610 Pagination

INF3173
Principes des systèmes d'exploitation

Jean Privat

Université du Québec à Montréal

Hiver 2021

Pagination

Principe

- Découper toute la **mémoire physique**
  En **page physique** (ou **cadres**, *frame*, *page frame*) de taille fixe
  `sysconf(_SC_PAGESIZE)` donne la taille des pages du système
  (4ko par exemple)
- Découper tout l'**espace d'adressage** des processus
  En **pages logiques** (ou page virtuelle) de même taille
- Associer **efficacement** (CPU) les pages logiques aux physiques

Le gagnant actuel

- Offert par la plupart des processeurs
- Utilisé par la plupart des systèmes d'exploitation
- Assez couteux et complexe côté processeur (INF4170)
- Simple, souple et puissant côté système d'exploitation

Pagination pour le MMU

Adresse logique décomposée

- Numéro de page logique
- Adresse dans la page (décalage ou offset)
- Exemple: page de 4ko, 48 bits d'adresse logique =
  36 bits (numéro de page logique) + 12 bits (décalage (2^{12}=4k))

Traduction avec une table

- Associer numéro de page logique $\rightarrow$ numéro de page physique
- Les associations sont stockées dans la **table des pages**

Exemple très naïf

Pages

- Taille: 4 octets
- Taille du décalage (en bits):

Physique

- Adresse: 5 bits
- Espace d'adressage (octets):
- Nombre de pages:

Logique

- Adresse: 4 bits
- Espace d'adressage (octets):
- Nombre de pages:

Exemple très naïf (suite)

- Page: 4 octets. Adresse physique: 5 bits. Adresse logique: 4 bits

Pagination pour le système d'exploitation

- Une table des pages par processus
- Le système d'exploitation
  - Configure et maintient chaque table des pages
  - Positionne la table du processus actif lors des changements de contextes

Chez Linux

- `/proc/PID/pagemap` (tableau binaire) pour chaque page logique, quelle page physique est associée (+ info supplémentaire)
- `/proc/kpagecount` (tableau binaire) pour chaque page physique, combien de pages logiques y sont associées

En vrai

- La mémoire des processus est **beaucoup** plus riche que ce que le processeur offre
- La table des pages du MMU est trop bas niveau, trop spécifique et trop limitée
- Des structures de données additionnelles sont nécessaires
- Le système gère des zones virtuelles regroupant plusieurs pages: les lignes de `pmap(1)`
- On y reviendra...

Table des pages

Où est la table ?

- Registres ? Non, la table est trop grande !
- Un gros bloc en mémoire ? Où est ce bloc ?

Solution habituelle

- Registre privilégié pour l'adresse de la table (CR3 chez x86)
- Tables d'indirection en RAM

Questions

- L'adresse dans CR3 est-elle logique ou physique ?
- Un processus peut-il modifier la valeur du registre CR3 ?
- Un processus peut-il modifier la table des pages ?

Avantage de la pagination

Souplesse maximale

- Permet de mettre différents morceaux de mémoire
- Permet d'utiliser tout l'espace d'adressage (ou presque)
- Indépendant du nombre et de l'utilisation des morceaux
- Possibilité d'avoir des droits fins (lecture, écriture, exécution)
- Possibilité de partager des pages physiques entre processus
  - Pas forcément avec la même page logique
  - Pas forcément avec les mêmes droits
  - On y reviendra...

Question

- Une page physique peut-elle être associée à plusieurs pages logiques différentes ?

Trop couteux en espace !

- Une seule table d'indirection ne passe pas à l'échelle

Exemple

- 48 bits d'adressage logique
  - 36 bits de numéro de page logique
  - 12 bits de décalage
- 8 octets par entrée de la table
  - Adresse de base de la page physique
  - Métadonnées (droits de la page, etc.)
  - Bits réservés
- Taille de la table des pages ?

Trop couteux en espace !

- Une seule table d'indirection ne passe pas à l'échelle

Exemple

- 48 bits d'adressage logique
  - 36 bits de numéro de page logique
  - 12 bits de décalage
- 8 octets par entrée de la table
  - Adresse de base de la page physique
  - Métadonnées (droits de la page, etc.)
  - Bits réservés
- Taille de la table des pages ?
  - 2^{36} * 8o =
  - 512Go par table =
  - 512Go par processus (oups!)

Pagination multi-niveaux

- Découper l'adresse logique en plusieurs morceaux
- L'adresse d'une table + un morceau donne un champ dans la table
- Chaque champ d'une table indique
  - Soit l'adresse de la table suivante à consulter
  - Soit qu'il n'y a pas de table suivante: faute CPU

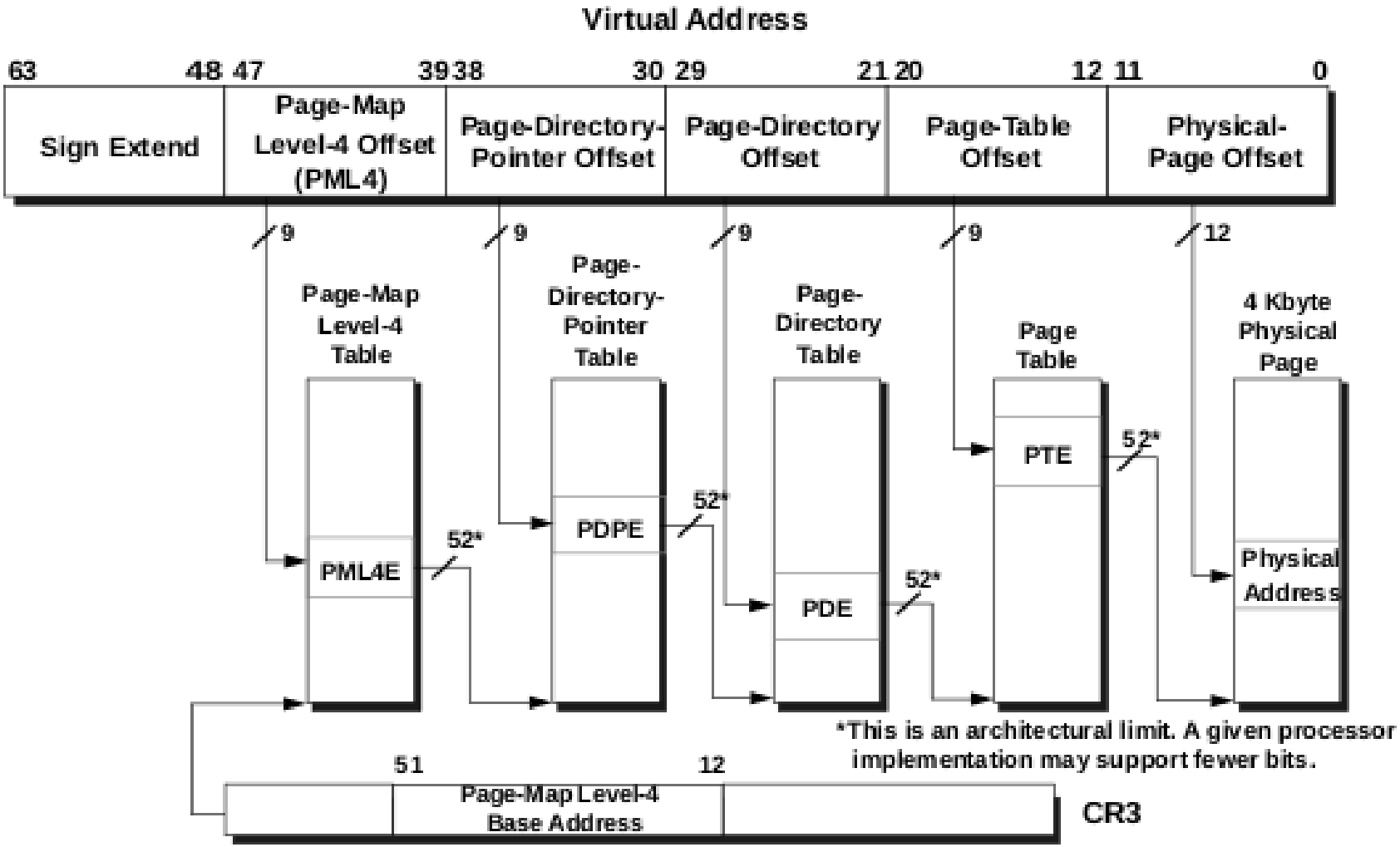

Figure 5-17. 4-Kbyte Page Translation—Long Mode

Source: AMD64 Architecture Programmer's Manual, Volume 2 System Programmation, page 136, rev. 3.36, octobre 2020, AMD Corporation.

Multi-niveau : pourquoi on y gagne ?

- L'espace d'adressage des processus est plein de vide
- Ne remplir que les tables intermédiaires nécessaires

$\longrightarrow$ Élagage de l'arbre des tables de pages

Dans la vraie vie

Beaucoup de détails techniques (et historiques)

Plusieurs schémas possibles, et configurables

- x86-64: souvent 48bits d'adressage sur 4 niveaux (mode long 4k)
- 57bits d'adressage sur 5 niveaux chez de récents processeurs Intel

Taille des pages variable

- Plusieurs tailles et schémas peuvent cohabiter en même temps
- x86-64: 4ko, 2Mo, 1Go

Autres fonctionnalités

- Peut se combiner avec la segmentation (x86)
  - Adresse logique $\rightarrow$ adresse linéaire $\rightarrow$ adresse physique
- Métadonnées supplémentaires (on y reviendra)...

Trop couteux en temps !

- Accéder à la mémoire coute trop d'accès mémoire

Exemple

- 4 niveaux : 4 tables + RAM finale
- Un accès mémoire logique = 5 accès mémoire physique
  - Chercher dans 4 tables + la donnée finale
  - Plus 5 additions, 4 vérifications des droits, etc.
- Les performances sont divisées par 5
- Et ceci pour chaque accès à la mémoire

TLB et caches processeurs

TLB (*translation lookaside buffer*)

- Cache les dernières traductions logiques $\rightarrow$ physiques
- Cas idéal fréquent : 0 accès mémoire pour traduire
- Cas pas idéal rare: faire toutes les indirections nécessaires

Caches CPU

- Cache le contenu de la RAM
- Évite l'accès à la RAM complètement

Les détails dans un autre cours

- INF4170 - Architecture des ordinateurs