

Séance 4

Gestion de la mémoire d'un système embarqué



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

Rappels

- **Gestion des processus** et kernel gérant le multitâche
 - Définition et manipulation de Process Control Block (PCB)
 - Démarrage d'un programme et changement de contexte
 - Exemple de gestion de processus dynamiques
- **Ordonnancement** de processus sur le système

Terminologie, algorithme et cas des systèmes embarqués
- Coordination de processus par **synchronisation**

Mécanismes Sleep/Wakeup et sémaphore

Objectifs

- Fonctionnement du **Memory Management Unit**
 - Registres et activation du MMU
 - Contrôle d'accès de domaine et statut d'erreur
- **Adresse virtuelle** et traduction en adresse physique
 - Table de traduction des sections et pages
 - Traductions de références de sections et pages

Espace d'adresses

- Exécution du **gestionnaire de reset** au démarrage et au reset
 - Code du gestionnaire exécuté en mode superviseur (SVC)
 - Copie de la table des vecteurs à l'adresse 0
 - Initialisation de piles pour interruptions et gestion d'exceptions
- Exécution du **programme de contrôle** du système
 - Même espace d'adresses dans modèle de processus statiques
 - Tâche mal conçue est un danger pour tout le système

Mode d'exécution

- Importance de pouvoir isoler l'exécution des processus

Utilisation des modes d'exécution proposé par ARM

- Mode user sans privilèges avec trois voies de sortie

- Mode privilégié spécifique pour la gestion des exceptions
- Mode FIQ ou IRQ pour la gestion des interruptions
- SWI produit un passage en mode supervisé (SVC)

- Gestion des appels systèmes à l'aide de SWI

MMU

Memory Management Unit

- Deux missions principales du **Memory Management Unit**
 - Traduction d'adresses virtuelles en adresses physiques
 - Contrôle des accès mémoire en vérifiant les permissions
- **Trois composants hardware** pour réaliser ces fonctions
 - Un Translation Lookahead Buffer (TLB), mémoire cache rapide
 - De la logique pour réaliser le contrôle d'accès
 - De la logique pour parcourir la table de traduction

MMU de ARM

- Accès mémoires sur base de **sections ou de pages**
 - Pagination à un niveau pour gestion par sections
 - Pagination à deux niveaux pour gestion par pages
- Niveau 1 de description de **tables de pages** au niveau 2
 - Deux tailles de pages supportées (4 Ko et 64 Ko)
 - Chaque page est découpée en quatre sous-pages
- Support de **domaines** pour définir des droits d'accès

Possibilité de définir 16 domaines différents

Registre MMU

- MMU traité comme un **co-processeur** pour le processeur ARM
 - Onze registres de 32 bits pour contrôler ses opérations
 - Accès aux registres du MMU avec instructions MRC et MCR
- Principaux **registres** présents dans le MMU
 - Registre de contrôle pour configurer le MMU (*C1*)
 - Translation Table Base Register (*C2*)
 - Domain Access Control Register (*C3*)
 - Fault Status Register (*C5*) et Fault Address Register (*C6*)
 - TLB Operations Register (*C8*)

Activation MMU

- Activation du MMU via le bit M du registre $C1$

Bit remis à zéro lors d'un reset, désactivant le MMU

- Trois étapes à exécuter pour activer le MMU

- 1 Programmer les registres TTB et DAC
- 2 Programmer les descripteurs tables de pages (niveaux 1 et 2)
- 3 Activer le MMU en passant le bit M de $C0$ à 1

- Désactivation du MMU en passant bit M de $C0$ à 0

En désactivant au préalable le cache de données

Contrôle d'accès de domaine

- Total de **16 domaines** définis par deux bits dans le DAC
 - Un client utilise le domaine
 - Un manager contrôle le comportement du domaine
- **Quatre valeurs** possibles pour chaque domaine
 - 00 pas d'accès, provoque un *domain fault* le cas échéant
 - 01 client, accès contrôlé par bits AP (section ou page)
 - 10 pas d'accès (réservé)
 - 11 manager, pas de vérification par rapport à AP

Statut d'erreur

- Source de la **dernière erreur** d'instruction/donnée dans FSR

Information liée au type d'erreur (traduction, accès...)

- Quatre bits indiquent le **statut de l'erreur**

Quatre bits concernent le domaine lors de l'erreur

Priorité	Source	Taille	Statut	Domaine
Haute	Alignement	–	00x1	Invalide
	Abandon externe	Premier niveau	1100	Invalide
		Second niveau	1110	Valide
	Traduction	Section	01x1	x
	Domaine	Section	11x1	x
	Permission	Section	11x1	x
	Abandon externe	Section ou page	10x0	Invalide
	Basse			

Adresse virtuelle



Traduction d'adresse

- Traduction d'adresse virtuelle du CPU en adresse physique
 - Nécessaire pour accès mémoire externe par le CPU
 - Dérivation et vérification des permissions d'accès
- Information de traduction dans la **table de traduction**

Située dans la mémoire physique, pointée par registre TTB
- Deux niveaux de descripteurs dans la table de traduction
 - Descripteur de tables des pages réfère à un second niveau
 - Descripteur de sections pour accès direct à la bonne section

Table de traduction

- Table de traduction est celle de **premier niveau**

Zone totale de 16 Ko avec 4096 entrées de 4 octets

- Chaque entrée correspond à un **descripteur**

Base d'une section ou d'une table des pages de niveau 2

- **Deux bits de poids faible** définissent le type de descripteur

- 00 pour signaler une faute et 11 est réservé
- Type 01 définit une table et 10 définit une section

Référence de section

- **Descripteur de section** identifie un bloc de 1 Mo de mémoire
 - Bits 31:20 adresse de base de la section de 1 Mo
 - Bits 11:10 donne les permissions d'accès (AP)
 - Bits 8:5 spécifie l'un des 16 domaines possibles
 - Bits 3:2 contrôlent le cache et le buffer d'écriture
- **Permissions d'accès** dépendent de la configuration du MMU

Bits S et R, pour contrôle en mode user ou supervised

31	20 19	12 11 10 9 8	5 4 3 2 1 0	
Adresse de base section		AP	Domain	1 C B 1 0

Traduction référence de section

- **Adresse virtuelle** (VA) en deux parties
 - Bits 31:20 représentent indice dans la table (12 bits)
 - Bits 19:0 représentent offset dans la section (20 bits)
- **Adresse physique** (PA) construite en concaténant
 - Bits 31:20 de l'entrée de la table de traduction
 - Bits 19:0 de l'adresse virtuelle
- Sans oublier les **vérifications de permissions** d'accès
Bits 8:5 du domaine dans le DAC, et vérification bits 11:10 (AP)

Référence de page (1)

- Descripteur de tables des pages au premier niveau
 - Bits 31:10 adresse de base de la table des pages de niveau 2
 - Bits 8:5 spécifie l'un des 16 domaines possibles
- Bits de poids faible du descripteur niveau 2 pour type de page
 - 00 pour signaler une faute et 11 est réservé
 - Type 01 définit une large page et 10 définit une petite page

31	10	8	5	4	1	0
Adresse de base table pages		Domain	1		0	1

Référence de page (2)

- Descripteur de tables des pages au second niveau
 - Bits 31:16 ou bits 31:12 adresse physique cadres en mémoire
 - Bits 11:4 permissions d'accès pour sous-pages
 - Bits 3 et 2 pour indiquer si données cachable ou bufferable
- Deux tailles de pages possibles avec quatre sous-pages

Grande page occupe 64 Ko et petite page occupe 4 Ko

31	16 15	12 11 10	9 8	7	6	5	4	3	2	1	0
			ap3	ap2	ap1	ap0	C	B	0	1	
31	12 11 10	9 8	7	6	5	4	3	2	1	0	
		ap3	ap2	ap1	ap0	C	B	1	0		

Traduction référence de petite page

- **Adresse virtuelle** (VA) en trois parties
 - Bits 31:20 représentent indice dans la table niveau 1 (12 bits)
 - Bits 19:12 représentent indice dans la table niveau 2 (8 bits)
 - Bits 11:0 représentent offset dans la page (12 bits)
- **Adresse physique** (PA) construite en concaténant
 - Bits 31:12 de l'entrée de la table niveau 2
 - Bits 11:0 de l'adresse virtuelle
- Sans oublier les **vérifications de permissions** d'accès

Appartenance au bon DAC, puis vérification des AP au niveau 2

Exemple de gestion mémoire (1)

- Création de **sections de 1 Mo** pour mapper VA à PA

Exemple avec 256 Mo de RAM et 2 Mo d'espace pour E/S

- Création d'un **mapping identité** entre les blocs

256 adresses basses en VA mappées sur PA directement

- Programme constitué de **deux composants**

- Fichier en langage d'assemblage pour activer le MMU
- Fichier en C pour créer table des pages niveau 1

Exemple de gestion mémoire (2)

- Configuration et activation du MMU dans gestionnaire reset

Créer la table de traduction, la référer et activer MMU

```
1 // (m1): construction table de traduction des pages
2 BL mkptable
3
4 // (m2): fixer registre TTB à 0x4000
5 MOV R0, #0x4000
6 MCR p15, 0, R0, c2, c0, 0
7 MCR p15, 0, R0, c8, c7, 0
8
9 // (m3): domain0 en mode client
10 MOV R0, #1
11 MCR p15, 0, R0, c3, c0, 0
12
13 // (m4): activer le MMU
14 MRC p15, 0, R0, c1, c0, 0      // R0 = c1
15 ORR R0, R0, #0x00000001
16 MCR p15, 0, R0, c1, c0, 0      // c1 = R0
17 NOP
18 NOP
19 NOP
```

Exemple de gestion mémoire (3)

■ Création de la **table de traduction** des pages

Créer les 4096 entrées et remplir 258 entrées pour le mapping

```
1 int mkptable()
2 {
3     int i, pentry, *ptable;
4
5     printf("1. Build level-1 pgtable at 16 Ko\n");
6     ptable = (int*) 0x4000;
7     for (i = 0; i < 4096; i++)
8         ptable[i] = 0;
9
10    printf("2. Fill 258 entries to IP map 258 Mo VA to PA\n");
11    pentry = 0x412;      // AP=01, domain=0000, CB=00, type=02
12    for (i = 0; i < 258; i++) {
13        ptable[i] = pentry;
14        pentry += 0x100000;
15    }
16
17    printf("3. Finished building level-1 page table\n");
18    printf("4. Return to set TTB, domain and enable MMU\n");
19 }
```

Crédits

- <https://www.flickr.com/photos/wolfgangfoto/4613756745>
- <https://www.flickr.com/photos/anniebrightstar/15248861796>