Gestion Mémoire

Plan

- Rôle de la mémoire
- Espace d'adressage
- Allocation régions mémoire de tailles variables
- Gestion de la mémoire centrale par zones
- Gestion de la mémoire centrale par pagination

Rôle de la mémoire

- Stocker instructions et données des programmes exécutés par le processeur
- Repérer les objets par leur adresse dans la mémoire
- Espace d'adressage
  - ensemble des unités mémoire (octets) individuellement addressables depuis le CPU
  - Mécanisme(s) de construction de l'adresse de chaque unité de l'espace

Unités mémoire

- Unités adressables
  - octet de 8 bits (`char`)
  - mot de 16 bits (`short`)
  - mot de 32 bits (`int`)
  - mot de 64 bits (`long long`)
- Alignement multiple de la taille
  - Non imposé (Intel, PowerPC) mais plus efficace
  - Imposé sinon exception (Sparc)

Représentation alignée

```
struct test_t {
    int   a;
    char  b;
    short c;
    char  d;
    int   e;
};

struct test_t test = {11, 22, 33, 44, 55};
```

```
.globl test
        .data
        .align 4
        .type    test, @object
        .size    test, 16
test:
        .long    11
        .byte    22
        .zero    1
        .value   33
        .byte    44
        .zero    3
        .long    55
```

Avec attribut "__packed__"

```
struct test_t {
    int   a;
    char  b;
    short c;
    char  d;
    int   e;
} __attribute__((__packed__));

struct test_t test = {11, 22, 33, 44, 55};
```

```
.globl test
    .data
    .type   test, @object
    .size   test, 12
test:
    .long   11
    .byte   22
    .value  33
    .byte   44
    .long   55
```

Plan

- Rôle de la mémoire
- Espace d'adressage
- Allocation régions mémoire de tailles variables
- Gestion de la mémoire centrale par zones
- Gestion de la mémoire centrale par pagination

Espace d'adressage

- Espace d'adressage linéaire
  - Adresse = valeur entre 0 et 2^n - 1
  - 4 giga-octets avec adresse sur 32 bits
- Espace d'adressage segmenté
  - ensemble d'espaces linéaires disjoints
  - adresse = (numéro segment, position dans segment)
  - (dernière position S) + 1 ≠ première position(S + 1)
  - segment instructions, données globales, pile

Programme et espace adressage

- Objets d'un programme
  - fonctions, variables, constantes désignés par un nom symbolique unique
  - taille des objets calculée par compilateur
  - éditeur de liens attribue position absolue (adresse) des objets dans espace d'adressage
- Taille et adresses des instructions et des données stockés dans programme binaire

Espace d'adressage physique

- Inclut mémoire centrale + espace d'E/S
- Adresses objets = adresses physiques
- Simple
  - Premiers ordinateurs
  - Petits systèmes enfouis
- 2 inconvénients
  - Adresses des programmes inter-dépendantes
  - Espace d'adressage limité à la taille de la mémoire physique disponible

Espace d'Adressage Virtuel

- Adresses virtuelle objets ≠ adresses physiques
  - Indépendantes des conditions d'exécution
    - De la taille de la mémoire centrale
    - De la présence d'autres programmes en mémoire
- Correspondance entre adresse virtuelle et adresse en mémoire physique
  - établie dynamiquement par fonction de topographie
  - S'appuie sur mécanisme matériel de translation adresse virtuelle -> adresse physique

Gestion Espace d'Adressage

- Espace d'adressage virtuel découpé en régions
  - Adresses contigües, tailles variables
- Régions allouées statiquement au chargement du programme : code, données globales, piles
- Régions gérées dynamiquement pendant exécution du programme
  - Objets créés dans le tas (malloc/free, new/delete)
  - Objets mappés par programme
    - Fichiers
    - Mémoire partagée

Plan

- Rôle de la mémoire
- Espace d'adressage
- Allocation régions mémoire de tailles variables
- Gestion de la mémoire centrale par zones
- Gestion de la mémoire centrale par pagination

Allocation Régions (1)

- Allocation de régions de tailles variables
  - Espace libre = ensemble de blocs de ≠ tailles
- Taille bloc libre > taille demandée
  - Reste bloc libre de plus petite taille
- Fragmentation externe
  - Somme taille des zones libres > taille demandée
  - Taille de chaque zone libre < taille demandée

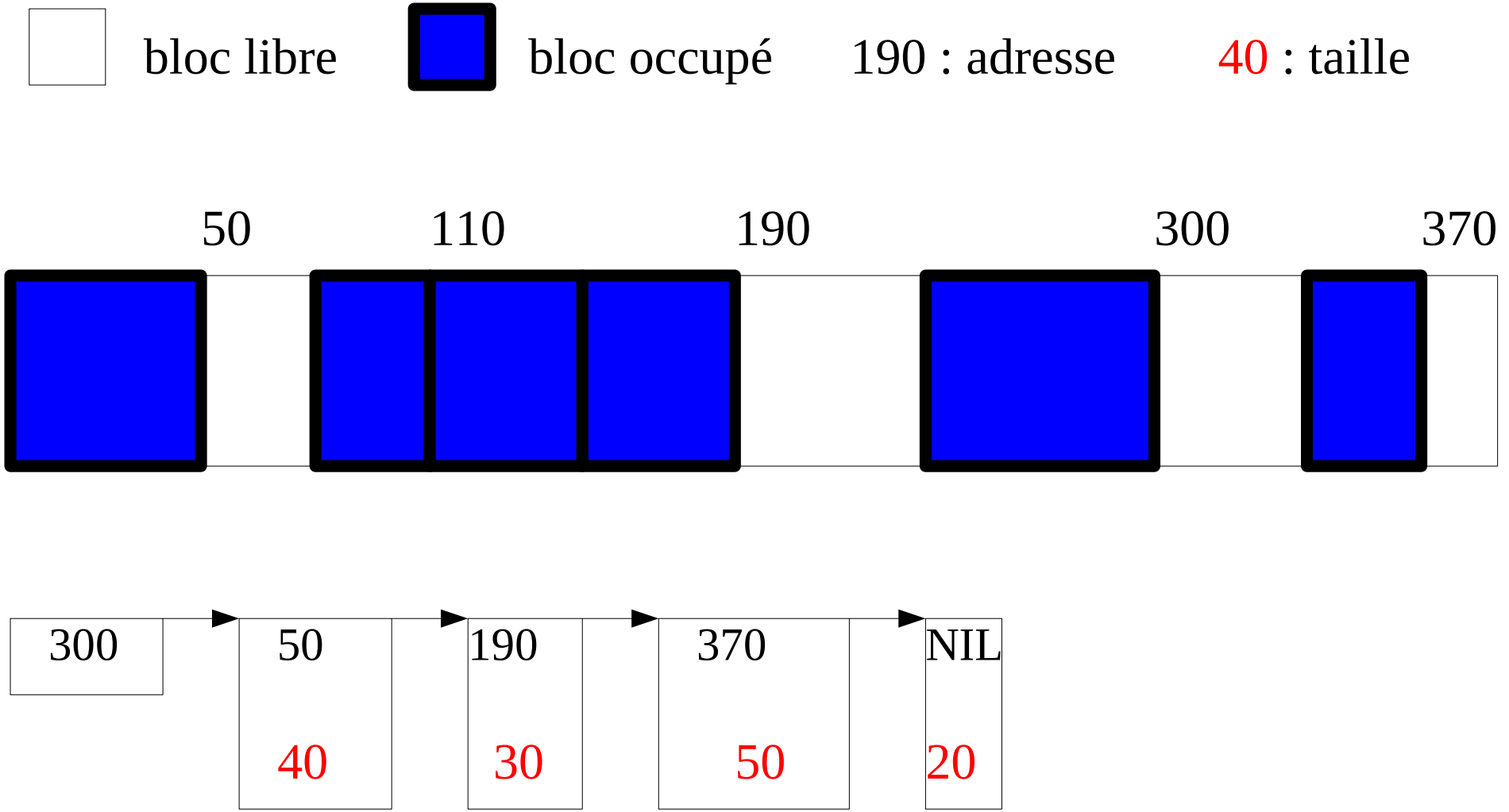
Allocation Régions (2)

- Réunir 2 blocs libres adjacents
  - Minimiser fragmentation
- Immédiatement, lors de la libération d'un bloc
  - retrouver éventuel bloc libre adjacent du bloc libéré
    - Liste des blocs libres par adresses croissantes
  - Pénalise opération de libération
- Plus tard
  - À partir d'un seuil minimum de blocs libres
  - Lorsque allocation non satisfaite immédiatement

Allocateur "First-Fit" (1)

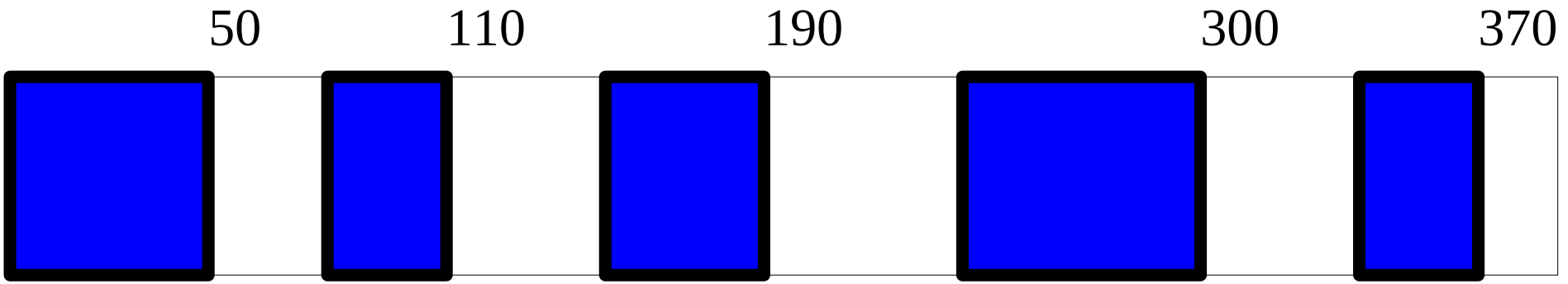
- Alloue depuis le premier bloc libre qui convient
- Liste de blocs libres gérée en FIFO ou par adresses croissantes
- Chaque bloc libre contient
  - Sa taille
  - Adresse bloc suivant dans la liste
- Allocation doit au pire parcourir toute la liste

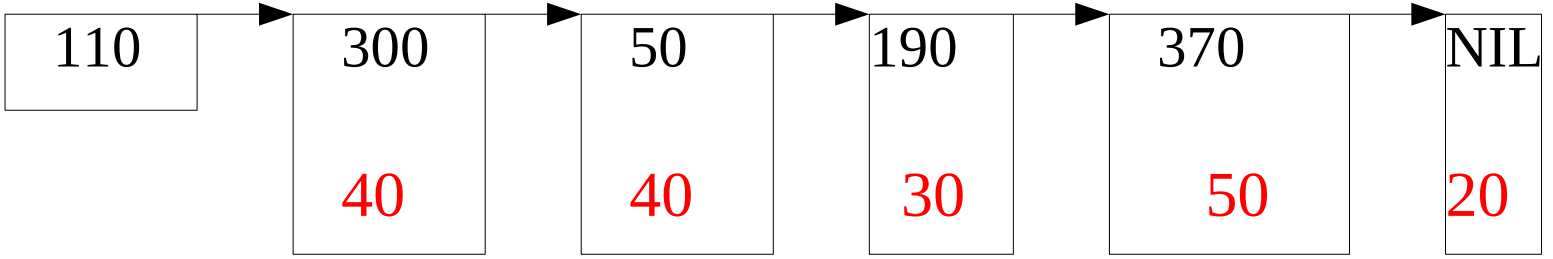
Allocateur First-Fit (2)



Allocateur First-Fit (3)

Libération bloc de taille 40 à l'adresse 110

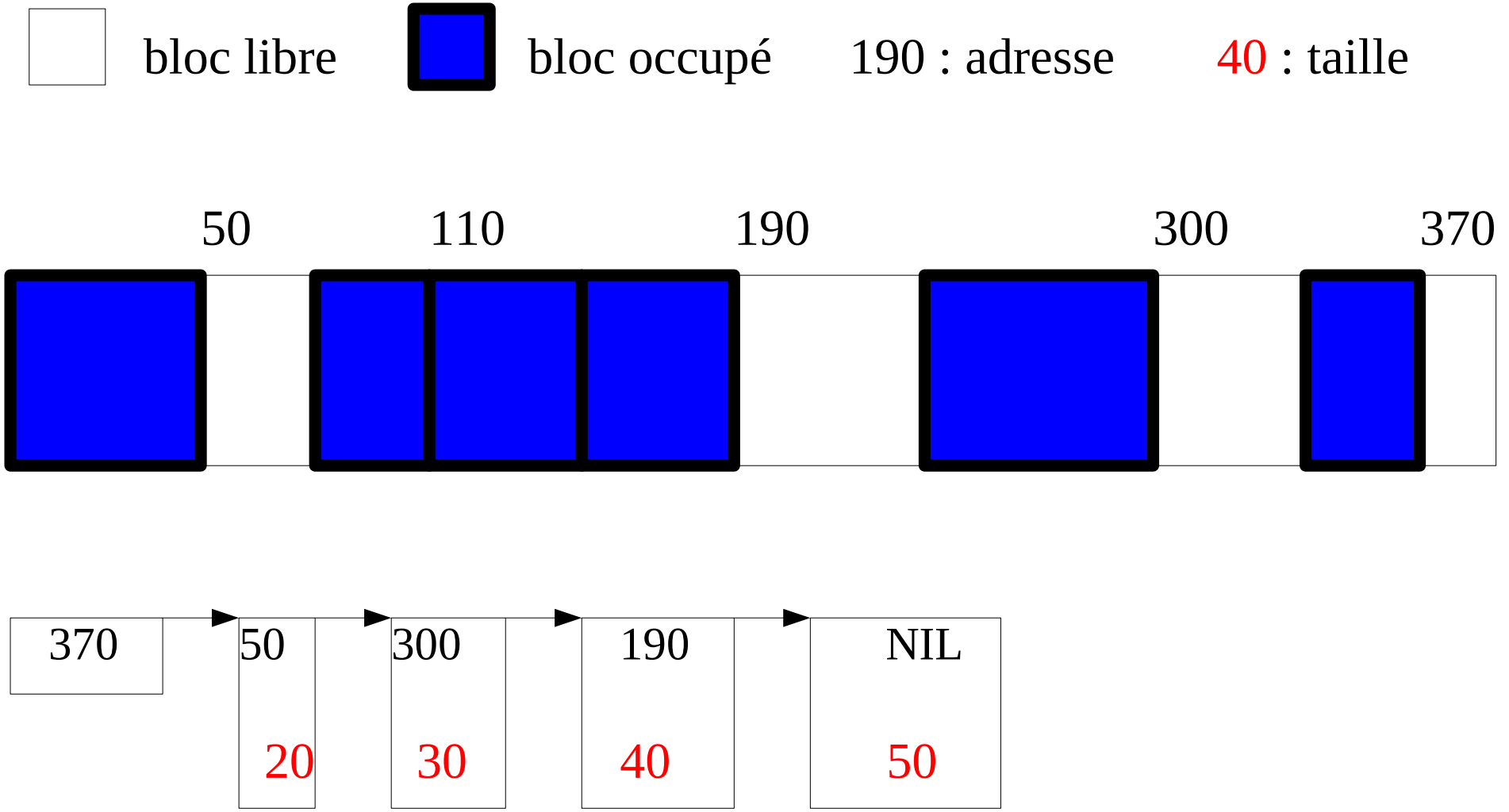




Allocateur "Best-Fit" (1)

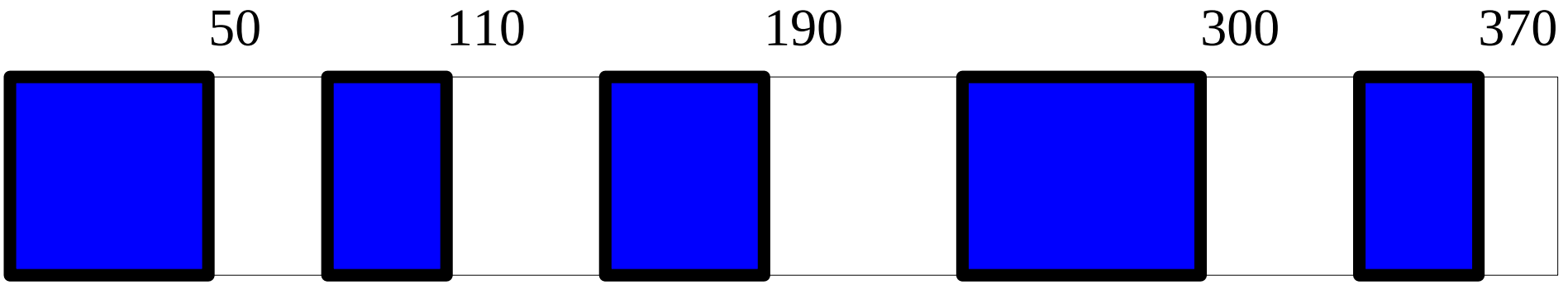
- Alloue depuis le bloc libre qui laisse le plus petit résidu
- Liste de blocs libres gérée par tailles croissantes
- Chaque bloc libre contient
  - Sa taille
  - Adresse bloc suivant dans la liste
- Libération rapide
- Allocation doit au pire parcourir toute la liste

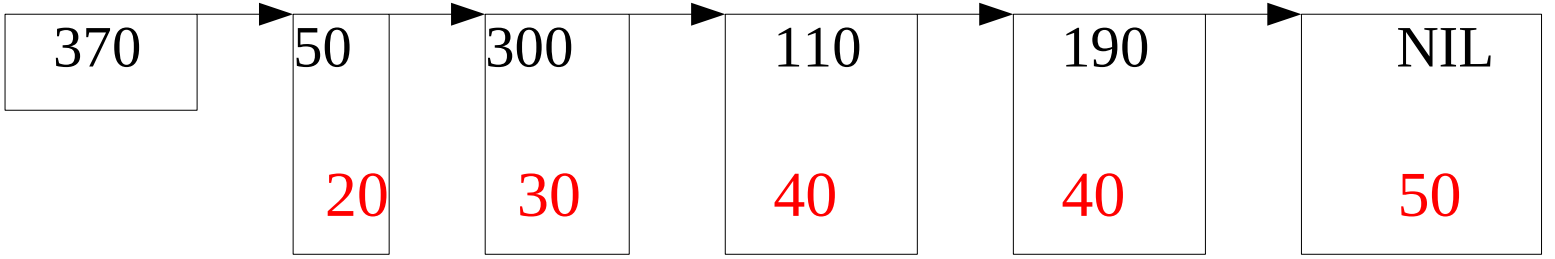
Allocateur Best-Fit (2)



Allocateur Best-Fit (3)

Libération bloc de taille 40 à l'adresse 110





Plan

- Rôle de la mémoire
- Espace d'adressage
- Allocation régions mémoire de tailles variables
- Gestion de la mémoire centrale par zones
- Gestion de la mémoire centrale par pagination

Gestion espace virtuel par zones (1)

- Traduction addresses basé sur registres de translation d'adresse
  - Adresse physique début de zone
  - Taille multiple d'une taille minimum (puissance de 2)
  - Droits d'accès (lecture/écriture/exécution)
- Jeu de plusieurs registres
  - code, données globales, pile, tas
- Allocation de la mémoire centrale en zones de taille variables, multiples d'une taille minimum

Gestion espace virtuel par zones (2)

- Chargement global des programmes
  - Au lancement du programme
- Allocation dynamique
  - Si espace libre après zone du tas, l'allouer et changer la taille du registre de translation
  - Sinon, allouer nouvelle zone et copier contenu ancienne zone, puis changer adresse et taille registre de translation

Gestion espace virtuel par zones (3)

- Mémoire centrale étendue avec espace de va-et-vient ("swap space") sur disque
  - Stocker temporairement partie régions mémoire des programmes bloqués (données, tas et pile)
  - Code rechargeable depuis fichier binaire
- Allouer nouvelles zones mémoire après réveil du programme
  - Si nécessaire, doit vider programme bloqué dans espace de va-et-vient pour libérer de la mémoire

Gestion espace virtuel par zones (4)

- Technique simple au niveau matériel
- Pas souple au niveau gestion mémoire
  - Fragmentation potentielle
  - Allocation dynamique pas simple
- Coûteuse à l'exécution
  - Chargement global => latence au démarrage

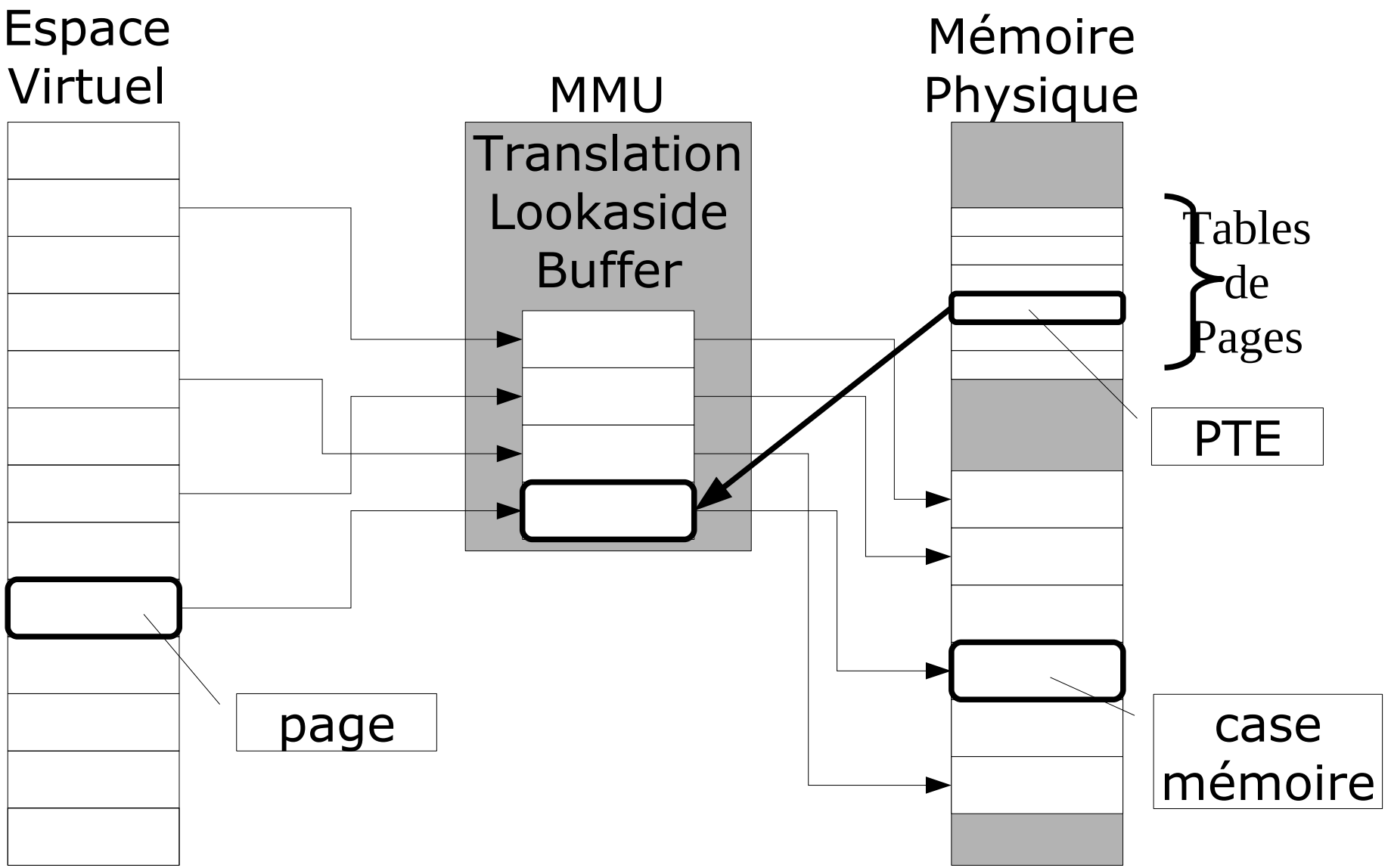
Plan

- Rôle de la mémoire
- Espace d'adressage
- Allocation régions mémoire de tailles variables
- Gestion de la mémoire centrale par zones
- Gestion de la mémoire centrale par pagination

Espace Virtuel Paginé (1)

- Mémoire centrale découpée en cases de taille fixe
- Espace virtuel divisé en pages de même taille
- Chaque page définie dans espace virtuel associée à une case de mémoire centrale
  - Opération d'association dynamique
  - 2 pages contigües ne sont pas nécessairement rangées dans des cases contigües
- Conversion adresse virtuelle => adresse physique
  - Réalisé par MMU (Memory Management Unit)

Espace Virtuel Paginé (2)



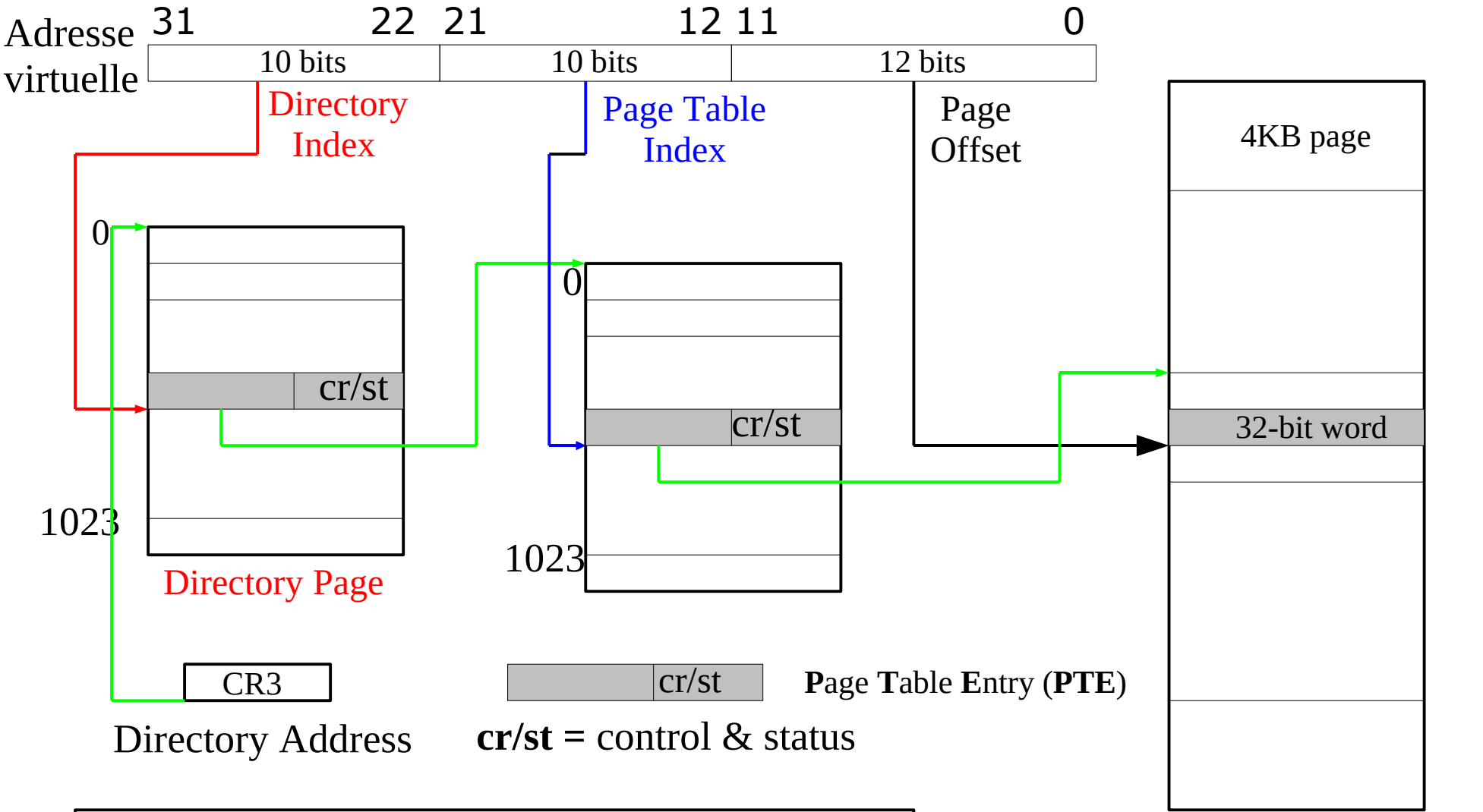
Espace Virtuel Paginé (3)

- MMU décompose adresse virtuelle en 2 parties
  - Partie déplacement dans la page (de taille maximum égale à la taille de page)
  - Partie numéro de page interprétée par MMU pour obtenir adresse de case correspondante
  - Adresse physique = adresse case | déplacement
- Numéro de page
  - Index direct dans table de pages
  - structure "arborescente" multi niveaux
    - Intel : 2 niveaux (1 Directory + Page Tables)

Espace Virtuel Paginé (4)

- Page Table Entry (PTE) décrit une page d'un espace virtuel
- Adresse de la page physique associée, si valide
- Bits de contrôle
  - Page physique associée valide
  - Droits d'accès (lecture, écriture, exécution)
- Bits de status
  - Page accédée
  - Page modifiée

MMU Intel


Adresse virtuelle
31
22
21
12
11
0
10 bits
10 bits
12 bits
Directory Index
Page Table Index
Page Offset
4KB page
0
cr/st
1023
Directory Page
0
cr/st
1023
32-bit word
CR3
Directory Address
cr/st
Page Table Entry (PTE)
cr/st = control & status
Translation Lookaside Buffer (TLB) = cache for PTEs
Mémoire physique

Espace Virtuel Paginé (5)

- Résoud problème de fragmentation de la mémoire centrale et de l'espace de va-et-vient
  - Gestion dynamique fine et efficace
- Autres avantages
  - Partage de pages entre espaces virtuels
  - Permet de "mapper" des fichiers dans espace virtuel d'un processus
- Inconvénients
  - Mécanisme matériel complexe
  - Fragmentation interne : pages non complètement occupées