GIF-3002 SMI et Architecture du microprocesseur

Ce cours discute de l'impact du design du microprocesseur sur le système entier. Il présente d'abord l'architecture du cœur ARM Cortex M3. Ensuite, le cours discute des impacts de l'architecture du microprocesseur sur le code compilé, la taille du code et la vitesse d'exécution des instructions. Enfin, le cours présente le support du microprocesseur au système d'exploitation.

1 Architecture du cœur ARM Cortex M3

Le cœur ARM Cortex M3 sera présenté en classe à partir des éléments suivants :

Le site web pour ARM :

http://www.arm.com/products/processors/cortex-m/cortex-m3.php

La datasheet du cœur ARM fournie avec le kit de développement :

CortexM3_TRM_r2p0.pdf

2 Impact de l'architecture du microprocesseur sur l'exécution d'instructions

2.1 Vitesse d'exécution des instructions

La première tâche d'un microprocesseur est d'exécuter des instructions et l'architecture du microprocesseur à un impact direct sur la vitesse d'exécution des instructions. Voici quelques informations reliant l'architecture du microprocesseur à la vitesse d'exécution des instructions :

- Un microprocesseur avec un pipeline d'instruction exécutera habituellement presque une instruction par coup d'horloge.
    - Un pipeline d'instruction ajoute de la complexité au niveau des branchements, des dépendances de données et des ressources du microprocesseur. Cela ajoute aussi de la complexité lors de la gestion des interruptions.
    - Le compilateur gère plusieurs aspects du pipeline d'instruction pour vous :
        - Il peut optimiser la séquence d'instruction pour augmenter le parallélisme entre les instructions (ILP)
        - Lorsqu'une instruction utilise le PC, le compilateur calcule ajoute un offset pour tenir compte de la valeur de PC lorsque l'instruction sera exécutée plutôt que lors de la lecture de l'instruction.
- L'architecture Harvard est nécessaire pour lire des instructions tout en manipulant des données en mémoires.
    - Il y a souvent trois bus sortant d'un microprocesseur ayant une architecture Harvard : un bus pour lire les instructions, un bus pour lire les constantes (les données dans la mémoire FLASH) et un bus pour accéder à la mémoire/périphériques.

- La vitesse des mémoires contenant les programmes (FLASH ou bande magnétique) limite souvent la vitesse de lecture des instructions. Un microprocesseur avec une cache pourra accéder à ses instructions plus rapidement. Par ailleurs, il est presque toujours possible d'exécuter des programmes à partir de la RAM du microcontrôleur : elle est plus rapide que la FLASH mais le pipeline d'instruction sera moins performant (si les instructions sont dans la RAM, on ne peut pas lire une instruction et modifier une données en mémoire!).
    - Lorsqu'une cache est utilisée, le programmeur/compilateur doit favoriser la localité.
    - Lorsqu'une cache est utilisée, le programmeur doit tenir compte de temps d'accès variable aux données ou instructions dans la cache.
- Il faudra exécuter plusieurs instructions pour faire des opérations arithmétiques sur des fractions lorsque le microprocesseur supporte des entiers seulement.
    - Lorsque vous ajouter un float ou un double dans un système où le microprocesseur traite des entiers seulement, les librairies de manipulation des fractions sont souvent automatiquement inclues dans le projet par l'environnement de développement.
- Plusieurs microprocesseurs ont des instructions permettant de mettre ou lire des valeurs sur la pile. Ces instructions, souvent plus longues à exécuter que des instructions normales, sont plus rapides qu'une séquence de POP ou une séquence de PUSH, mais elles demandent un traitement spécial par le microprocesseur.
    - Que se passe-t-il dans le ARM Cortex M3 si une interruption survient pendant un load multiple?
- Plusieurs microprocesseurs supportent une instruction qui multiplie et additionne des registres (instruction MAC --- Multiply and ACcumulate). Ces instructions sont nécessaires pour des opérations efficaces en traitement de signal.
    - Comment fait-on une FFT (Fast Fourier Transform)!?!
- Plusieurs microprocesseurs ont des instructions dont l'exécution est conditionnelle à la valeur d'un registre afin de réduire le nombre et l'impact des branchements sur le pipeline d'instruction.
- Dans plusieurs applications, un coprocesseur est nécessaire afin de réaliser toutes les tâches de l'application dans les temps prescrits. Dans beaucoup de cas, le contrôleur de DMA sert de coprocesseur et gère beaucoup de transferts de données.
    - Dans les systèmes avec DMA, il y a souvent plusieurs bus accédant aux périphériques.
    - Il faut s'assurer de la cohérence des données lorsque le système possède des caches et du DMA.
- Un prédicteur simple de branchement peut réduire significativement la pénalité moyenne reliée aux branchements. De nos jours, mêmes certains microprocesseurs basse gamme inclus une unité de prédiction des branchements.

2.2 Taille des instructions et des programmes

La plupart des microprocesseurs modernes ont une architecture RISC plutôt que CISC. Les programmes sont plus gros avec plusieurs instructions courtes qu'avec quelques

instructions longues. Voici quelques éléments qui feront augmenter la taille des programmes ayant une architecture RISC :

- Il faudra plus d'instructions pour exécuter une tâche complexe.
- Si l'instruction requise pour exécuter une tâche simple n'est pas disponible, il faudra plusieurs instructions pour exécuter la tâche.
- Si les paramètres de l'instruction ne peuvent entrer dans l'instruction, il faudra remplacer l'instruction par une combinaison d'instruction
    - Exemple 1: vous voulez faire un saut de 0x10000000, mais l'instruction JMP permet de faire un saut sur 20 bits seulement.
    - Exemple 2 : vous voulez mettre l'adresse d'une variable (adresse sur 32 bits), dans un registre, mais l'instruction MOV Reg, #Adresse de Variable n'accepte que des constantes de 24 bits seulement.

Les instructions spéciales d'un microprocesseur permettent souvent de réduire le nombre d'instructions nécessaires à l'exécution d'une tâche fréquente.

2.3 Consommation d'énergie

Un aspect de plus en plus important des systèmes microprocesseurs est la consommation d'énergie : de plus en plus de SMI sont alimentés par des batteries (téléphones cellulaires, IPAD, notebook, etc.) avec une quantité d'énergie limitée. Comme le microprocesseur est le cœur des SMI, sa consommation énergétique est très critique et l'architecture des microprocesseurs s'oriente de plus en plus en fonction de ce paramètre.

Voici quelques éléments de l'architecture du microprocesseur qui peuvent être modifiés afin réduire la consommation de puissance :

- Diminuer la fréquence d'horloge du microprocesseur, mais augmenter la performance de ce qui est effectué pendant un cycle d'horloge. Autrement dit, réduit la profondeur du pipeline d'instruction.
- Réduire le nombre de transistors dans le microprocesseur.
- Permettre l'alimentation du microprocesseur avec un voltage plus bas.
- Avoir des modes d'opération qui économisent l'énergie très efficaces :
    - Très peu d'énergie consommée quand le microprocesseur est en sommeil
    - Une transition rapide entre les modes de sommeil et l'état normal ou vice versa
    - La possibilité de désactiver les parties du microprocesseur qui sont inutilisée.
- Optimiser le jeu d'instruction et son exécution pour obtenir une meilleure performance par joule consommé par le microprocesseur.

3 Support au système d'exploitation

Bien qu'il soit possible de faire un système d'exploitation simple sans support matériel, certaines tâches des systèmes d'exploitation plus complexes requièrent des fonctionnalités additionnelles implémentées avec des portes logiques dans le microprocesseur. Ce chapitre présente diverses composantes *matérielles* intégrées dans le microprocesseur afin de permettre au système d'exploitation d'opérer.

3.1 Interruptions et OS

3.1.1 Interruption de Timer

Certains microprocesseurs, à l'instar du ARM Cortex M3, incluent une minuterie (timer) dédiée à l'interruption du système d'exploitation. Ce timer est périodique par défaut, et facile à configurer. De plus, la latence de ce timer spécialisé est petite.

> *Pour les cœurs ARM Cortex M3, il s'agit du Systick timer. Contrairement aux autres timer, celui-ci est intégré dans le cœur. Il ne s'agit pas d'un périphérique externe (au cœur) ajouté par le manufacturier intégrant le ARM Cortex M3.*

3.1.2 Latence des interruptions

Dans un système d'exploitation préemptif , la latence des interruptions est critique : du temps de microprocesseur est gaspillé à chaque entrée et sortie dans l'interruption du système d'exploitation.

La section 4.10.4 du cours 11 discute de latence des interruptions.

La section 4.5 du cours 3 discute du matériel présent dans le ARM Cortex M3 afin de réduire la latence des interruptions.

3.1.3 Interruption logicielle et appel du superviseur (SVC)

Un appel du superviseur (supervisor call --- SVC) est une interruption logicielle. Il s'agit d'une instruction spéciale supportée par le microprocesseur qui permet de déclencher une interruption.

Les SVCs sont utiles pour appeler des fonctions du système d'exploitation comme l'admission/création de tâche, l'accès à un périphérique, l'ouverture d'un fichier ou autre.

Habituellement, l'instruction SVC a pour paramètre un numéro d'interruption, celle qui doit être exécutée. Des registres prédéterminés servent également souvent de paramètres à la routine d'interruption traitant le SVC. Par exemple, si une tâche veut lancer une autre tâche, il faudra qu'elle mettre l'adresse de la nouvelle tâche dans un registre avant de lancer le SVC.

Utiliser un SVC plutôt qu'un appel de fonction à plusieurs avantages : le mode d'exécution de la fonction change ainsi que sa priorité, l'abstraction du matériel par le système d'exploitation est plus facile, le système d'exploitation peut mieux gérer la séquence d'exécution de ses routines, et plus…

> *Quand le système d'exploitation charge une application en mémoire et lui alloue du temps de microprocesseur, le programmeur de l'application ne connaît pas les adresses des fonctions du système d'exploitation. Il ne connaît pas davantage le matériel exact sur lequel sera exécuté l'application, ni les autres applications qui rouleront en même temps que son application. Utiliser des SVCs (avec d'autres facettes du système d'exploitation!) permet de contourner en partie ou en totalité ces difficultés.*

3.2 Memory Management Unit

Plusieurs opérations reliées aux accès à la mémoire ne peuvent être effectuées par le système d'exploitation sans support du microprocesseur parce que l'exécution serait trop lente autrement, voire même impossible. Par exemple, il faut du support matériel dans le microprocesseur pour protéger la mémoire (restreindre l'accès à la mémoire pour chaque application), pour effectuer des translations d'adresse efficacement, pour manipuler des données/instructions dans des caches ou pour gérer l'accès aux bus de données/instructions.

3.2.1 Protection de la mémoire

Le MMU contient souvent une unité de protection de la mémoire. Cette unité restreint l'accès à certaines régions de la mémoire définies par les registres du MMU.

Le MMU permet à un système d'exploitation d'empêcher les applications gérées par celui-ci de corrompre la mémoire réservée pour les autres applications ou pour le système d'exploitation lui-même. Le MMU permet aussi de contrôler l'accès à des mémoires externes selon leurs caractéristiques (par exemple, il permettra de lire uniquement une mémoire FLASH externe). De plus, le MMU peut aider à empêcher des erreurs de cohérences de données.

Un MMU découpe les adresses en régions et chaque région a des attributs. Par exemple, dans le ARM Cortex M3, une région peut être en lecture seule, en écriture seule, accessible en mode superviseur seulement (à partir d'une interruption), non cacheable (cette région de la mémoire ne peut être mise en cache pour assurer la cohérence des données) ou non partageable (quand la plage d'adresse désigne un périphérique).

3.2.2 Translation d'adresse

Le MMU gère la translation d'adresse avec du matériel spécialisé.

Ce sujet est abordé dans le cours d'Architecture des microprocesseurs.

3.2.3 Gestion de l'accès aux caches

Le MMU gère les accès aux caches.

Ce sujet est abordé dans le cours d'Architecture des microprocesseurs.

3.3 Banques de Registres

Plusieurs microprocesseurs ont des banques de registres afin de faciliter le changement de contexte d'un système d'exploitation.

Chaque banque de registre est une copie matérielle des registres. Une seule banque est accessible à la fois et le microprocesseur change de banque en fonction d'instructions spéciales ou de son mode d'opération.

Le système d'exploitation peut utiliser des banques de registres différentes afin de changer de contexte plutôt que de sauvegarder le contexte sur la pile. Habituellement, une banque de registre est utilisée pour toutes les applications et une autre banque est utilisée par le système d'exploitation lui-même.