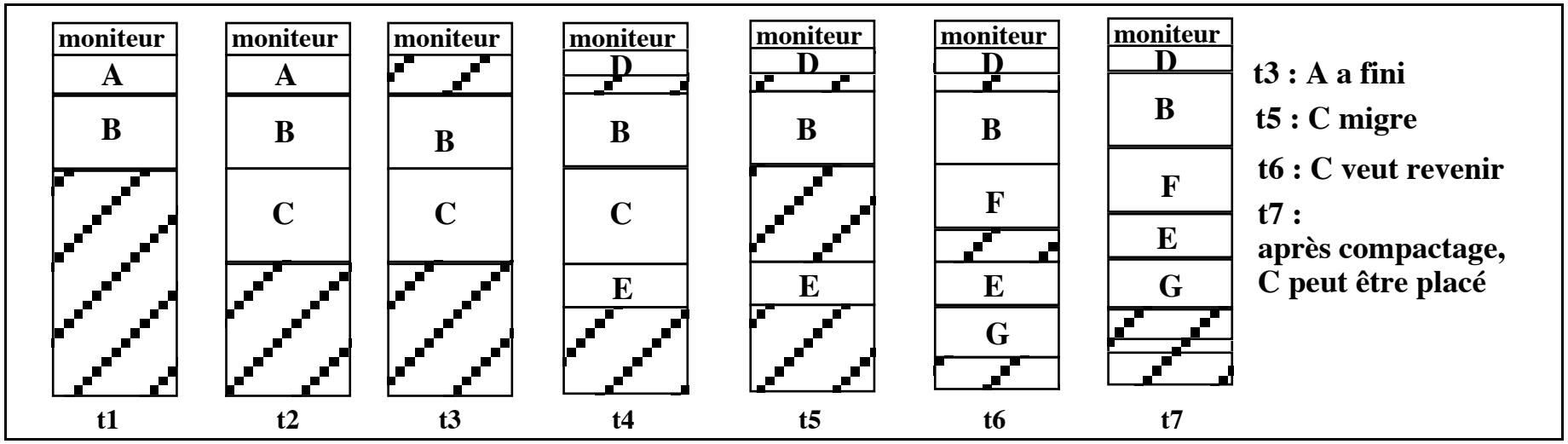
- **translation d'adresse : registre de base**
- **protection entre partitions et du moniteur : bits de protection ou registre de base et de limite**
- **toute l'image du processus doit être chargée dans une partition**
- **augmentation de la demande d'un processus déjà placé : va et vient et changement de partition**
- **va et vient pour les processus passés de prêt à bloqué ou en fin de quantum (cas des premiers Unix)**

ÉTATS DES PROCESSUS AVEC VA ET VIENT AVEC MÉMOIRE SECONDAIRE



Un processus ramené en mémoire secondaire n'est plus candidat à l'unité centrale

MULTIPROGRAMMATION A PARTITIONS VARIABLES (MVT)



si fragmentation, compactage lourd sauf si instruction spéciale

augmentation de l'allocation à un processus déjà placé (toute l'image doit être chargée dans une seule zone):
- **le déplacer vers une zone plus grande**
- **déplacer un autre processus vers mémoire secondaire et récupérer sa zone**
- **attribuer au départ un peu plus de mémoire en prévision des extensions**

ALLOCATION DE ZONE DE TAILLE T : trouver une zone libre de taille Z telle que Z ≥ T

Premier Accord ("First Fit") : la première zone qui convient
Meilleur Accord ("Best Fit") : la zone qui laisse le plus petit résidu Z - T
Résidu Maximal ("Worst Fit") :la zone qui laisse le plus grand résidu Z -T

Simulations (Shore 1965) : Premier accord et Meilleur Accord se valent. Résidu Maximal est mauvais.

Autres méthodes d'allocation : Buddy system, Fibonacci

EXEMPLES COMPARATIF D'ALLOCATION PAR ZONE

Utilisable pour toute allocation de mémoire par zone:
image processus, tampons E/S ou réseau, données dynamiques (tas, "heap"), serveurs de données, images, sons granules consécutifs de mémoire secondaire

premier exemple	Premier Accord	Meilleur Accord	Résidu Maximal
état initial	2000, 1500	1500, 2000	2000, 1500
demande de 1200	800, 1500	300, 2000	800, 1500
demande de 1600	ECHEC!	300, 400	ECHEC!
perdu en fragmentation	2300	700	2300

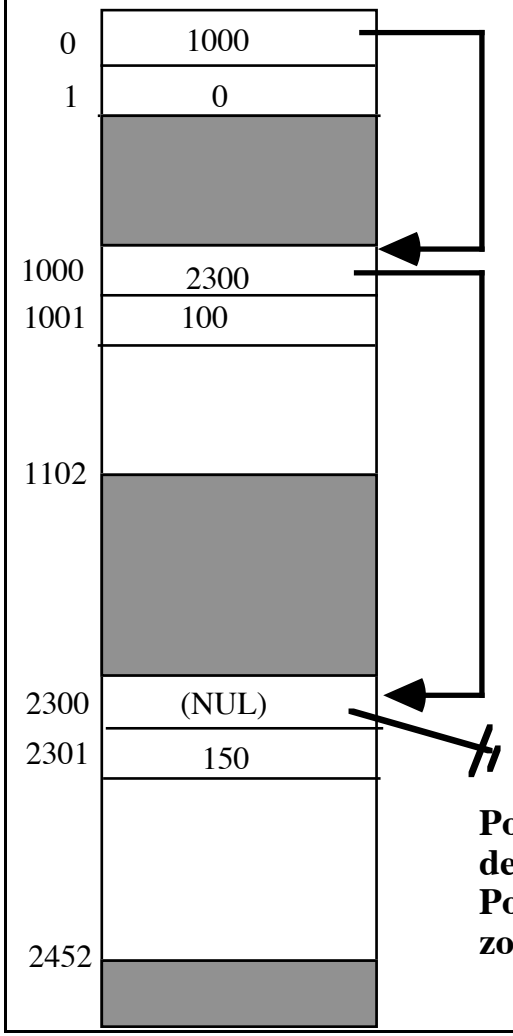
Ici c'est Meilleur Accord qui est le meilleur

deuxième exemple	Premier Accord	Meilleur Accord	Résidu Maximal
état initial	2000, 1500	1500, 2000	2000, 1500
demande de 1200	800, 1500	300, 2000	800, 1500
demande de 1400	800, 100	300, 600	800, 100
demande 700	100, 100	ECHEC!	100, 100
perdu en fragmentation	200	900	200

ici, c'est Premier Accord qui est le meilleur

FRAGMENTATION EXTERNE :
toute taille Z de zone libre est inférieure à la taille T demandée
mais
le total de toutes les tailles Z de zone libre est supérieur à la taille T demandée

REPRÉSENTATION DES ZONES LIBRES



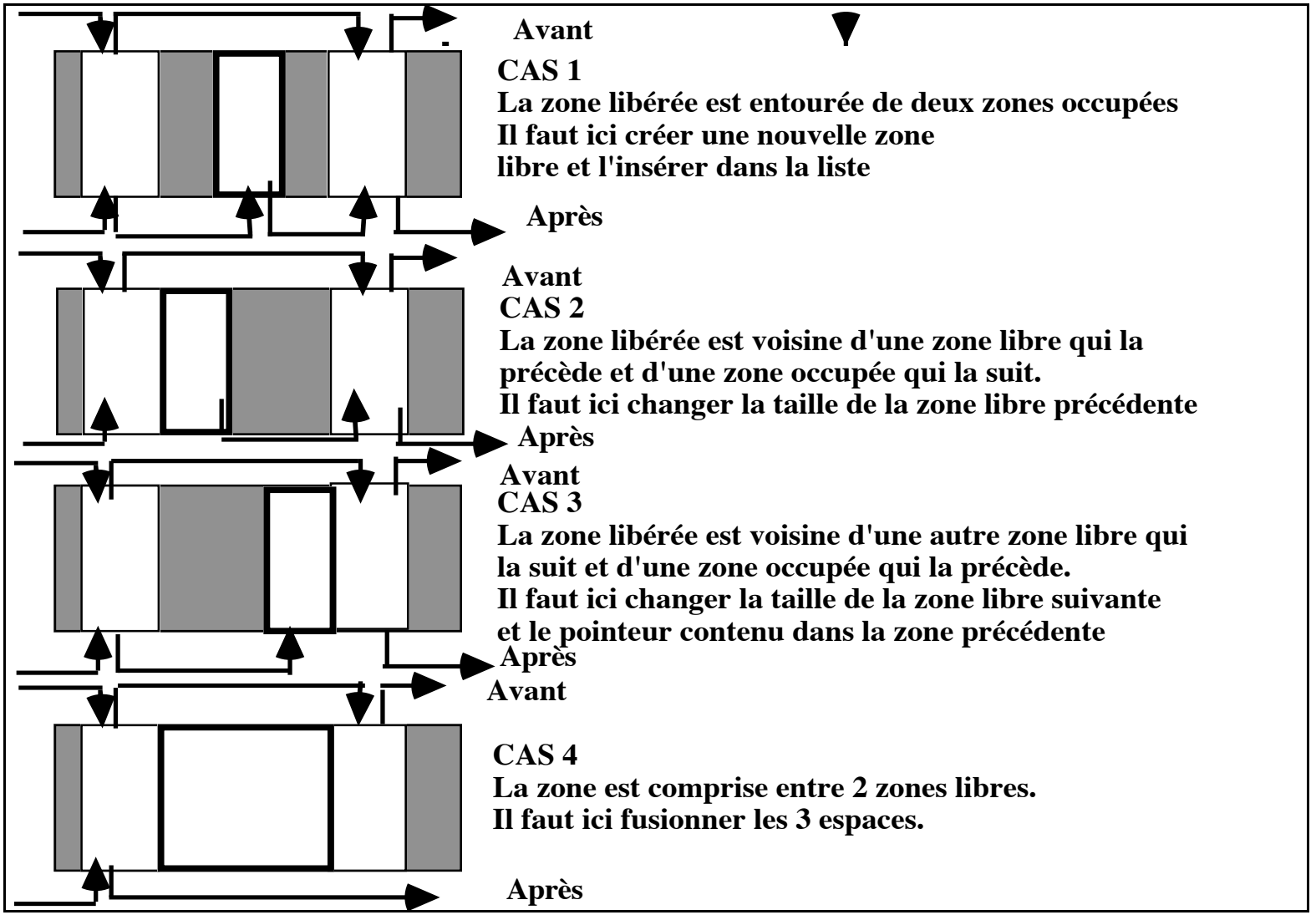
La première case indique l'adresse de la zone libre suivante

La deuxième indique la taille de la zone (sans compter les deux premiers mots)

La première zone libre est à l'adresse 0 et vaut 0 pour simplifier l'algo de recherche.

Pour Premier Accord, on chaîne selon les adresses croissantes des zones libres (comme sur la figure)
Pour Meilleur Accord, on chaîne selon les tailles croissantes de zone libre

OPÉRATIONS À LA LIBÉRATION D'UNE ZONE



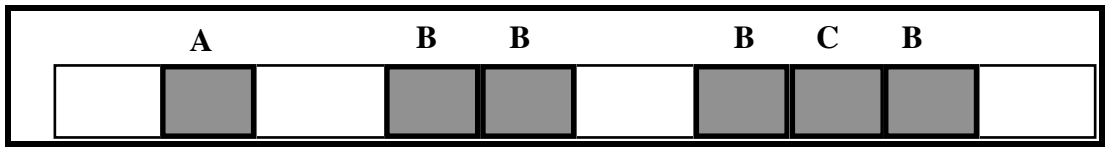
RÈGLE DES 50% (Knuth) : RELATION ENTRE L'ÉVOLUTION DE M ET CELLE DE N

On note : N = nombre de zones allouées M = nombre de zones libres

À L'ALLOCATION : supposons qu'on crée un résidu avec la probabilité p (p voisin de 1),

donc l'allocation des N zones entraîne (1 - p)*N diminutions de M

À LA LIBÉRATION : elle entraîne une variation de M déterminée par les trois types de zones allouées



A : zone entourée par 2 zones libres. A sa libération, M diminue de 1. Soit nA leur nombre.

B : zone contigüe à une zone libre. A sa libération, M est inchangé. Soit nB leur nombre.

C : pas de zone libre autour. A sa libération, M augmente de 1. Soit nC leur nombre.

Donc la libération des N zones entraîne que M augmente nC fois et diminue nA fois

On a évidemment : N = nA + nB + nC et M = (2*nA + nB)/2 + e,

avec e = 0, 1 ou 2 selon la configuration des bords, donc e << M , e négligeable

[Sur le dessin ci-dessus : nA = 1, nB = 4, nC = 1 donc N = 6 et M = 4 et e = 1]

Si le système est à l'équilibre, alors le nombre moyen des zones libres reste constant

l'augmentation de M = la diminution de M

d'où nC = nA + (1 - p)*N ou encore nC - nA = (1 - p)*N

Or on a vu que si M >> e, alors N - 2*M = nC - nA ou encore N - 2*M = (1 - p)*N

Donc 2M = N*p Le nombre des zones libres est à peu près égal à la moitié du nombre des zones allouées

AUTRES ALLOCATIONS PAR ZONES

• Objets passifs créés dynamiquement :

- par le programme,

- par un serveur d'objets, d'images, de sons (sur disques),

- par le système de fichiers (sur disque ou en mémoire),

- par un serveur de base de données, (sur disque ou en mémoire),

- par le gestionnaire de messages,...

• Demandes imprévisibles,

• Tailles souvent très variables

• Quand le nombre de tailles différentes est limité :

- gestion de files de blocs libres, un par taille possible

- gestion par subdivision spécifique permettant de regrouper les blocs libres en compagnons :

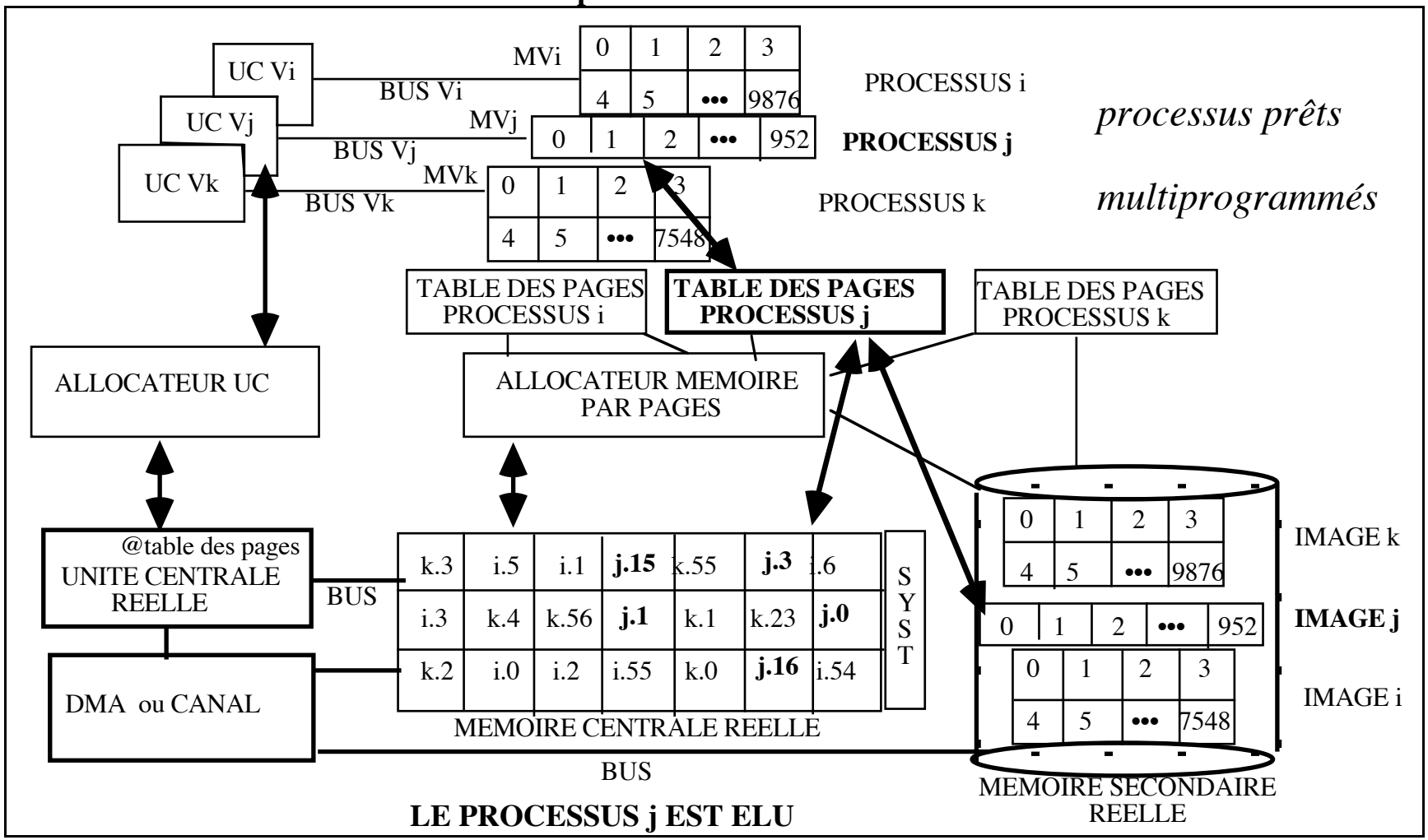
de tailles égales multiples de deux : "buddy system"

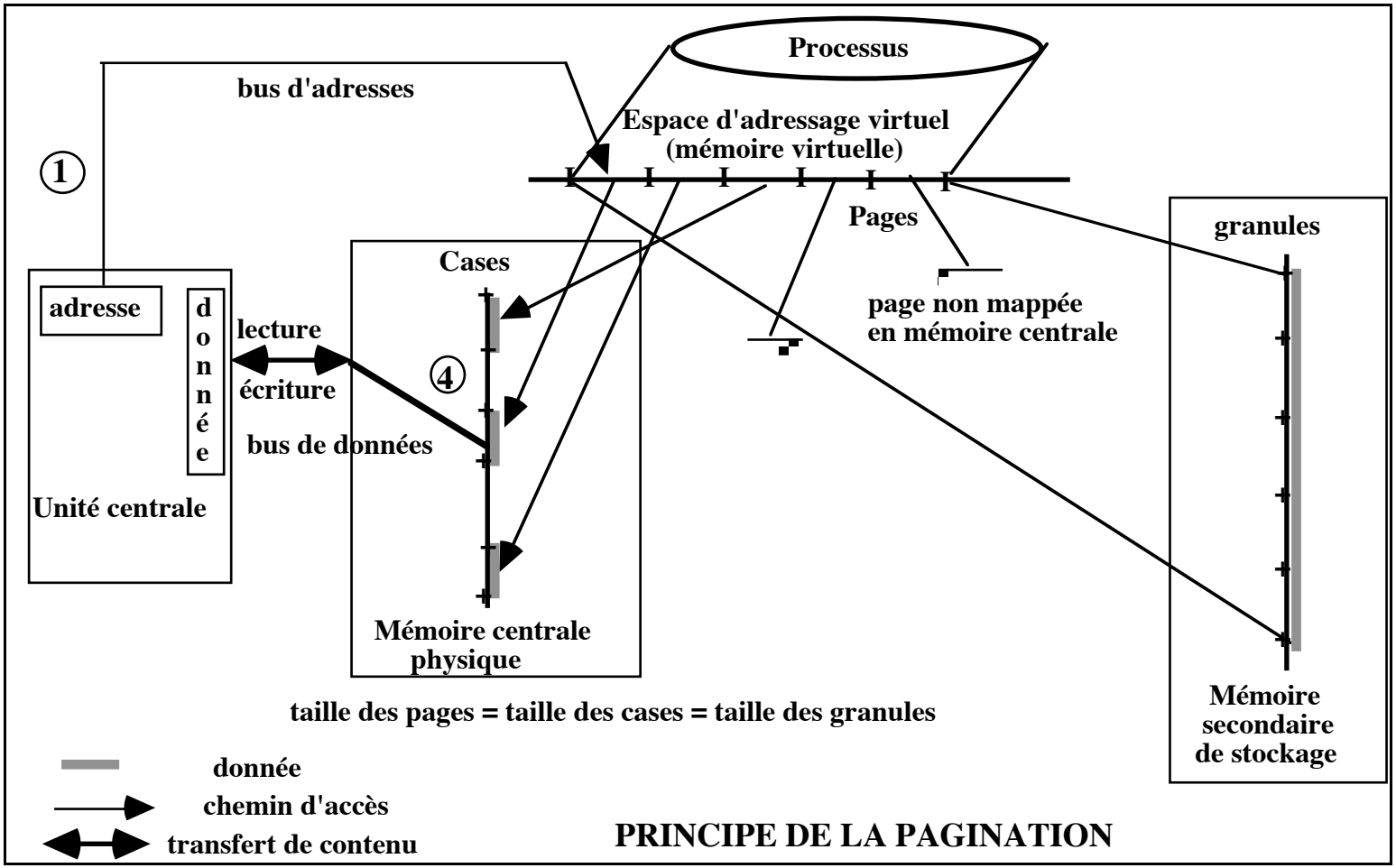
de tailles inégales : "Fibonnacci"

et simplifiant le regroupement à la libération, mais apportant un risque de fragmentation interne

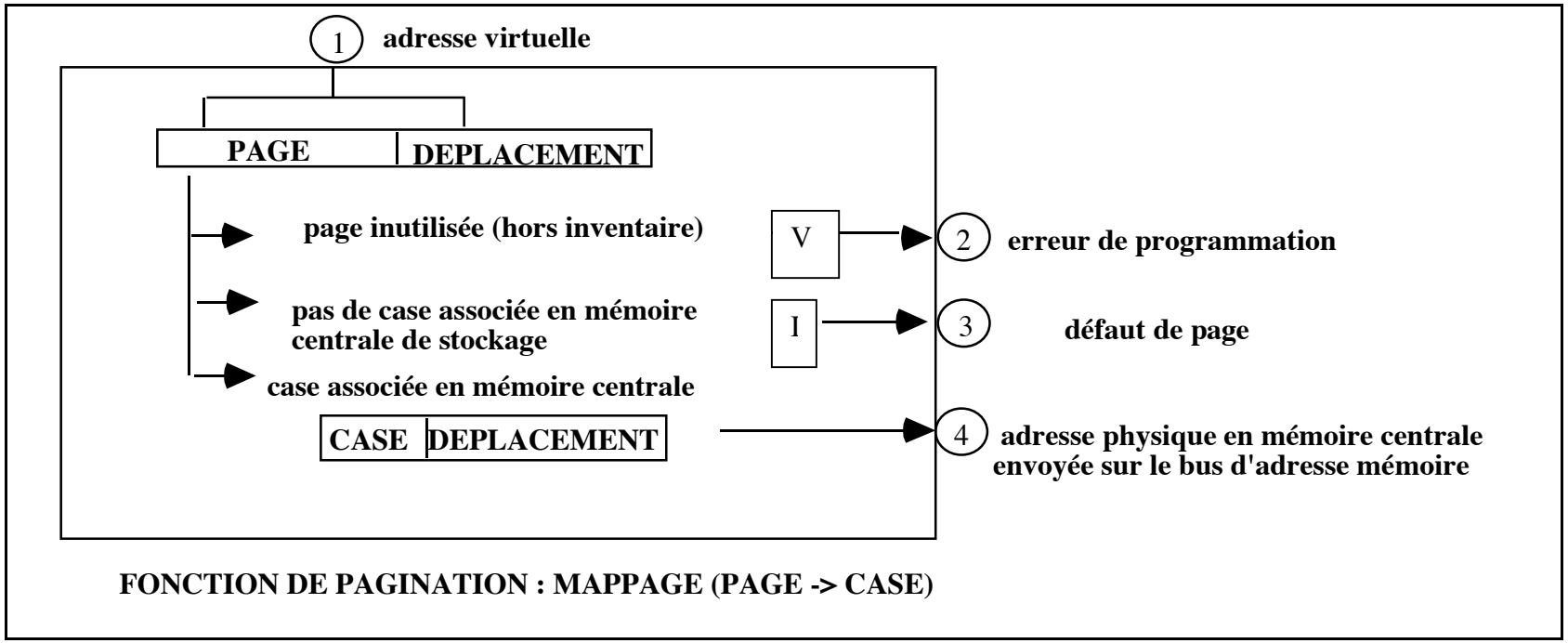
• Architectures à adressage segmenté et chargement de plusieurs segments, chacun dans une zone. Chaque segment est repéré par un registre de base et déplacement.

PLACEMENT PAR PAGINATION
processus = image (code + données + pile) en exécution sur processeur virtuel + mémoire virtuelle




Processus
bus d'adresses
Espace d'adressage virtuel
(mémoire virtuelle)
1
Pages
granules
Cases
adresse
donnée
lecture
écriture
4
bus de données
page non mappée
en mémoire centrale
Unité centrale
Mémoire centrale
physique
taille des pages = taille des cases = taille des granules
Mémoire
secondaire
de stockage
donnée
chemin d'accès
transfert de contenu
PRINCIPE DE LA PAGINATION

MAPPAGE DE L'ADRESSE LOGIQUE VIRTUELLE VERS L'ADRESSE PHYSIQUE



EXEMPLES DE MAPPAGES

ESPACE D'ADRESSAGE

Adresse virtuelle sur 24 bits : 12 bits PAGE + 12 bits DEPLACEMENT
Espace d'adressage de 16 Mo : 4 K pages possibles + 4K déplacements possibles dans chaque page

soit une adresse virtuelle binaire : 000000110010010010000001
découpage :

000000110010	010010000001

PAGE : 32 + 16 + 2 = 50, DEPLACEMENT : 1024 + 128 + 1 = 1153
(50)* 4096 + 1153 = 204800 + 1153 = 205953 en décimal

MÉMOIRE CENTRALE

a) si 8 bits pour CASE : taille mémoire centrale (1 Mo) < taille espace d'adressage (16 Mo)

soit la CASE = 11000000 ou (256 + 128) = 384 en décimal d'où CASE 384, DEPLACEMENT 1153

11000000	010010000001

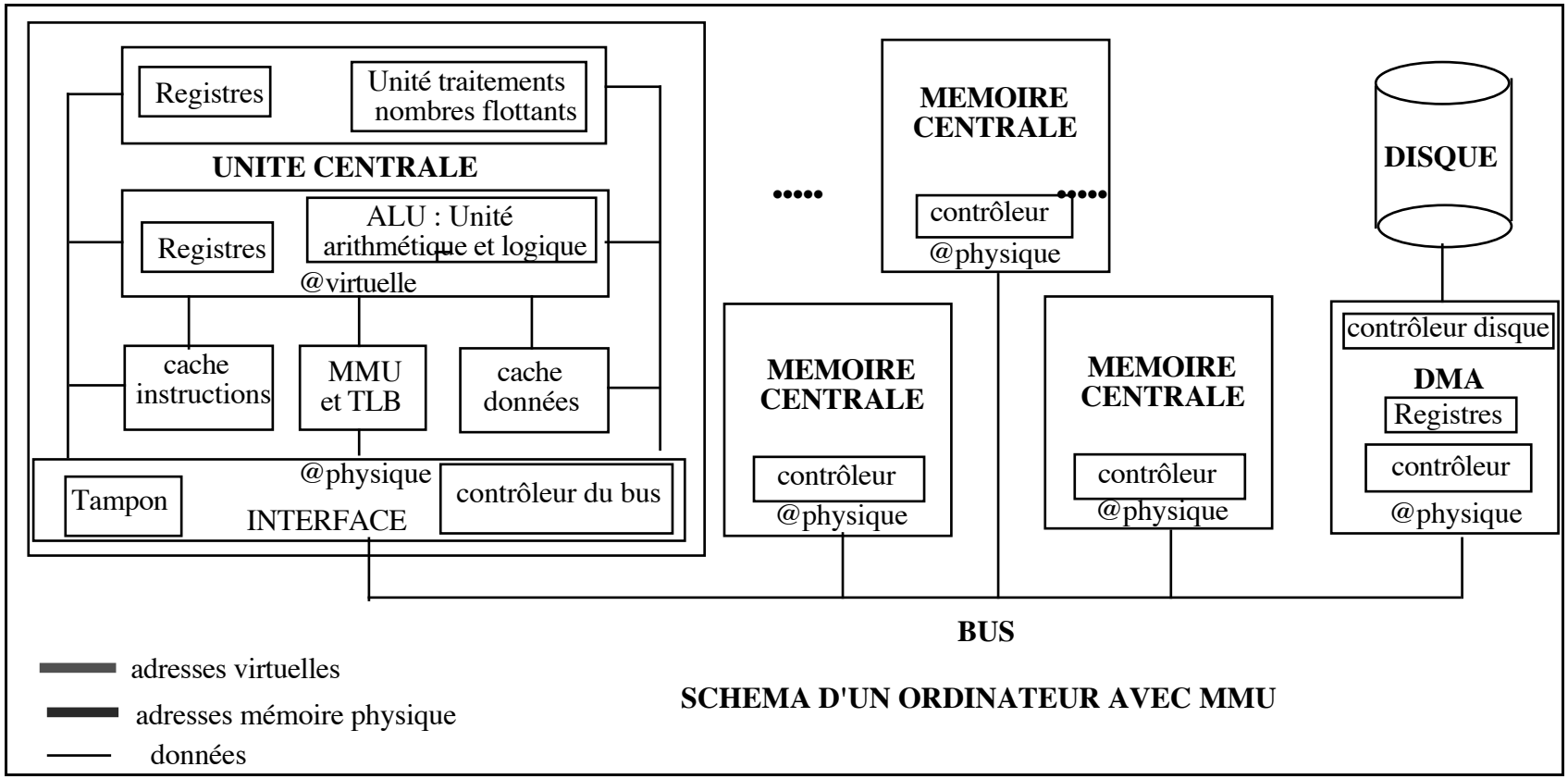
d'où adresse physique en mémoire centrale : (384*4096) + 1153 = 1574017 en décimal

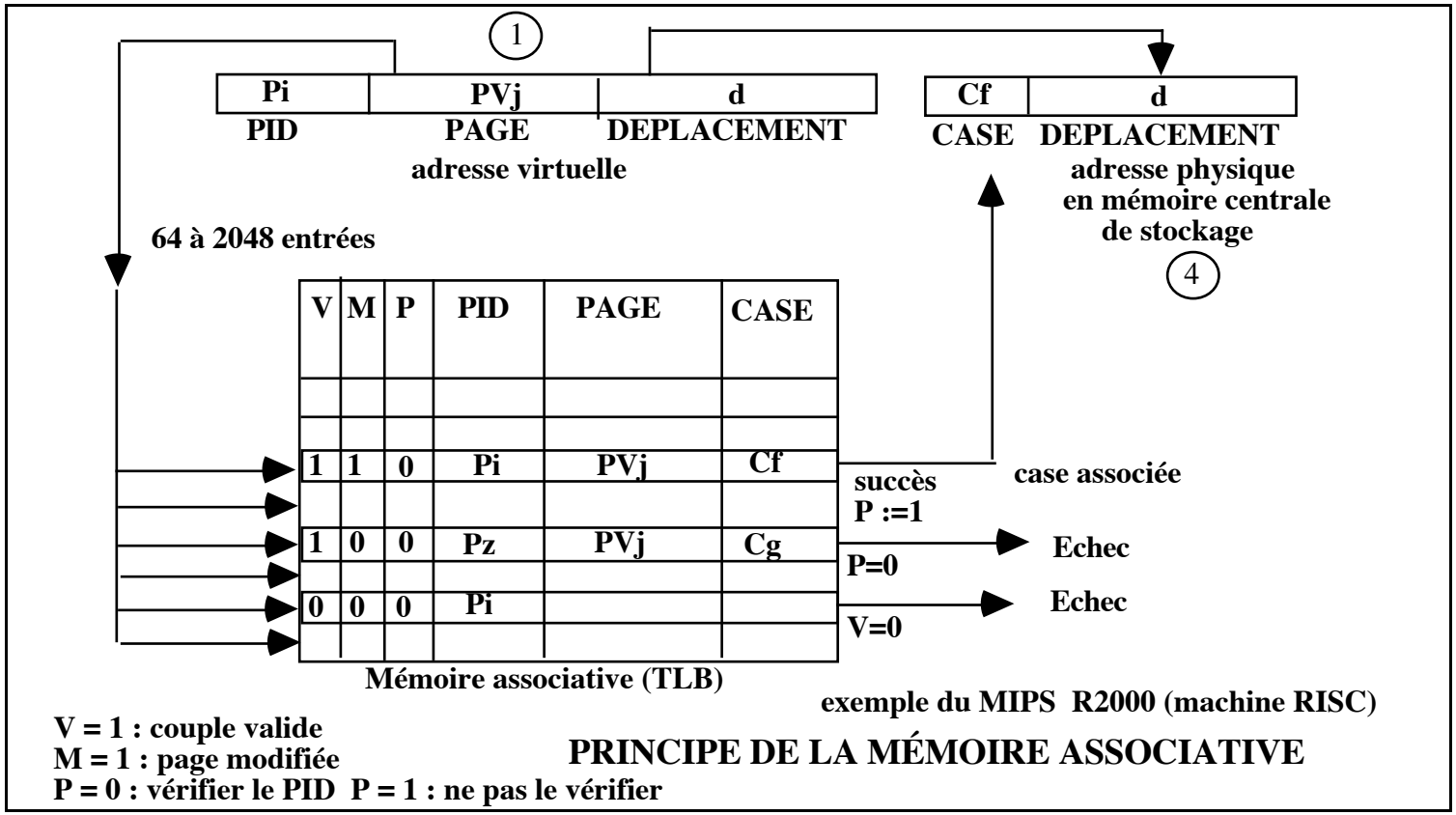
b) si 14 bits pour CASE : taille mémoire centrale (64 Mo) > taille espace d'adressage (16 Mo)

soit la CASE = 00000001000010 ou (128 + 2) = 130 en décimal : CASE 130 DEPLACEMENT 1153

00000001000010	010010000001

d'où adresse physique en mémoire centrale : (130*4096) + 1153 = 533633 en décimal


Registres
Unité traitements nombres flottants
UNITE CENTRALE
Registres
ALU : Unité arithmétique et logique
@virtuelle
cache instructions
MMU et TLB
cache données
@physique
Tampon
INTERFACE
contrôleur du bus
MEMOIRE CENTRALE
contrôleur
@physique
MEMOIRE CENTRALE
contrôleur
@physique
MEMOIRE CENTRALE
contrôleur
@physique
DISQUE
contrôleur disque
DMA
Registres
contrôleur
@physique
BUS
adresses virtuelles
adresses mémoire physique
données
SCHEMA D'UN ORDINATEUR AVEC MMU



Avec la clé PAGE, recherche dans la mémoire associative TLB ("translation look-aside buffer")

• si SUCCÈS, on vérifie le PID (éviter une page avec même PVj pour un autre processus)
à ne faire qu'au premier accès à cette page, d'où mise à 1 du bit P

• si ÉCHEC, appel superviseur pour recherche dans la table des pages en mémoire centrale

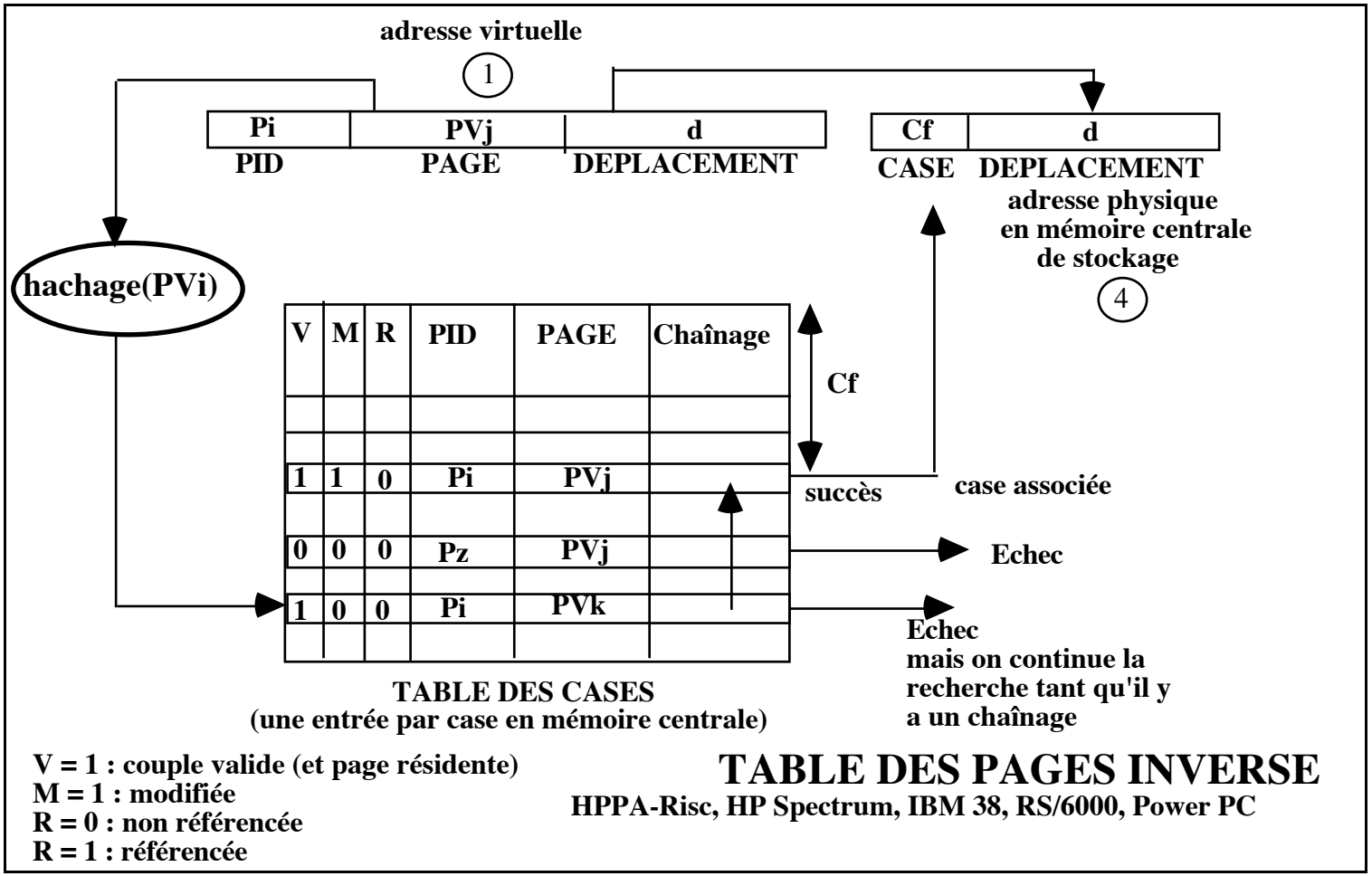
•• si la page PVj est résidente, trouver une place (remplacement dans la mémoire associative)

•• sinon d'abord trouver une case de mémoire centrale puis une place dans la mémoire associative

A l'élection d'un nouveau processus, on remet à zéro tous les bits P

GRANDES MÉMOIRES VIRTUELLES (64 BITS)

Table des pages de taille énorme de l'ordre de 2^{20} entrées ($2^{20} = 10^6$)
Entre la mémoire associative et la table des pages classiques, qui peut être stockée sur disque, (pagination de la table des pages), on met une table des pages inverse



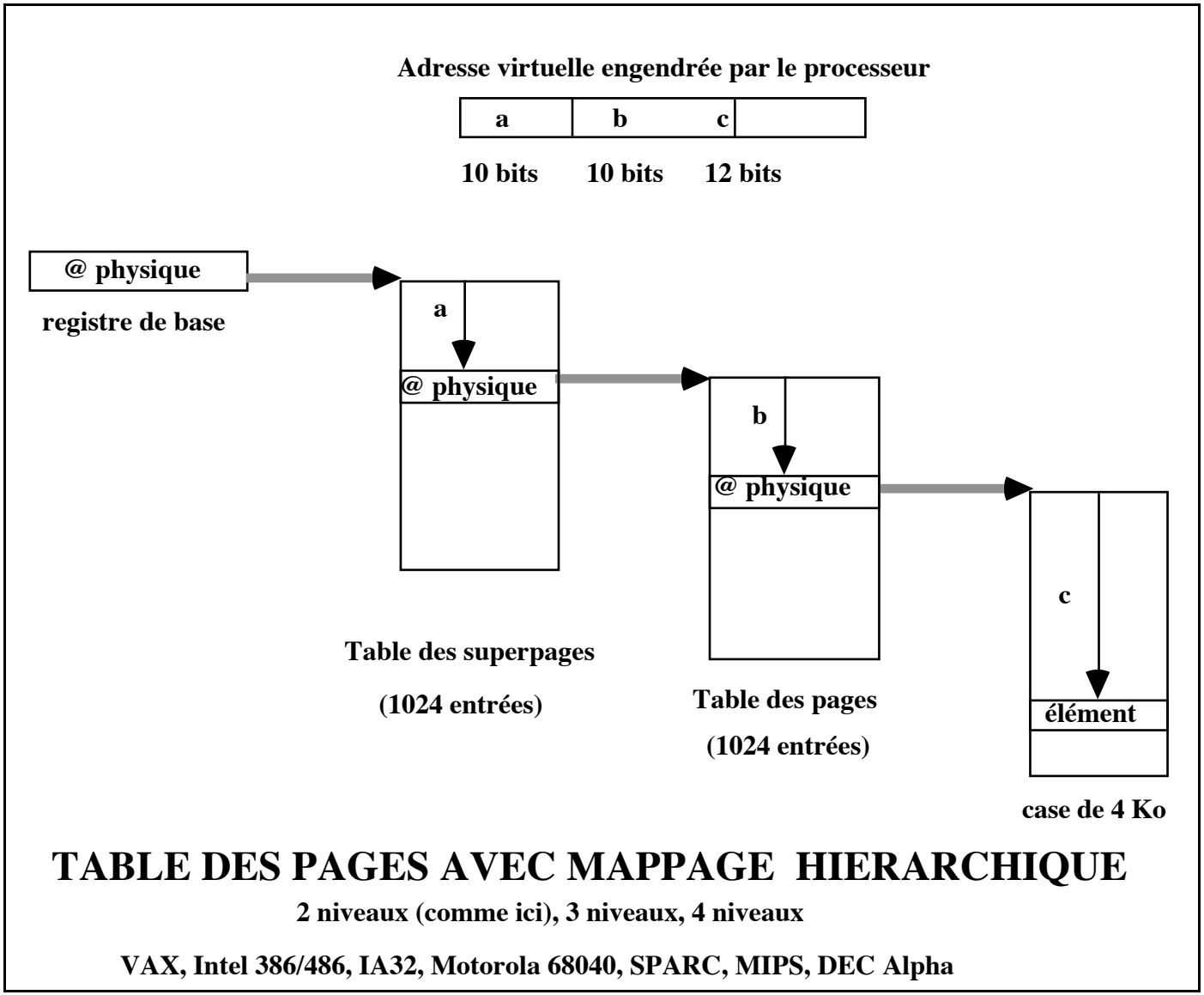


TABLE DES PAGES AVEC MAPPAGE HIERARCHIQUE

2 niveaux (comme ici), 3 niveaux, 4 niveaux

VAX, Intel 386/486, IA32, Motorola 68040, SPARC, MIPS, DEC Alpha

PAGINATION

FRAGMENTATION : Pas de fragmentation interne ; peu de fragmentation interne

PARTAGE DE PAGE ET DE CASE PAR PLUSIEURS PROCESSUS (code réentrant, données commune)
table des pages partagées si espace virtuel partagé par convention explicite

TAILLE DES PAGES
petite page : peu de fragmentation mais beaucoup d'entrées-sorties et grande taille de la table des pages
grande page : fragmentation interne, meilleur pour les entrées-sorties et la taille des pages
Il existe des petits objets (données dans les programmes, descripteurs de ressources et de processus
et de grands objets (fichiers, images bitmap)
compromis autour de 2K ou 4K mais aussi pages de taille variable (4Ko ou 4Mo)

VERROUILLAGE DE PAGES PENDANT LES ENTRÉES-SORTIES
verrou de case qui ne peut être sélectionné pour remplacement
ou pas d'entrées-sorties dans l'espace virtuel utilisateur

INFLUENCE DE LA STRUCTURE DES PROGRAMMES : améliorer la localité

AUTRES UTILISATIONS DYNAMIQUES DE LA MÉMOIRE

Stockages intermédiaires ou persistants de données dans des serveurs en mémoire centrale ou sur disque pour :
- des bases de données,
- des tampons de messages dans les réseaux,
- des cache web,
- des serveurs de flux multimédia (image video, son,..).

On retrouve dans ces serveurs les techniques d'allocation par zone ou par pagination, ainsi que les politiques de remplacement qui seront vues au chapitre 5.

• Architectures à adressage segmenté. Chaque segment peut être chargé dans une zone contiguër ou être paginé.