CHAPITRE III : GESTION DES E/S PHYSIQUES

1 Introduction

Au cours de son exécution, un programme interagit avec l'environnement extérieur. Cette interaction permet à l'utilisateur, d'une part, d'alimenter le programme avec les données à traiter et d'autre part, de récupérer le résultat du traitement. Pour ce faire, des **organes d'entrées/sorties** (appelés aussi **périphériques**) sont utilisés comme interface entre l'utilisateur et le système.

On appelle **Entrée/Sortie** (**Input/Output**) toute opération de transfert d'informations (données et programmes) entre l'ordinateur (processeur et mémoire centrale) et les organes externes (périphériques de stockage et de communication).

2 Matériel

2.1 Les périphériques

Un périphérique (**Device**) est un appareil qui interagit avec l'UC et la mémoire. Certains périphériques sont branchés à l'intérieur de l'ordinateur (disques durs, …etc.) alors que d'autres sont branchés sur des interfaces externes de l'ordinateur (clavier, écrans, souris, imprimantes, …etc.).

Il existe deux grandes catégories de périphériques, les **périphériques blocs** (**Block Devices**) et les **périphériques caractères** (**Character Devices**).

- **Périphériques caractères :** Ils envoient ou reçoivent les données octets par octets. Parmi les périphériques caractères, on peut citer : le clavier, la souris, les imprimantes, les terminaux, …etc. Les données sont transmises les unes derrière les autres, on parle alors d'**accès séquentiel** (**Sequential Access**).
- **Périphériques blocs :** Ils acceptent les données par blocs de taille fixe, chaque bloc ayant une adresse propre. Parmi les périphériques blocs, on peut citer : les disques, la carte vidéo, …etc. Le grand avantage par rapport aux périphériques caractères est qu'il est possible d'aller lire ou écrire un bloc à tout moment, on parle alors d'**accès aléatoire** (**Random Access**)**.**

2.2 Les contrôleurs

Un contrôleur (**Controller**) est une unité spéciale, appelée aussi **module d'E/S** (**I/O module**) ou **coupleur**, qui sert d'**interface** entre le périphérique et l'UC. Par exemple, le module d'E/S servant d'interface entre l'UC et un disque dur sera appelé contrôleur de disque.

Les contrôleurs d'E/S ont plusieurs fonctions. En voici les principales:

- Lire ou écrire des données du périphérique.
- Lire ou écrire des données de l'UC/Mémoire. Cela implique du décodage d'adresses, de données et de lignes de contrôle. Certains modules d'E/S doivent générer des interruptions ou accéder directement à la mémoire.
- Contrôler le périphérique et lui faire exécuter des séquences de tâches.
- Tester le périphérique et détecter des erreurs.

- Mettre certaines données du périphérique ou de l'UC en mémoire tampon afin d'ajuster les vitesses de communication.

Un contrôleur dispose, pour chaque périphérique qu'il gère, de trois (03) types de registres :

- **Registres de données** (**Data Registers**) : destinés à contenir les informations échangées avec le périphérique. Ils peuvent être lus (entrée) ou écrits (sortie).
- **Registre d'état** (**State Register**) : qui permet de décrire l'état courant du coupleur (libre, en cours de transfert, erreur détectée,...).
- **Registre de contrôle** (**Control Register**) : qui sert à préciser au coupleur ce qu'il doit faire, et dans quelles conditions (vitesse, format des échanges,...).

Le contrôleur ou coupleur a la responsabilité de déplacer les données entre le(s) unité(s) périphérique(s) qu'il contrôle et sa mémoire tampon ou buffer. La taille de ce buffer varie d'un contrôleur à un autre, selon le périphérique contrôlé et son unité de transfert.

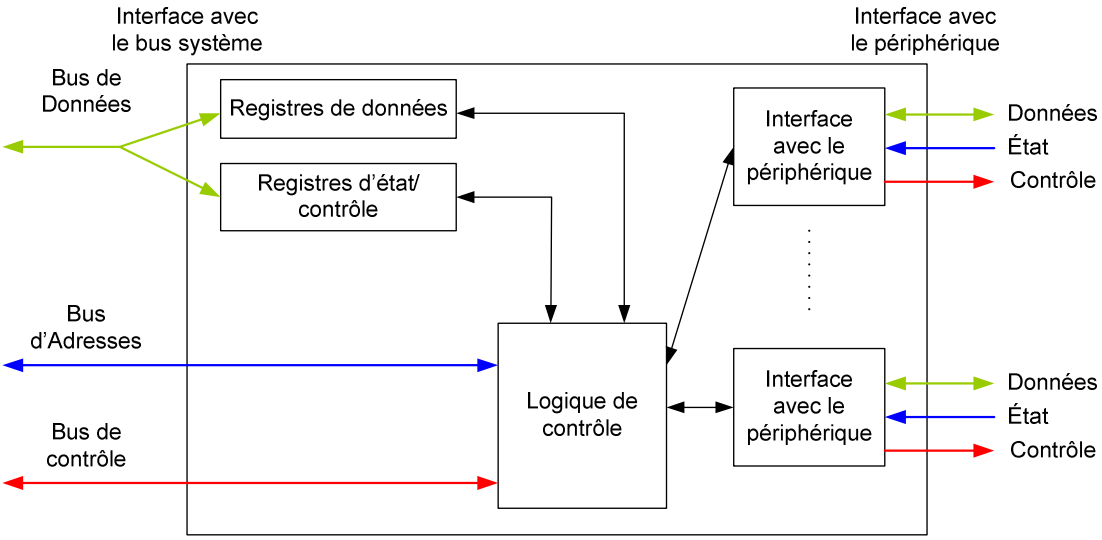

Figure 3.1 : Principaux composants d'un coupleur

Le coupleur possède aussi une logique de contrôle pour décoder l'adresse et les lignes de contrôle (ou pour faire du DMA), et une ou plusieurs interfaces avec un ou plusieurs périphériques (Voir Figure 3.1).

2.3 Les canaux

Sur les gros ordinateurs, des canaux d'E/S (**I/O Channels**) allègent le travail du processeur principal pour sa communication avec les contrôleurs (contrôle et synchronisation). Un canal d'E/S est un processeur spécialisé qui gère un ou plusieurs périphériques.

2.4 Les bus

Les contrôleurs d'E/S sont connectés sur des bus, reliés à d'autres bus par des contrôleurs de bus (souvent appelés interfaces ou ponts). Le processeur et la mémoire sont eux-mêmes sur des bus.

Chaque bus a ses propres caractéristiques. Les bus peuvent être fort différents. Néanmoins, tous les bus ont une largeur comprenant un nombre de lignes de données et d'adresses, une

vitesse de communication, un type de connecteur et un protocole qui décrit la façon dont sont échangées les données sur le bus.

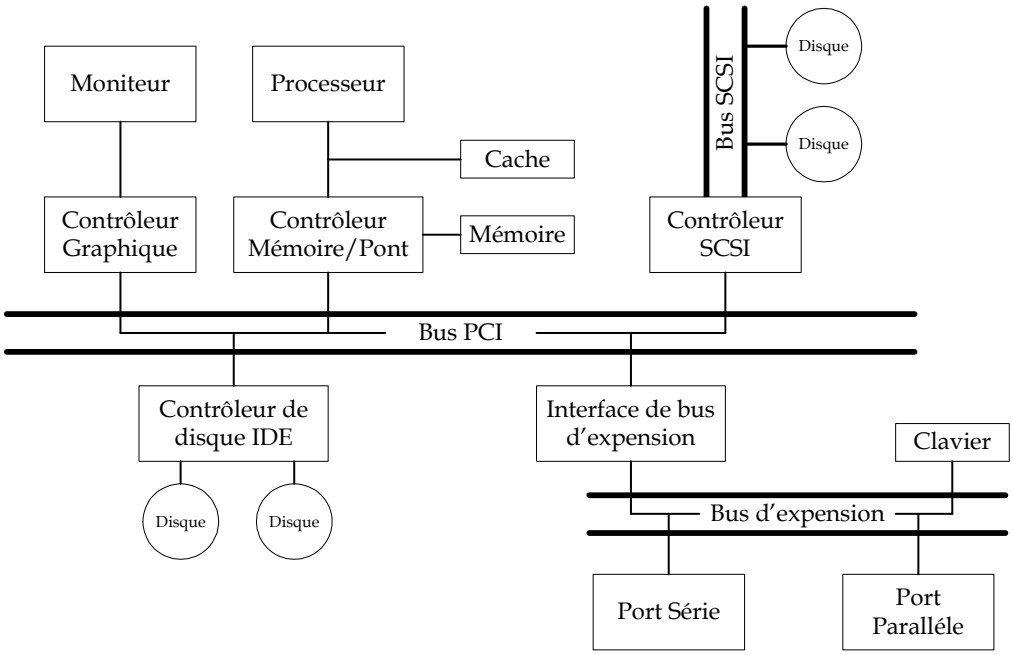

Figure 3.2 : Architecture typique des Bus d'un PC

3 Projection des E/S

La communication entre le processeur et le coupleur se fait par l'intermédiaire des **registres du coupleur**. La **désignation** de ces registres fait appel à l'une des deux techniques suivantes :

1. **Mappage sur les ports** (**Port-Mapped I/O, PMIO**) : Dans le cas où les périphériques et la mémoire centrale ont chacun leur propre espace d'adressage. L'accès s'effectue via des **instructions spécialisées** (**IN** et **OUT** en assembleur) qui permettent d'accéder au périphérique.
2. **Mappage en mémoire** (**Memory-Mapped I/O, MMIO**) : Dans le cas où les périphériques et la mémoire centrale partagent le même espace d'adressage. Les registres peuvent alors être lus ou modifiés par des instructions ordinaires. Dans ce cas, les E/S sont dites **couplées** ou **mappées en mémoire**.

4 Modes de pilotage d'une E/S physique

Le pilotage d'une unité périphérique par le processeur central nécessite une synchronisation entre les actions du processeur (où s'exécute le pilote) et l'unité périphérique pilotée.

Cette synchronisation est nécessaire au processeur (pilote) pour connaître si le périphérique est prêt, si l'opération d'E/S est terminée, …etc.

4.1 E/S physique directe

C'est une E/S physique **contrôlée par le processeur** central d'où l'appellation directe. Dans ce cas, deux modes peuvent être utilisés :

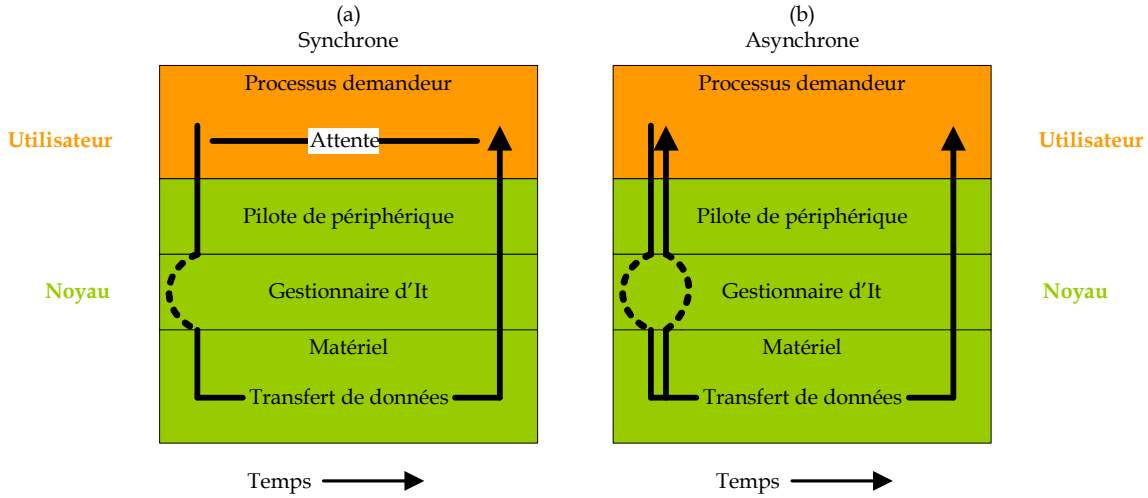

Figure 3.3 : Modes de pilotage des E/S directes

- **Mode synchrone**

  Dans ce mode, le processeur (pilote) est mobilisé à suivre l'opération d'E/S pendant toute la durée du transfert (**polling**). La fin du transfert est détectée par la consultation d'un indicateur spécifique au coupleur ou au périphérique.

  Pour démarrer une opération d'E/S, le processeur (pilote) charge les informations nécessaires dans les registres appropriés du contrôleur de périphérique. Celui-ci examine à son tour le contenu de ces registres pour déterminer l'action à effectuer (lecture, écriture, …etc.) et lance le transfert.

  La forme générale du pilote est la suivante :

  Pilote synchrone d'E/S

  - Initialise le transfert (sens du transfert : lecture, écriture, adresse du périphérique, …etc.).
  - Vérifie la disponibilité du périphérique.
  - Lance le transfert.
  - Reste en attente (active) jusqu'à la fin du transfert.

  Fin.

  Inconvénients

  Le CPU exécute une boucle d'**attente active** (**busy loop**), à la place de laquelle il pourrait faire des calculs pour le compte d'autres programmes ⇒ Perte de temps CPU.

- **Mode asynchrone**

  Dans ce mode de pilotage, le processeur est libéré du contrôle de fin de transfert. Une **interruption** est générée par le coupleur du périphérique avertissant ainsi le processeur (pilote) de la fin du transfert.

  Durant l'opération du transfert, le processeur peut exécuter un autre programme.

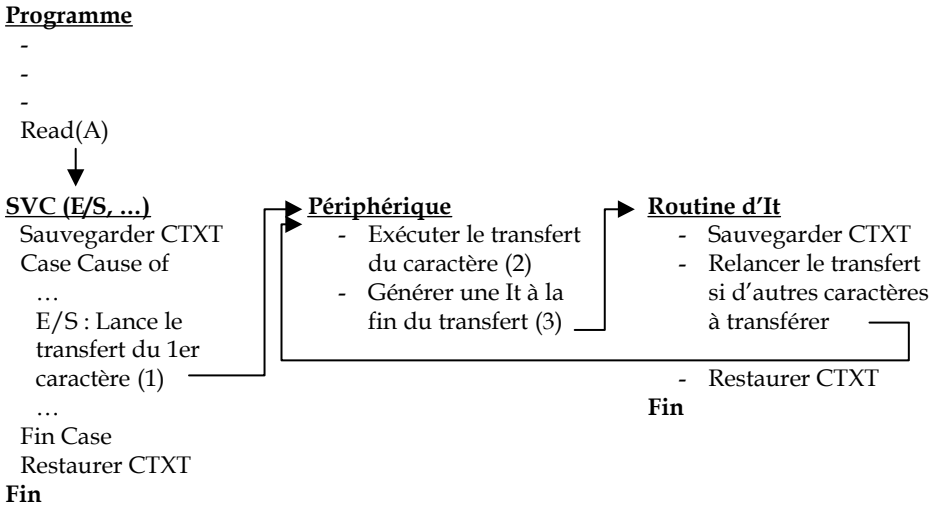

Figure 3.4 : Schéma général d'exécution d'une E/S asynchrone

Le rôle du programme usager est restreint à lancer le transfert du 1er caractère, le transfert des caractères suivants est lancé par la routine d'interruption, elle-même appelée à la fin du transfert de chaque caractère.

Dans ce mode, le pilote gère l'interface coupleur du périphérique, traite les interruptions émises, détecte et traite les cas d'erreurs.

Avantage

Utilisation plus rationnelle du CPU. En effet, durant le transfert des caractères, le processeur peut exécuter d'autres traitements (programmes).

Inconvénient

Perte de temps occasionnée par l'exécution des routines d'interruption, et des changements de contextes et programmes.

4.2 E/S physique indirecte

L'E/S physique est indirecte lorsque ce n'est plus le processeur qui se charge du suivi du déroulement de cette E/S. Ceci se fait soit en utilisant un **contrôleur d'accès direct à la mémoire** (DMA) ou en utilisant des processeurs spécialisés dits **canaux**.

A. Le contrôleur DMA (Direct Memory Access)

Pour éviter à l'unité centrale d'intervenir à chaque transfert de caractère, les ordinateurs utilisent un composant supplémentaire appelé **contrôleur à accès direct à la mémoire**, ou **DMA** (**D**irect **M**emory **A**ccess). Selon les architectures, le DMA est propre à un périphérique, au disque par exemple, ou partagé par plusieurs périphériques. Il peut être connecté entre un contrôleur de périphériques et le bus mémoire (Voir Figure 3.5), permettant ainsi aux périphériques d'accéder à la mémoire sans passer par le CPU.

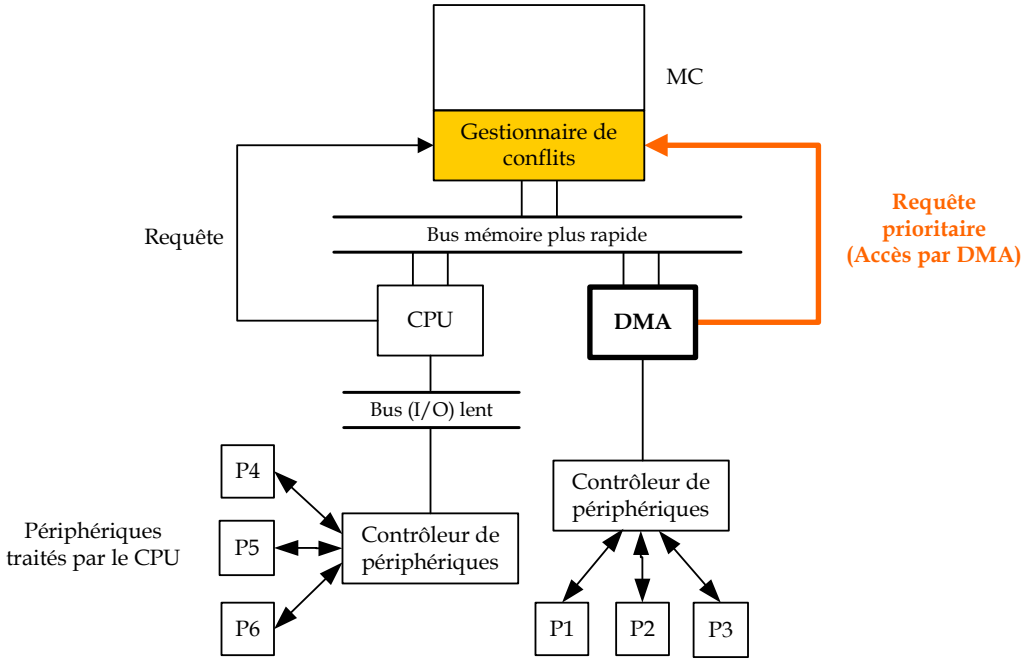

Figure 3.5 : Architecture typique avec DMA

Donc, l'utilisation du DMA décharge l'UC d'une partie importante du travail de contrôle et d'exécution des E/S. Le DMA se charge entièrement du transfert d'un bloc de données. L'UC initialise l'échange en communiquant au contrôleur DMA :

- L'identification du périphérique concerné.
- Le sens du transfert.
- L'adresse en MC du 1er mot à transférer.
- Le nombre de mots à transférer.

La programmation d'un coupleur DMA se fait, comme pour une interface par l'écriture de données dans des "registres". Pour cela, le DMA dispose (Voir Figure 3.6) d' :

- Un **registre d'adresse** (**Adress Register**) qui va contenir l'adresse où les données doivent être placées ou lues en mémoire.
- Un **registre de données** (**Data Register**).
- Un **compteur** (**Data Count**) qui compte le nombre de données échangées.
- Un dispositif de commande capable d'exécuter le transfert.

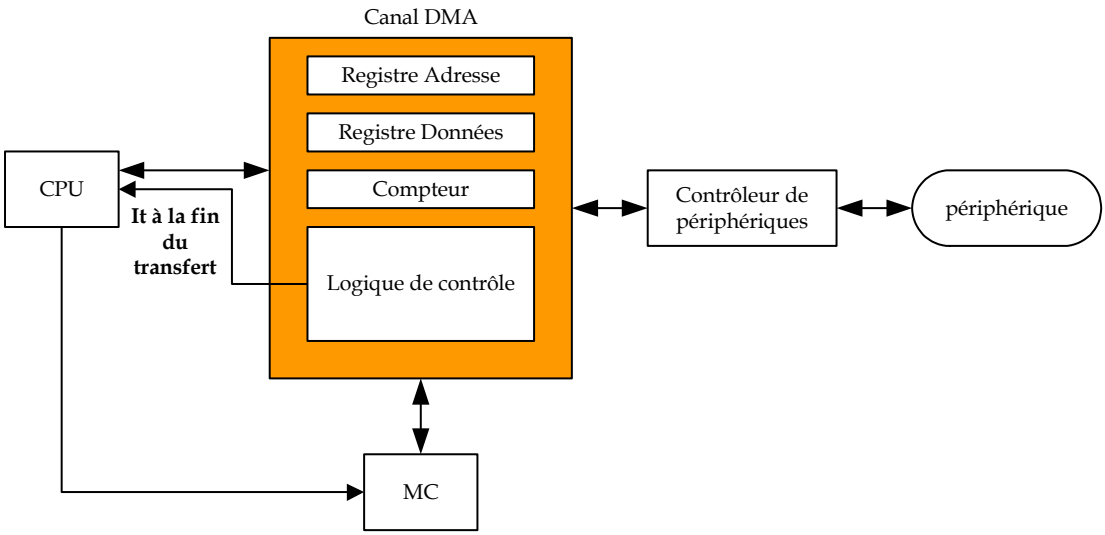

Figure 3.6 : Registres d'un contrôleur DMA

1. Processus de réalisation d'un transfert DMA

Le principe de fonctionnement est simple. L'unité centrale transmet au DMA les paramètres d'une commande à exécuter (en général, il s'agit du transfert d'une suite de caractères entre un périphérique et un tampon en mémoire) ;

Le DMA commande alors en totale autonomie l'ensemble du transfert des caractères ;

En fin de commande, le DMA envoie une interruption à l'unité centrale pour lui indiquer que le transfert est terminé et que le DMA est à nouveau disponible pour recevoir une commande.

Pratiquement, le DMA contient, pour chaque commande en cours d'exécution, une série de registres contenant l'ordre à exécuter, une adresse en mémoire centrale et un compteur d'octets. A chaque transfert de caractère, l'adresse est augmentée de 1 et le compteur diminué de 1. L'interruption de fin de transfert est envoyée au processeur lorsque le compte d'octets atteint la valeur 0.

La figure ci-dessous (Figure 3.7), résume les principales étapes d'un transfert DMA :

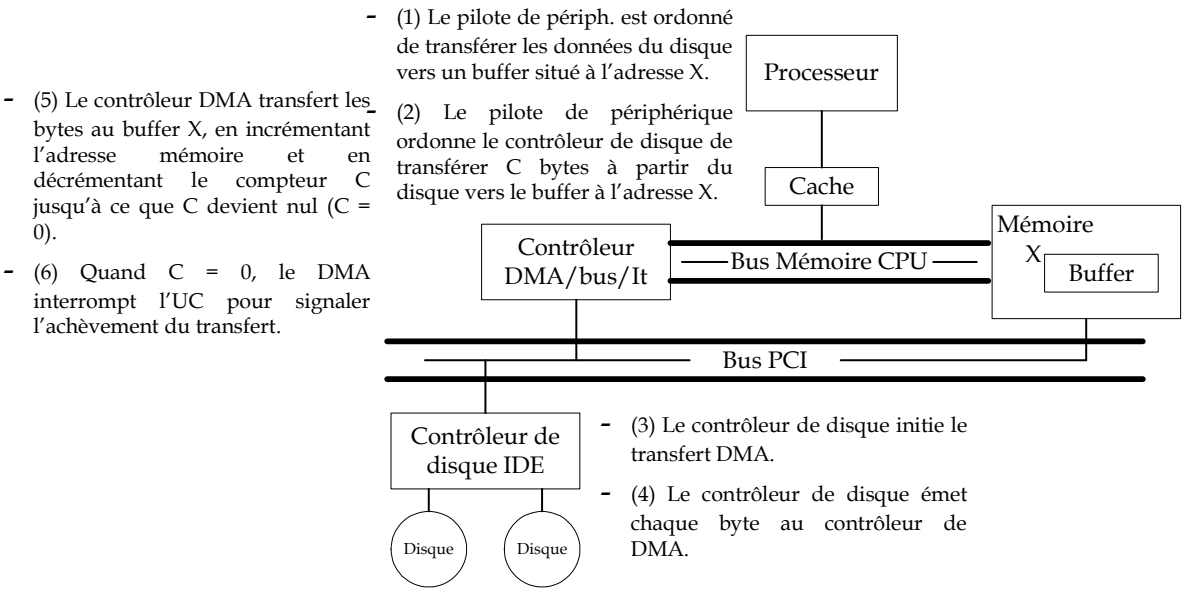

Figure 3.7 : Processus d'un transfert DMA

2. DMA et vol de cycles mémoire

Un mécanisme DMA nécessite :

- La mise en place d'un chemin de passage entre le canal et la MC, matérialisé par un bus.
- Un gestionnaire des **conflits d'accès** à la MC entre le canal et le CPU.

Pour cela, une technique de **vols de cycles** (**Cycle Stealing**) est utilisée (Voir Figure 3.8).

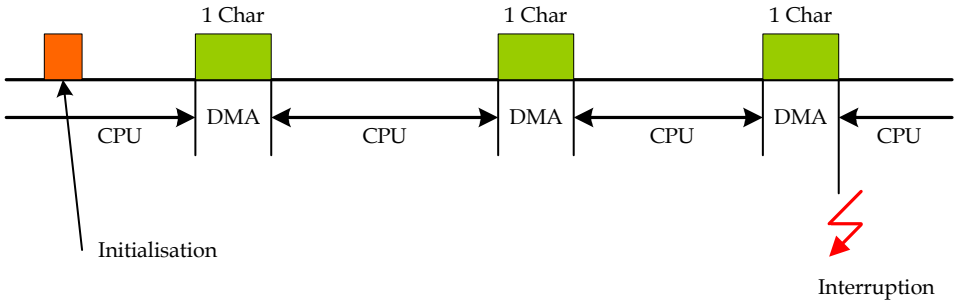

Figure 3.8 : Principe de vol de cycles

Elle consiste à freiner le fonctionnement du processeur en prenant des cycles mémoires pendant lesquels le contrôleur DMA transférera de l'information. En effet, les contrôleurs DMA sont prioritaires sur le processeur central pour l'accès à la mémoire, car ils doivent réagir rapidement à des événements externes.

5 Traitement d'E/S simultanées

Lorsqu'une requête d'E/S arrive en mode synchrone, une requête est en cours d'exécution à la fois, puisque le CPU attend la fin d'E/S.

Néanmoins, en mode asynchrone et mode DMA, l'E/S demandée est lancée. Puis, le contrôle est rendu immédiatement à un programme utilisateur, qui peut formuler de nouvelles requêtes d'E/S.

A cet effet, le S.E maintient une table contenant une entrée pour chaque périphérique d'E/S ; c'est la **table d'état des périphériques** (**Device-Status Table**).

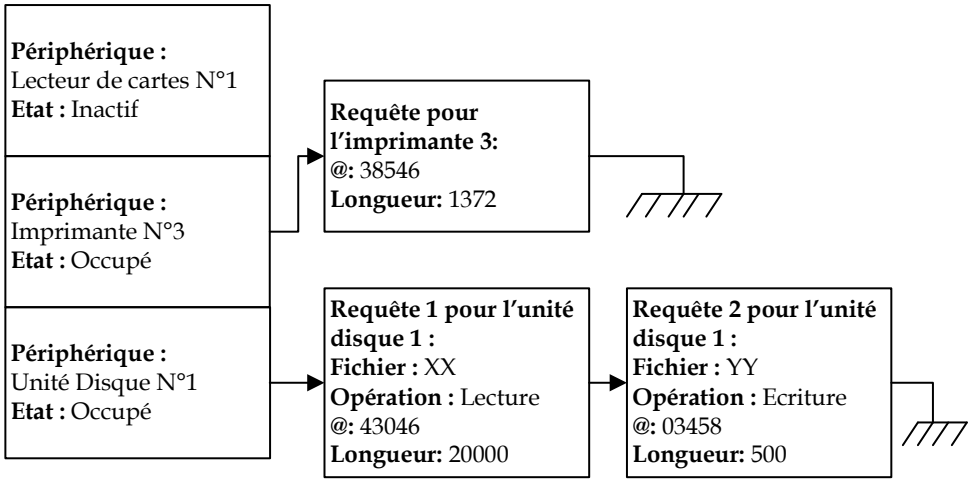

Figure 3.9 : Table d'état des périphériques

Chaque entrée de la table (Voir Figure 3.9) indique le type du périphérique, son adresse et son état (inactif ou occupé), ainsi qu'une liste des requêtes formulées d'E/S. Cette liste contient, pour chaque requête, le type d'opération, l'adresse de données et la longueur des données, …etc.

6 E/S Tamponnées

Bien que le rôle final d'une E/S soit l'échange de caractères entre un périphérique et une zone de mémoire d'un utilisateur, les logiciels d'E/S utilisent souvent une zone intermédiaire, appelée **tampon d'E/S** (**I/O Buffer**), dans la **mémoire du système**. L'utilisation des tampons est justifiée par l'**amélioration des performances** d'un périphérique.

Exemples

- Dans un échange avec un disque magnétique, l'essentiel du temps pris par l'échange vient du positionnement du bras et du délai rotationnel. Il est alors classique de ranger dans un tampon le contenu de toute une piste.
- La lecture de caractères au clavier peut se faire par anticipation ; c'est à dire que l'utilisateur peut frapper des caractères avant que le programme ait envoyé un ordre d'entrée. Les caractères frappés sont stockés dans le tampon et seront plus tard transférés dans une zone utilisateur.
- Dans un système en temps partagé, l'espace mémoire d'un utilisateur peut être vidé de la mémoire principale. C'est en particulier le cas lorsqu'un processus est bloqué en attente de la fin d'une E/S lente. Le recours à un tampon système est alors obligatoire.

7 Couches Logicielles d'E/S

Le système d'exploitation s'occupe de gérer les E/S à l'aide de quatre (04) niveaux de logiciel (Voir Figure 3.10), soit :

- **Logiciel** faisant partie de l'**espace utilisateur** (**User-Space Software**). Il représente les **procédures standards** (**programmes de bibliothèque**), appelées à partir des programmes pour formuler une requête d'E/S au superviseur et mettant en œuvre des fonctions supplémentaires (**Ex.** formatage des E/S, gestion des **spools**, …etc.).

- **Logiciel indépendant du matériel** (**Device-independent Software**). Le rôle principal de cette couche est de donner une interface uniforme (commune à toutes les E/S) au logiciel des utilisateurs. Les principales fonctionnalités offertes par cette couche sont :
  - Adressage des périphériques par leur nom,
  - Protection des périphériques,
  - Mise en mémoire tampon de données,
  - Signalisation d'erreurs (erreurs de programmation comme écrire sur un disque inexistant, erreurs qui n'ont pu être résolues par les pilotes, ...),
  - Allocation et libération des périphériques dédiés (i.e. non partageables, en gérant une file d'attente par exemple...),
  - Fourniture d'une taille de bloc indépendante du périphérique.
- Un programme d'E/S appelé **pilote** (**Driver** ou **Handler**) commandant le fonctionnement élémentaire de chaque unité périphérique. Le pilote de périphériques est la seule partie logicielle à connaître toutes les spécificités du matériel. Le handler gère directement l'interface du coupleur du périphérique, traite les interruptions émises par celui-ci, détecte et traite les cas d'erreurs.

  Il est généralement **invisible** aux utilisateurs du système. Le handler contient donc les primitives permettant de commander le périphérique. Ce driver est constitué de deux procédures, quasiment indépendantes : une **procédure traitant l'initialisation d'un transfert** et une **procédure de traitement de l'interruption** associée à une fin de transfert.
- Le **gestionnaire d'interruptions** (**Interrupt Handler**) dont le rôle est d'informer le pilote associé au périphérique de la fin de l'E/S.

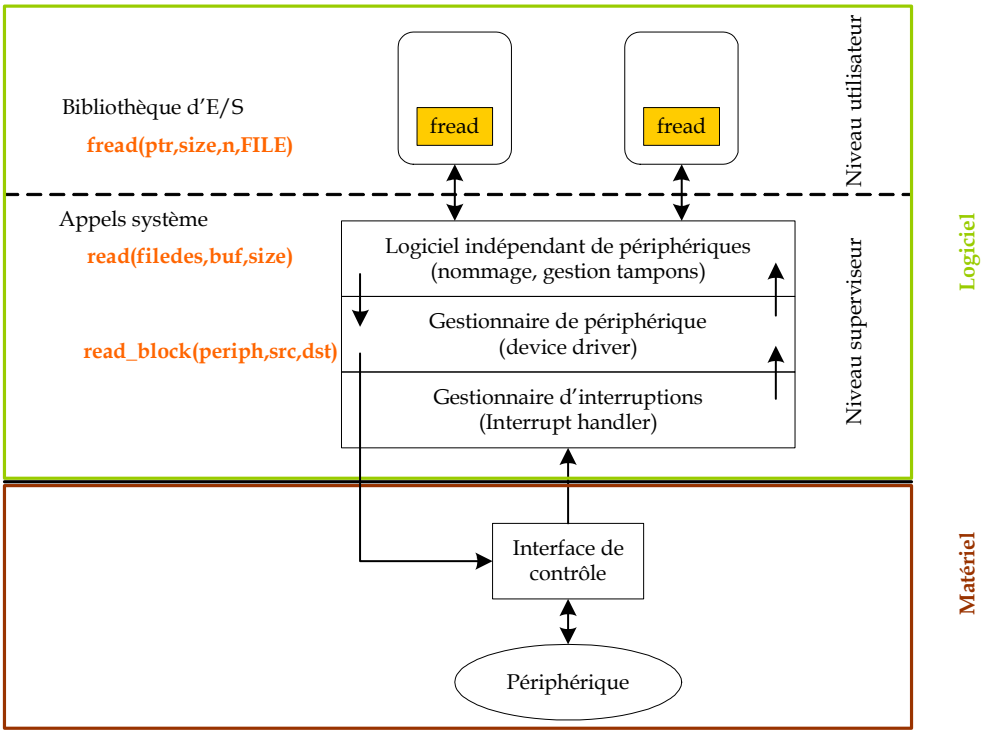

Figure 3.10 : Structure en couches d'un système d'E/S