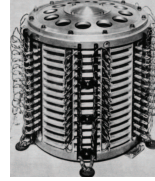


Exemple de configuration

ATLAS (1961) - Université de Manchester

- ▶ mots de 48 bits (8 caractères de 6 bits)
- ▶ 16K mots de mémoire à tores de ferrite (equiv. 96 Ko)
- ▶ 96 Kmots de mémoire à tambour (equiv. 596 Ko)



- ▶ Addition virgule fixe en $1.6\mu s$, multiplication virgule flottante $5\mu s$



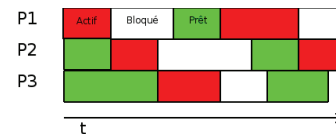
ça coûte très cher

Atlas

- ▶ commercialisé par Ferranti (Electronique, Manchester)
 - ▶ Ferranti Mark 1 (1951), premier ordinateur commercialisé
 - ▶ Meg (1954) virgule flottante
 - ▶ Mercury (1957), mémoire à tores de ferrite
- ▶ Prix commercial de l'Atlas : > 1 million de livres sterling

Q : comment rentabiliser ?

R : multi-programmation



(partage du temps)

Multi-programmation

Idées :

- ▶ plusieurs programmes en mémoire
- ▶ la gestion de périphériques est déléguée
- ▶ on profite des "temps morts"
- ▶ activation des processus à tour de rôle



⇒ meilleure gestion du temps
(Time is Money)

Mais : cohabitation
dans l'espace mémoire

Induit de nouveaux problèmes :

- ▶ ne pas gaspiller de place
- ▶ protéger les programmes
- ▶ sans compliquer la programmation

Partager
la mémoire

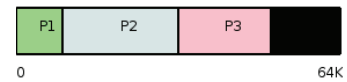
Espace mémoire



Où mettre les programmes ?

Idée ?

Espace mémoire



à la suite les uns des autres ?

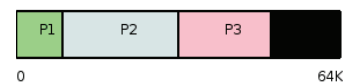
= allocation
contiguë

L'allocation contiguë

Principes

- ▶ Un processus : un espace en mémoire physique
- ▶ espace contiguë (plage d'adresses)

Espace mémoire



Problèmes

- 1. L'emplacement du code n'est connu qu'au chargement
- 2. Les espaces alloués sont
 - ▶ de tailles diverses
 - ▶ libérés dans un ordre imprévisible

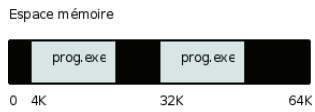
Plan

- 1. Problème : Indépendance code / position
- 2. solution : adresses logiques
- 3. mécanisme de génération d'adresses
- 4. protection de la mémoire

Indépendance code / position

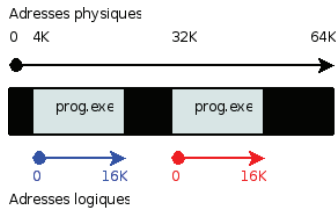
le problème

- ▶ Un exécutable peut être chargé à des emplacements différents (simultanément ou pas)



- ▶ Il doit fonctionner sans modification.

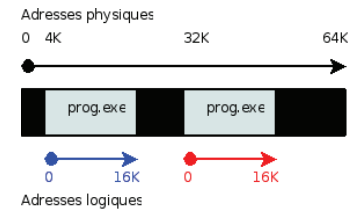
Solution :
adresses logiques



L'espace mémoire logique d'un processus commence toujours à son adresse zéro

Chaque processus utilise des adresses logiques

Génération
d'adresse
physique ?

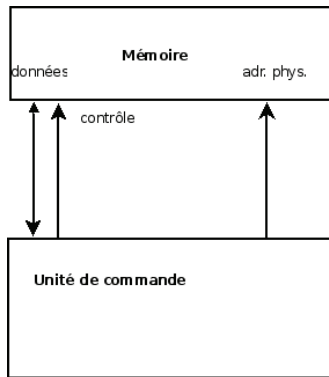


$$\text{adresse physique} = \text{adresse de base} + \text{adresse logique}$$

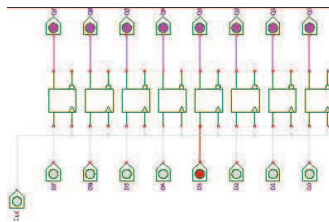
Génération d'adresse

la **génération d'adresse physique** est faite

- ▶ à chaque instruction
- ▶ par le matériel.



(architecture initiale)



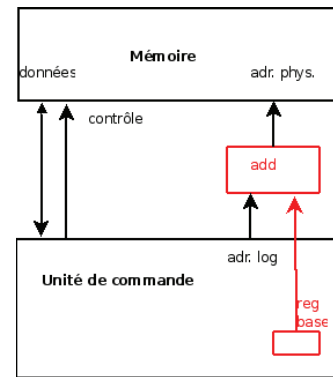
Registre 8 bits

(faible coût)

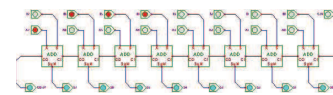
Modification du matériel

Génération d'adresse

- ▶ Un **Registre de base** contient l'**adresse de base** du processus
- ▶ **Additionneur** :
 - ▶ adresse présente sur le bus d'adresses logiques
 - ▶ + registre de base
 - ▶ ⇒ bus adresses physiques



avec registre de base

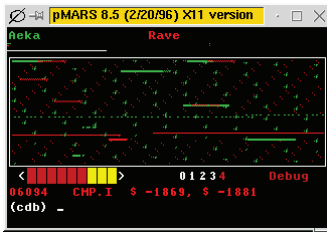


Additionneur 8 bits

Protection

Objectif

- ▶ Éviter qu'un processus n'accède à l'espace mémoire d'un autre



Jeu "Core War" (1984)

En vrai, on veut éviter.
Une idée ?

Protection

idée

une adresse logique **valide** est comprise entre

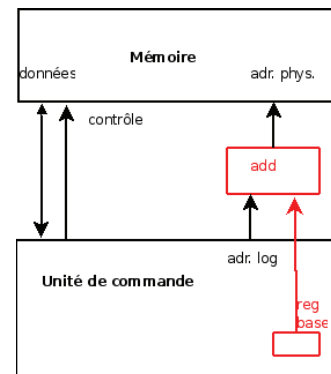
- ▶ 0
- ▶ la taille de l'espace logique (-1)

⇒ comparer
adresses logiques
et taille

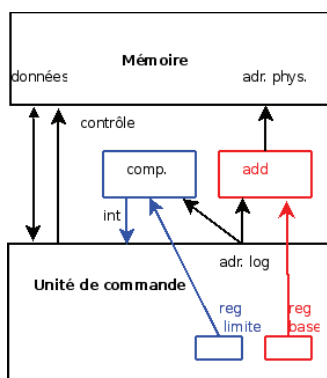
Modification du matériel (suite)

Vérification d'adresse logique

- ▶ **Registre limite** contenant la **taille de l'espace logique** du processus en cours
- ▶ **Comparateur** : bus adresses logiques + registre limite ⇒ interruption si dépassement



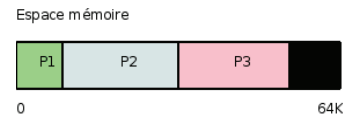
(avec registre de base)



avec registre limite

comparateur =
soustracteur
+ test de signe

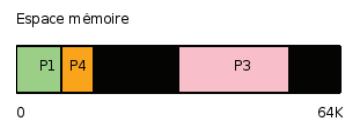
Exemple : un scénario de
 fragmentation :



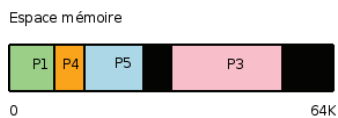
(0) 3 processus sont présents, P2 se termine ...



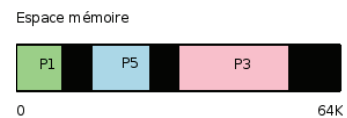
(1) allocation de P4 ...



(3) allocation de P5 ...



(4) libération de P4 ...



(5) l'espace libre est fragmenté



Remèdes à la fragmentation

Fragmentation :
 quels remède ?

Curatifs

- ▶ périodiquement, **translater** les zones allouées pour reconstituer une grande zone libre
- ▶ synonymes : **compactage**, relocation



Remèdes à la fragmentation (suite)

Préventifs

Limiter l'apparition de la fragmentation

- ▶ la stratégie d'allocation **first-fit** (premier bloc assez grand) est mauvaise
- ▶ la stratégie **best-fit** (plus petit bloc libre qui soit assez grand) est meilleure
- ▶ autres stratégies (buddy system)



Swapping

Plan

1. Idée, motivation
2. Avantages



Motivation

Applications interactives

(l'utilisateur est un périphérique très lent)



Avantages

- ▶ on peut charger beaucoup plus de processus en **mémoire virtuelle** que la mémoire centrale ne peut en contenir
- ▶ permet d'attendre des taux de charge élevés (rentabilisation)



Résumé

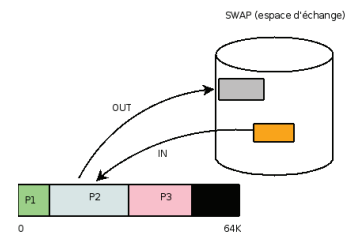
- ▶ Fragmentation consécutive à l'allocation de zones **contiguës** de **tailles diverses**
 - ▶ ce problème se retrouve ailleurs (gestion des disques, new, delete en programmation, ...)
 - ▶ les solutions préventives et curatives (best-fit, translation) sont plus ou moins satisfaisantes
- en pratique, on s'y prend autrement (voir + loin).



Swapping

Une autre idée

- ▶ Transférer sur disque (ou tambour) les processus inactifs
- ▶ les ramener le moment voulu



Echanges mémoire ↔ disque



Remarque

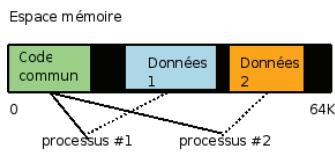
Même problème de gestion d'espace sur disque ?



Segmentation

Plan

1. Objectif
2. Idée de base
3. Généralisation
4. Tables locales / Globale



Deux processus exécutent le même code

Segmentation

Idée de base

- ▶ L'espace mémoire d'un processus comporte au moins deux **segments** :
 - ▶ le segment de **code**
 - ▶ le segment de **données**.
 - ▶ ...
- ▶ le segment de code est **commun** aux différentes instances du programme ;
- ▶ on préfère charger une seule instance du code en mémoire
- ▶ segment **partagé** entre les processus

Une idée de réalisation

Différencier 2 types d'adresses, selon le premier bit

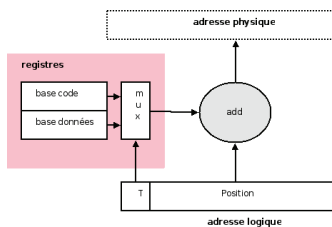
Bits d'une adresse



T=0 : position dans le segment de code
 T=1 : position dans le segment de données

génération d'adresses, comment ?

2 registres de base (code, données)



un **multiplexeur** sélectionne le registre de base à utiliser

On peut généraliser l'idée

Segmentation

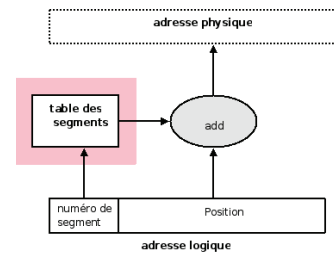
Généralisation

Une adresse logique comporte

- ▶ un **numéro de segment**
- ▶ une position (**offset**) dans le segment

La table des segments

- ▶ contient les **adresses de base** des segments.
- ▶ on y trouve aussi les tailles, les droits d'accès etc.



utilisation d'une table des segments

81/104

87/104

En fait,
c'est peu plus
compliqué

numéros de segment
- locaux
- globaux

83/104

84/104

Numéros de segments

Numérotation

- ▶ locale : propre à chaque processus
- ▶ globale : commune

Deux processus peuvent utiliser le même segment réel avec des numéros locaux (éventuellement) différents.

Exemple

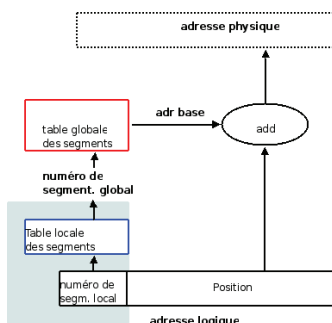
- ▶ P1 : code1(0), **bib. trigo(1)**, données2(2)
- ▶ P2 : code2(0), bib. matrices(1). **bib. trigo(2)**, données2(3)

85/104

86/104

Tables des segments

- ▶ La **table locale des segments** (TLS) d'un processus fournit le **numéro de segment global** à partir du **numéro de segment local**
- ▶ La **table globale de segments** (TGS) contient la description de chaque segment (longueur, pages, protections ...)



Génération d'adresses, TLS + TGS

87/104

88/104

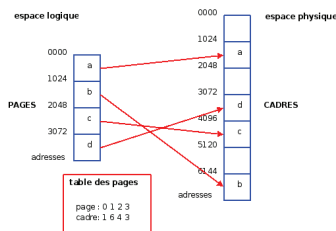
Résumé

- ▶ **adresse logique** : numéro de segment, et position
- ▶ utilisation d'une **table des segments**
- ▶ permet un gain de place important : on utilise les segments pour les **bibliothèques communes**.

Espace mémoire paginé

Plan

1. Idée
2. Table des pages
3. Génération d'adresses



espace paginé



Génération d'adresses

- ▶ le **numéro de page** provient des bits de poids forts
- ▶ il est converti en numéro de **cadre de page**
- ▶ combinaison avec les bits de poids faible (position dans la page)



Espace mémoire paginé

Contexte

Partage de la mémoire physique par des processus

Problème

causé par l'allocation et la libération d'espaces de tailles diverses.

Idée

- ▶ espace découpé en **pages de mêmes tailles**
- ▶ les pages d'un même espace ne sont pas forcément consécutives
- ▶ la **table des pages** indique la position en mémoire des pages d'un processus



Plus précisément

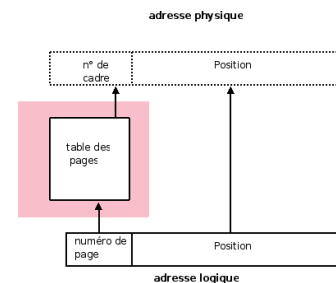
- ▶ espaces logiques découpés en **pages logiques** : habituellement 1K mots, 2K, 4K, 8K ...
- ▶ la mémoire physique est découpée en **cadres de pages** de la même taille.
- ▶ pour chaque processus, une **table des pages** établit la correspondance



Bits d'une adresse

Numéro de page	Position dans la page
----------------	-----------------------

format d'adresse logique



génération d'adresse



Mémoire virtuelle paginée

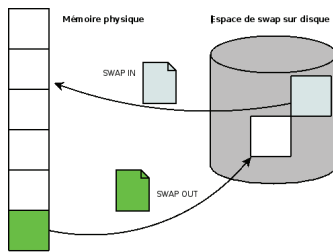
Plan

1. Mémoire virtuelle paginée
2. Défaut de page
3. Remplacement de page
4. Algorithmes

◀ ▶ ⏪ ⏩ 🔍 🔄

Circuit spécialisé MMU (memory management unit)

◀ ▶ ⏪ ⏩ 🔍 🔄



Swapping

◀ ▶ ⏪ ⏩ 🔍 🔄

Mémoire virtuelle segmentée-paginée

Plan

1. Principe
2. génération d'adresses

◀ ▶ ⏪ ⏩ 🔍 🔄

Mémoire virtuelle paginée

Principe

Combinaison de la pagination et du swapping.

- ▶ la table des pages connaît les pages présentes
- ▶ interruption en cas de **défaut de page** (la page référencée n'est pas en mémoire)

◀ ▶ ⏪ ⏩ 🔍 🔄

Pagination, suite

En cas de défaut de page

Le système d'exploitation reprend la main,

1. il trouve un emplacement libre (au besoin, évacuation sur disque)
2. y installe la page manquante,
3. met à jour la MMU
4. et relance l'instruction interrompue.

◀ ▶ ⏪ ⏩ 🔍 🔄

Algorithmes de remplacement de page

Pour choisir un emplacement à libérer, en général on choisit les pages "les moins utiles".
Diverses techniques existent. Voir TD.

◀ ▶ ⏪ ⏩ 🔍 🔄

Mémoire virtuelle segmentée paginée

Une adresse virtuelle comporte un numéro de segment, et un déplacement.

- ▶ chaque processus possède ses numéros de segment.
- ▶ chaque segment est composé de pages.

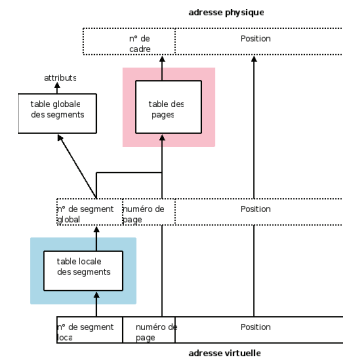
◀ ▶ ⏪ ⏩ 🔍 🔄

Implémentation

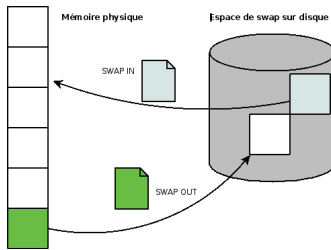
- ▶ une table (TLS) traduit le numéro de segment local en numéro de segment global.
- ▶ le couple (numéro de segment global, numéro de page) permet de retrouver le numéro de cadre de page correspondant
- ▶ le numéro de cadre de page et la position dans la page forment l'adresse physique.

La MMU utilise une *mémoire associative* pour mémoriser les correspondances

(num. de segment global, num. de page → num. de cadre de page)



génération d'adresse



Combinaison avec swapping



IBM 360/67 (1969)

Résumé

- ▶ Combinaison de
 - ▶ pagination,
 - ▶ segmentation,
 - ▶ swapping
- ▶ permet
 - ▶ la cohabitation entre les processus
 - ▶ la protection
 - ▶ zones partagées
- ▶ technique largement adoptée depuis les années 70