Matériel et architecture

SYE

2 - Materiel et architecture

Résumé du document

Definition

Table des matières

1. Architecture Matérielle 2
1.1. Composants du Système 2
1.2. Communication CPU et Périphériques 2
1.3. Exemple de code simplifié 2
1.4. Gestion de l'Adresse Mémoire 2
2. Modes d'Exécution du CPU 3
2.1. Instructions Privilégiées 3
2.2. 2.2 Modes Utilisateur et Noyau 3
2.3. Exemple de passage entre modes 3
3. Notions d'Interruption 4
3.1. Interruption Matérielle et Logicielle 4
3.1.1. Strucutre d'une interruption 4
3.2. Routine de Service d'Interruption (ISR) 4
3.3. Exemple d'ISR 4
4. Architectures des Systèmes d'Exploitation 5
4.1. Architecture Monolithique 5
4.2. Architecture Micronoyau 5
4.3. Exemple de structure d'architecture monolithique 5
4.4. Comparaison des Architectures 5

1. Architecture Matérielle

1.1. Composants du Système

Une architecture matérielle simplifiée d'un système comporte les éléments suivants :

- **Processeur (CPU)** : Exécute les instructions.
- **Mémoire (Caches, RAM)** : Stocke temporairement les données et les instructions.
- **Bus de périphériques** : Permet la communication entre le processeur et les périphériques.
- **Contrôleurs de périphériques** : Gèrent les communications entre le CPU et les périphériques spécifiques.

1.2. Communication CPU et Périphériques

L'interface entre le CPU et les périphériques se fait via deux types de requêtes :

- **Requêtes synchrones** : Initiées par le processeur (ex. lecture/écriture en RAM).
- **Requêtes asynchrones** : Initiées par les périphériques, comme les interruptions matérielles.

1.3. Exemple de code simplifié

Pour illustrer les opérations de lecture et écriture, voici un exemple pseudocode :

```
// Lecture d'un registre d'un contrôleur de périphérique
uint32_t read_register(uint32_t *reg) {
    return *reg;
}

// Écriture dans un registre
void write_register(uint32_t *reg, uint32_t value) {
    *reg = value;
}
```

1.4. Gestion de l'Adresse Mémoire

Le **Memory Management Unit (MMU)** est responsable de la traduction des adresses virtuelles en adresses physiques, permettant la virtualisation de la mémoire.

2. Modes d'Exécution du CPU

2.1. Instructions Privilégiées

Les processeurs modernes utilisent deux types d'instructions :

- **Instructions privilégiées** : Réservées au noyau pour des raisons de sécurité (ex. accès I/O).
- **Instructions non-privilégiées** : Exécutables par les applications en mode utilisateur.

2.2. 2.2 Modes Utilisateur et Noyau

Le CPU possède deux modes d'exécution :

- **Mode utilisateur** : Limité aux instructions non-privilégiées.
- **Mode noyau** : Accède aux instructions et aux régions mémoire protégées.

2.3. Exemple de passage entre modes

Le passage du mode utilisateur au mode noyau s'effectue via une interruption :

```
; Passage en mode noyau
INT 0x80 ; Appel système, qui interrompt le CPU et déclenche une ISR
```

3. Notions d'Interruption

3.1. Interruption Matérielle et Logicielle

Les interruptions permettent aux périphériques d'interrompre le CPU pour traiter des événements externes.

- Interruption matérielle : Vient d'un périphérique externe (clavier, disque, etc.) pour signaler un événement au processeur. Elle est **asynchrone** et nécessite une ligne d'interruption dédiée.
- Interruption logicielle : Déclenchée par un programme, souvent pour appeler une fonction système. Elle est synchronisée avec l'exécution du programme et ne dépend pas d'un signal externe.

3.1.1. Strucutre d'une interruption

1. interruption du programme en cours d'exécution
2. passage en mode kernel
3. execution de l'ISR (routine d'interruption)
4. passage en mode user
5. reprise du programme interrompu

3.2. Routine de Service d'Interruption (ISR)

Chaque interruption est associée à une ISR (Interrupt Service Routine) qui gère le traitement spécifique de l'interruption.

3.3. Exemple d'ISR

Voici un exemple simplifié d'une ISR en C :

```c
void interrupt_handler() {
    // Code de gestion de l'interruption
    acknowledge_interrupt(); // Acquitter l'interruption
    // Suite du traitement
}
```

4. Architectures des Systèmes d'Exploitation

4.1. Architecture Monolithique

Dans une architecture monolithique, tous les composants de l'OS (gestion mémoire, pilotes de périphériques, etc.) sont situés dans l'espace noyau. Cela offre :

- **Avantage** : Haute performance, car les composants communiquent directement.
- **Inconvénient** : Moins de sécurité, car une défaillance peut affecter tout le noyau.

4.2. Architecture Micronoyau

Dans une architecture de micronoyau, seules les fonctions de base sont dans l'espace noyau, tandis que les autres composants sont dans l'espace utilisateur.

- **Avantage** : Améliore la sécurité et l'extensibilité.
- **Inconvénient** : Performances réduites en raison de la communication inter-processus (IPC).

4.3. Exemple de structure d'architecture monolithique

```
Kernel Space:
- Process Scheduler
- Memory Manager
- Device Drivers
User Space:
- Applications (Ex: Serveur Web, Éditeur de texte)
```

4.4. Comparaison des Architectures

Type d'Architecture	Avantages	Inconvénients
Monolithique	Haute performance, Simplicité	Faible sécurité
Micronoyau	Sécurité, Extensibilité	Performance moindre