Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2

p.05 - Memory Management Unit

Classes T-2/I-2 // 2017-2018

Daniel Gachet | HEIA-FR/TIC

p.05 | 11.04.2018

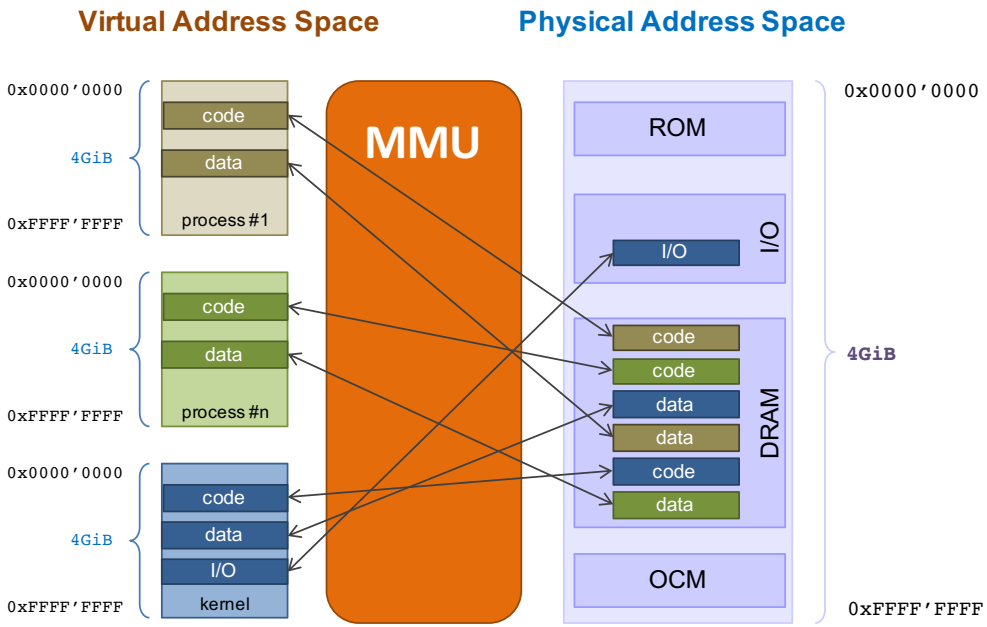
Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

Contenu

- Introduction
- Translation d'adresses
- Cache de la table de translation
- Mémoire cache vs MMU
- Implémentation du TI AM335x / Cortex-A8

Introduction

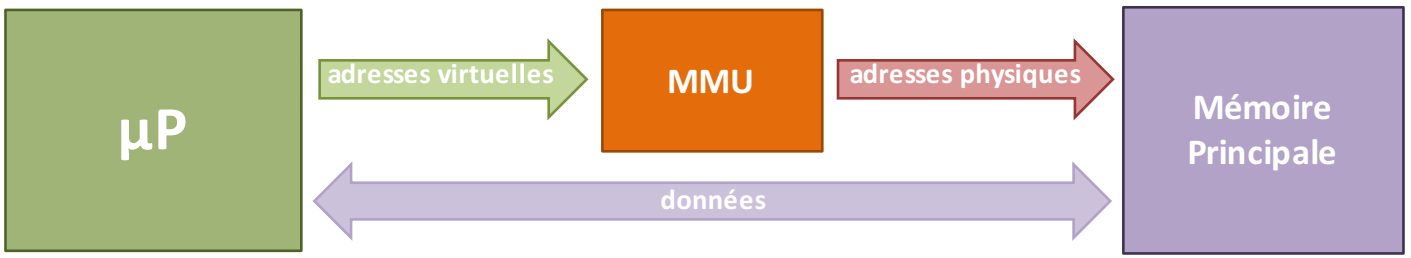
- Pour faciliter la gestion des tâches qu'un système informatique doit exécuter et pour augmenter la robustesse et la fiabilité, les systèmes d'exploitation modernes gèrent ces différentes tâches (processus) comme des applications indépendantes fonctionnant dans leur propre espace virtuel



- Dans ce contexte, la MMU (Memory Management Unit) est chargée de traduire les adresses virtuelles en adresses physiques

Espaces d'adressage

- Les accès à la mémoire principale du µP sont gérés par l'unité de gestion mémoire (MMU ⇔ Memory Management Unit)



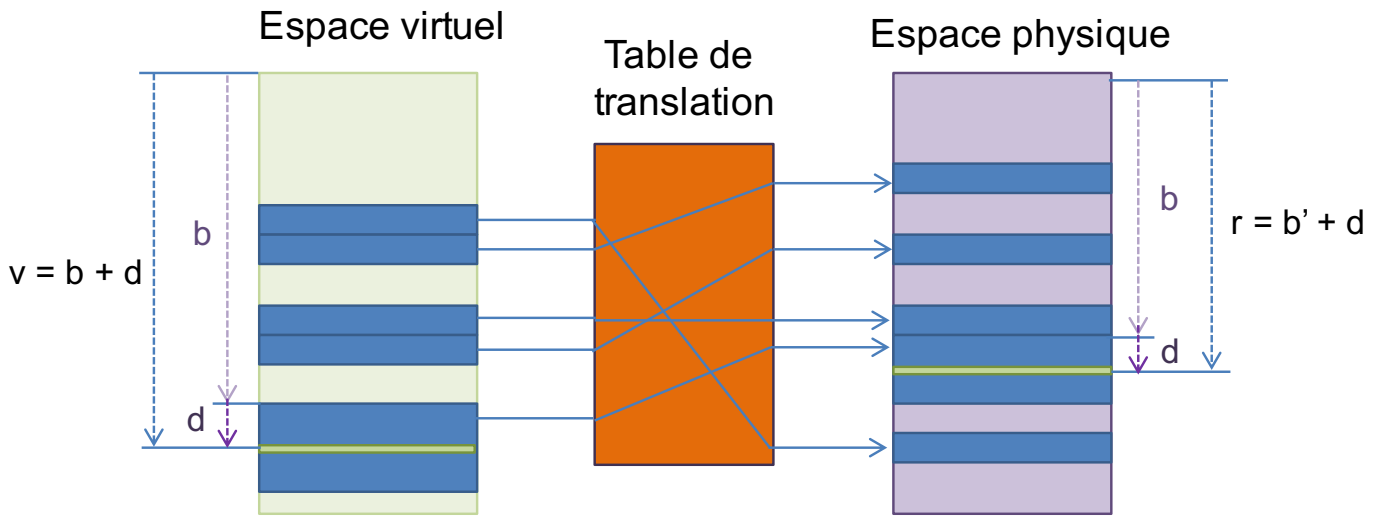
- Vue du µP, les adresses sont appelées adresses virtuelles (virtual address) ou adresses logiques (logical address)
- Vue de la mémoire principale, les adresses sont appelées adresses physiques (physical address) ou adresses réelles (real address)

Fonctions

- En plus de la traduction d'adresses virtuelles en adresses physiques, la MMU a deux fonctions supplémentaires
  - La protection des accès à la mémoire physique :
    elle procure une isolation hermétique du processus, en garantissant que le processus ne puisse pas sortir de son espace virtuel et accéder aux données d'autres processus
  - La gestion de la mémoire cache :
    elle décide quelles données peuvent être stockées dans la mémoire cache, car toutes les données traitées par le µP ne peuvent pas l'être, p. ex. les données contenues dans les registres des périphériques

Translation d'adresses

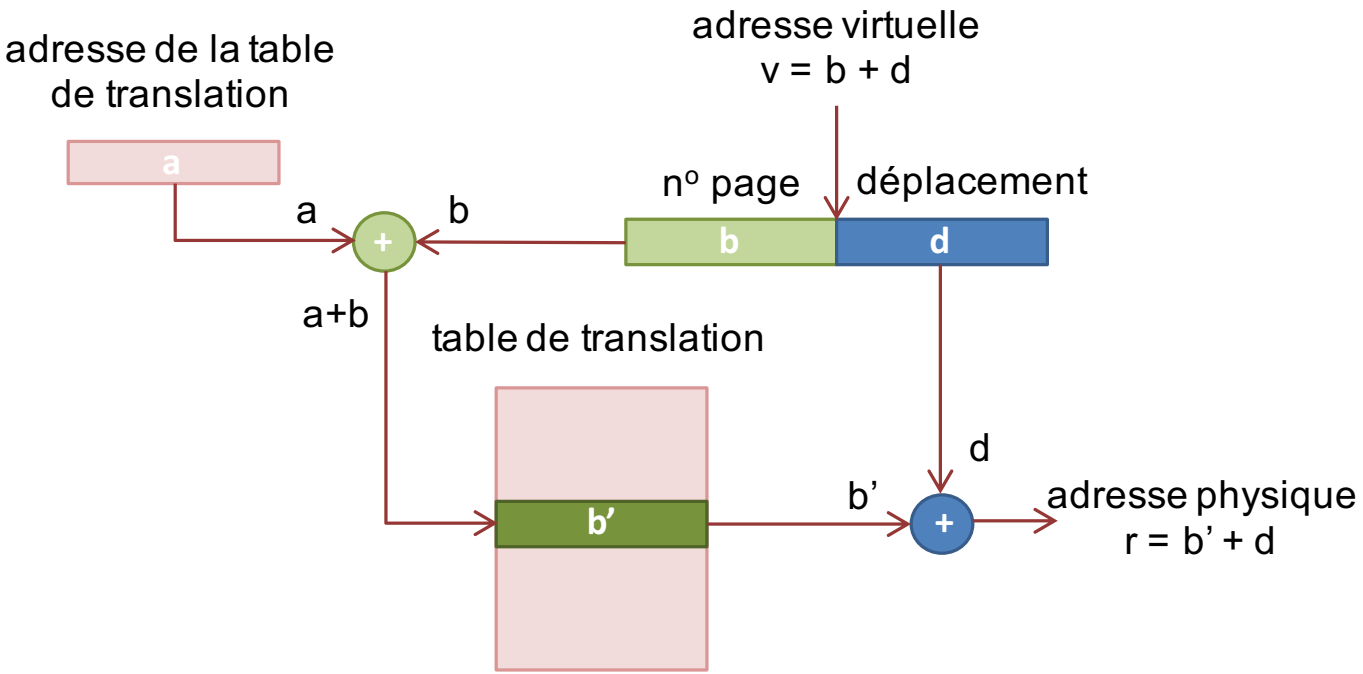
- Pour convertir une adresse virtuelle $(v = b + d)$ en une adresse physique $(r = b' + d)$, la MMU utilise une table de translation



- La table de translation se situe dans la mémoire principale

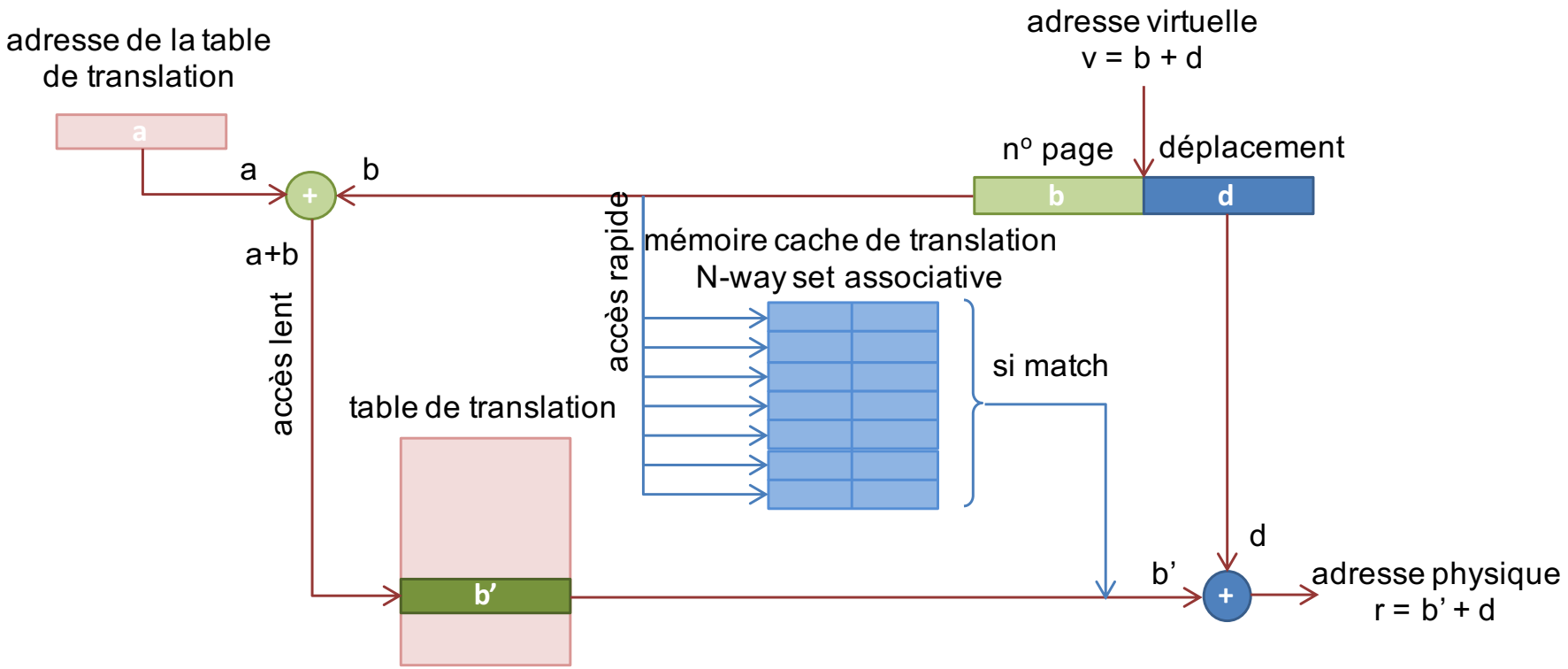
Translation d'adresses (II)

- Les bits supérieurs de l'adresse virtuelle sont utilisés pour accéder la table de translation, laquelle contient les adresses physiques



Cache de la table de translation

- Pour diminuer les temps d'accès sur la table de traduction située dans la mémoire principale, une partie de celle-ci est copiée dans une mémoire à l'intérieur de la MMU (du µP)
- Cette mémoire est nommée mémoire cache de translation (Translation Lookaside Buffer ⇔ TLB)

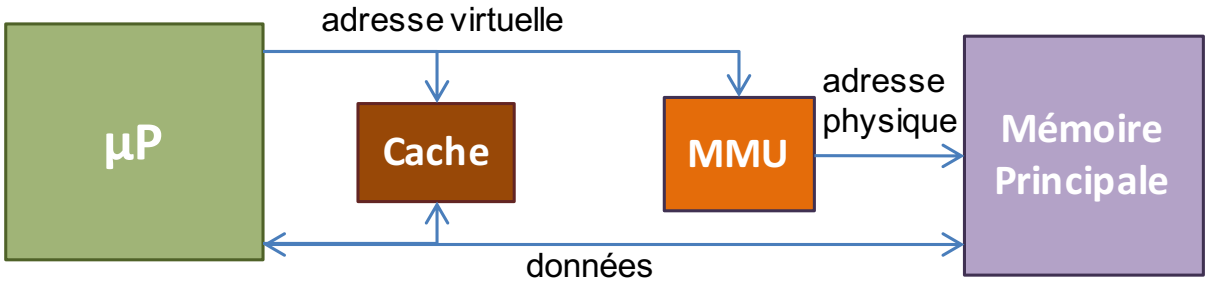


Mémoire cache vs MMU

- Les mémoires caches du processeur sont intimement liées à la MMU
  - Si la mémoire cache est située avant la MMU, on parle de cache virtuelle (virtual cache)
  - Si la mémoire cache est située après la MMU, on parle de cache physique (physical cache)

Cache virtuelle

- Les caches virtuelles utilisent les adresses virtuelles pour accéder les données stockées dans la mémoire cache
- L'utilisation de l'adresse virtuelle permet d'éviter la translation d'adresse pour accéder les données de la mémoire cache



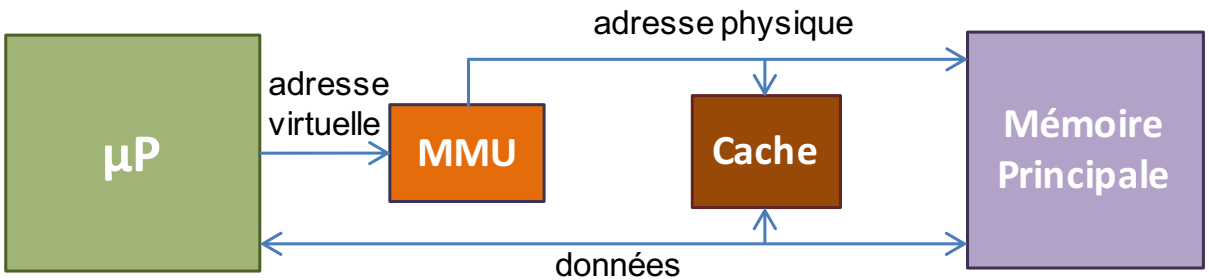
\+ temps d'accès très court

\- ambigüité des adresses synonymes

\- synchronisation de la mémoire principale avec le contenu de la cache nécessaire

Cache physique

- Les caches physiques utilisent les adresses physiques pour accéder les données stockées dans la mémoire cache
- L'utilisation de l'adresse physique nécessite la translation d'adresse pour accéder les données de la mémoire cache

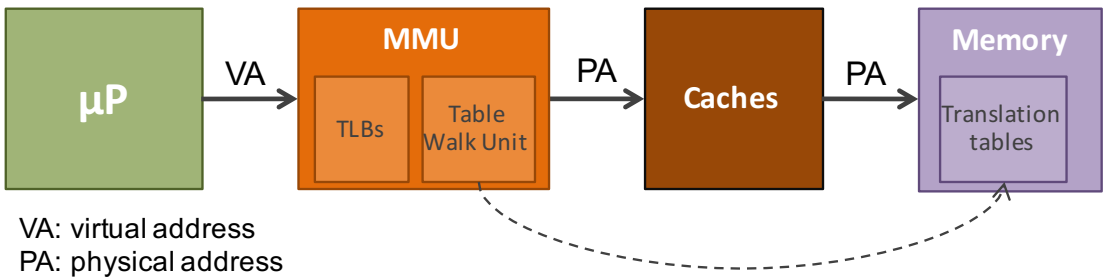


\+ pas manipulation de la cache lors de la commutation de processus

\- temps d'accès plus long

Implémentation du TI AM335x

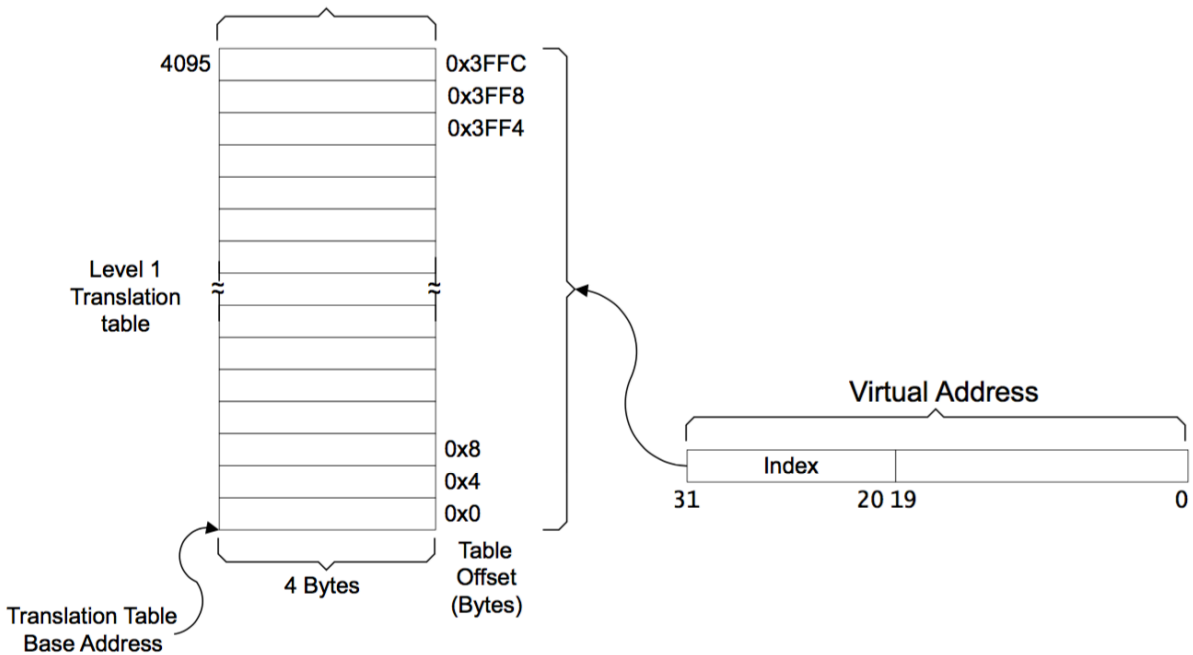
- En plaçant la MMU entre le µP et la mémoire cache, le µP TI AM335x basé sur un µP ARM Cortex-A8 implémente une cache physique



- Caractéristiques
  - TLB - Translation Lookaside Buffer
    - 32 entrées pour les instructions et 32 entrées pour les données
    - cache complètement associative
  - Implémentation de la table de translation sur 2 niveaux
    - 1er niveau : pointeurs sur 2e niveau, section de 1MiB, 16MiB et fault
    - 2e niveau : pages de 4KiB, 64KiB et fault
  - FIFO (round-robin) comme algorithme de remplacement

Implémentation du 1er niveau

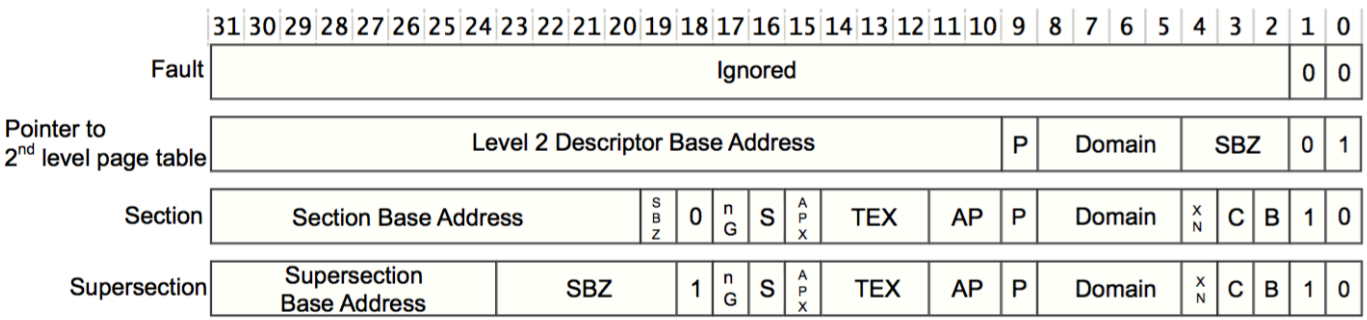
- La table de translation du 1er niveau compte 4096 entrées
- Les 12 bits de poids fort (bits 20 à 31) de l'adresse sont pris comme index pour adresser la table de translation



Ref : 03_cortex_a_programmer_guide.pdf, figure 9.4

Implémentation du 1er niveau (II)

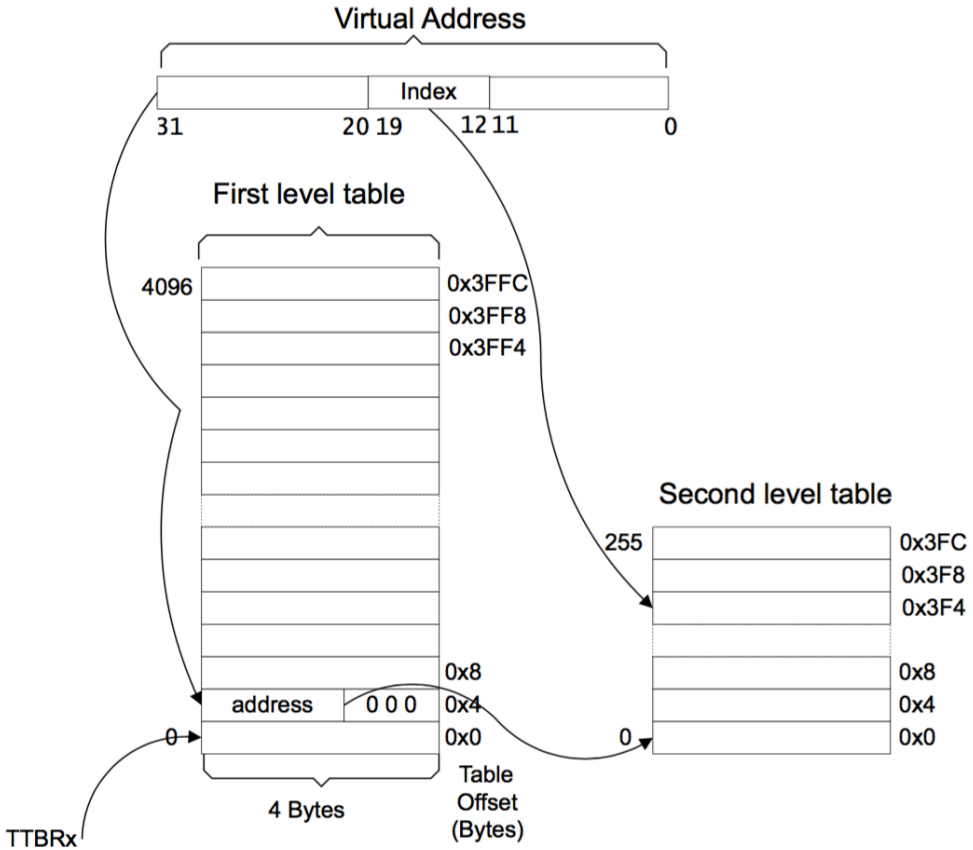
- Chaque entrée de table offre un des 4 types possibles de sections
  - section ”fault” produisant une erreur
  - pointeur sur un page du 2e niveau
  - section de 1MiB
  - section de 16MiB (utilise 16 entées de 1MiB et doit être alignée sur 16MiB)



Ref : 03_cortex_a_programmer_guide.pdf, figure 9.5

Implémentation du 2e niveau

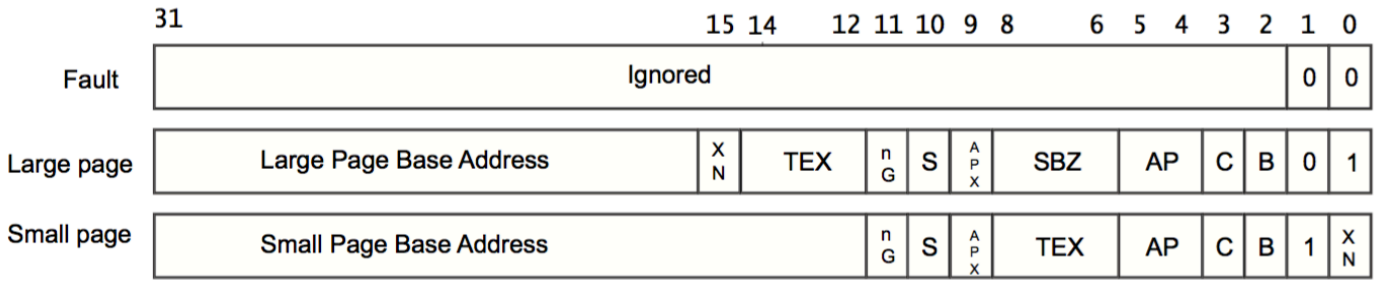
- La table de translation du 2e niveau compte 256 entrées
- Les bits 12 à 19 de l'adresse sont pris comme index pour adresser la table de translation



Ref : 03_cortex_a_programmer_guide.pdf, figure 9.9

Implémentation du 2e niveau (II)

- Chaque entrée de table offre un des 3 types possibles de pages
  - page ”fault” produisant une erreur
  - page de 4KiB
  - page de 64KiB (utilise 16 entées de 4KiB et doit être alignée sur 64KiB)



Ref : 03_cortex_a_programmer_guide.pdf, figure 9.8