RAM :

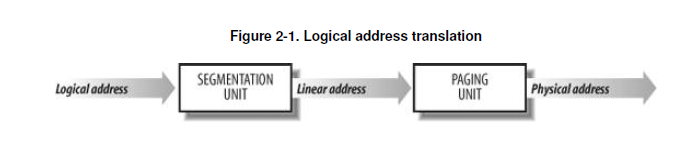
Une partie de la RAM fait office de cache

Une partie de la RAM stocke l’image du noyau et ses structures

Une partie de la RAM stocke les variables kernel dynamiquement allouée.

Le reste pour l’espace d’adresse user.

Mémoire dans Linux :



MMU fait la translation entre adresses physiques et adresses logiques (mémoire virtuelle)

Memory arbiter garantit les accès concurrents à la RAM (plusieurs CPU, DMA…)

**Physical memory**

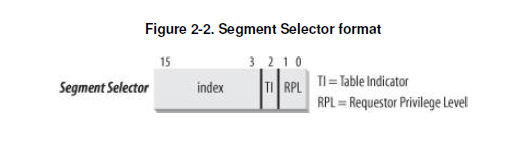
* Nodes : bande de mémoire référencé par pg\_data\_t (struct pglist\_data) dans linux/mmzone.h. Tous les nodes sont référencé dans une pgdat\_list . La liste est chainée le next est situé dans pg\_data\_t
* Un node est divisé en zones référencé par zone\_t (struct zone\_struct) dans linux/mmzone.h

Une zone est un groupement de pages avec des propriétés similaires.

* + ZONE\_DMA : @basses : first 16 Mb. Si le HW peut adresser la DMA sur any @, ZONE\_DMA est absente et ZONE \_NORMAL le remplace.
  + ZONE\_NORMAL : 16 -> 896 Mb
  + ZONE\_HIGHMEM -> >896 Mb : pages mappées dynamiquement

**Segmentation : tailles de segment variables**

16 bits pour le segment identifier (ou segment selector) et 32 pour l’offset dans le segment



CS : codesegment register

SS : stack segment register

DS : data segment register

**Pagination : tailles fixe**

Page (mémoire virtuelle) et frames (mémoire physique) de 4 ou 8 kb en RAM

Gère les droits.

Table des pages gère la correspondance

Page frame référencé par struct page et toutes les pages dans un tableau mem\_map

struct page {

unsigned long flags;

atomic\_t \_count;

atomic\_t \_mapcount;

unsigned long private;

struct address\_space \*mapping;

pgoff\_t index;

struct list\_head lru;

void \*virtual;

};

Flags : si dirty, locked… (32 flags au total)

Virtual : @virtuelle de la page (NULL si dans HIGH MEMORY)

Allocation de pages :

struct page \* alloc\_pages(gfp\_t gfp\_mask, unsigned int order)

Conversion virtuelles -> physique

void \* page\_address(struct page \*page)

Allocation de pages kernel :

void \* kmalloc(size\_t size, gfp\_t flags)