

## CHAPITRE V : GESTION DES ENTREES/SORTIES PHYSIQUES

### 5.1 MECANISMES D'INTERRUPTION :

Le mécanisme d'interruption qui se trouve à la base du module d'entrées/sorties des SE.

#### 5.1.1 Définition et Principe :

Une interruption est un mécanisme qui permet d'interrompre l'exécution d'un processus suite à un événement extérieur ou intérieur et de passer le contrôle à une routine dite "routine d'interruption" (traitant d'interruption ou interrupt handler).

Les interruptions peuvent d'être d'origines diverses, mais on les classe généralement en trois grands types :

- **externes** (indépendantes du processus) interventions de l'opérateur, pannes, etc.
- **déroutements** erreur interne du processeur, débordement, division par zéro, défaut de page (causes qui entraîne la réalisation d'une sauvegarde sur disque de l'image mémoire) , etc.
- **appels systèmes**, comme les demandes d'entrées-sorties par exemple.

Lorsque l'interruption se produit le processeur, après la fin de l'exécution de l'instruction en cours, transfère le contrôle à la routine d'interruption associée à l'événement. La routine d'interruption fait d'abord une sauvegarde du contexte du processus interrompu avant de réaliser son traitement. A la fin de celui-ci l, le contexte du processus interrompu est restauré ce qui lui permet de continuer son exécution convenablement à l'endroit où il a été interrompu.

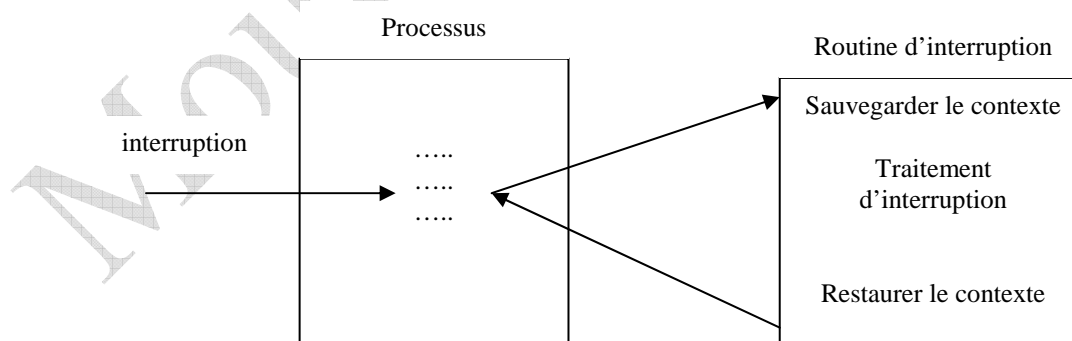


Fig. 5.1 Déroulement d'une routine d'interruption

#### 5.1.2 Vecteur d'interruptions :

Lorsque le signal d'une interruption arrive, il modifie l'état d'un indicateur (drapeau ou flag) qui est régulièrement testé par l'unité centrale. Une fois que le signal est détecté, il faut déterminer la cause de l'interruption. Pour cela on utilise un indicateur, pour les différentes causes d'interruption. On utilise

cet indicateur pour accéder à un vecteur d'interruptions qui associe à chaque type d'interruption l'adresse de la routine d'interruption correspondante. Un vecteur d'interruption a donc la structure suivante :

N° d'interruption	Adresse de la routine d'interruption
0	Adr0
1	Adr1
2	Adr2
3	Adr3
...	...
N	adrN

Fig. 5.2 Vecteur d'interruptions

### Cas de l'IBM PC :

Le vecteur d'interruption de l'IBM PC peut avoir 256 entrées correspondant chacune à une cause d'interruption. Les interruptions peuvent être de trois catégories : les déroutements, les interruptions matérielles et les interruptions logicielles.

Les **déroutements** sont au nombre de 6 et concernent :

- La division par zéro
- Le fonctionnement pas à pas (une interruption est générée à chaque instruction)
- Les problèmes pouvant apparaître lors de l'accès à la mémoire (défaut de page, par exemple)
- L'atteinte d'un point d'arrêt
- Le débordement numérique
- La non reconnaissance d'une instruction

Les **interruptions matérielles** sont au nombre de 16. Elles sont baptisées IRQ 0 à 15. Certaines sont préaffectées, mais beaucoup d'entre elles ne sont affectées à des événements périphériques que lors de la configuration de la machine. En voici quelques unes :

N° d'interruption	Adresse de la routine d'interruption
IRQ0	Horloge interne (18.2 par seconde)
IRQ1	clavier
IRQ2	
IRQ3	Interface série 2
IRQ4	Interface série 1
IRQ5	Le disque dur
IRQ6	Le lecteur de disquette
IRQ7	L'interface parallèle (imprimante)
IRQ8	Horloge temps réel

Fig. 5.3 Interruptions matérielles

Les interruptions logicielles utilisent le même mécanisme, si ce n'est qu'elles sont générées par des instructions spécifiques qui permettent d'appeler des fonctions du BIOS ou du

système d'exploitation sans en connaître les adresses d'implantation qui peuvent d'ailleurs varier d'une version à l'autre.

Lors de l'occurrence d'une interruption, ou d'un déroutement, le processeur reçoit un index qui pointe dans le vecteur d'interruptions.. Cet index est généré par le processeur dans le cas des déroutements. Il est fourni par un circuit de la carte mère appelé contrôleur d'interruption dans le cas d'une interruption matérielle et par l'instruction dans le cas d'une interruption logicielle. Au vu de cet index, le processeur accède à l'entrée correspondante de la table d'interruption.

### 5.1.3 Priorités et masquage des interruptions :

On peut envisager une hiérarchie de priorité des interruptions. Ainsi, lorsque plusieurs interruptions arrivent en même temps, c'est la plus prioritaire qui est pris en charge en premier. Cela implique aussi que la routine elle-même d'une interruption peut être interrompue par une interruption plus prioritaire (voir figure).

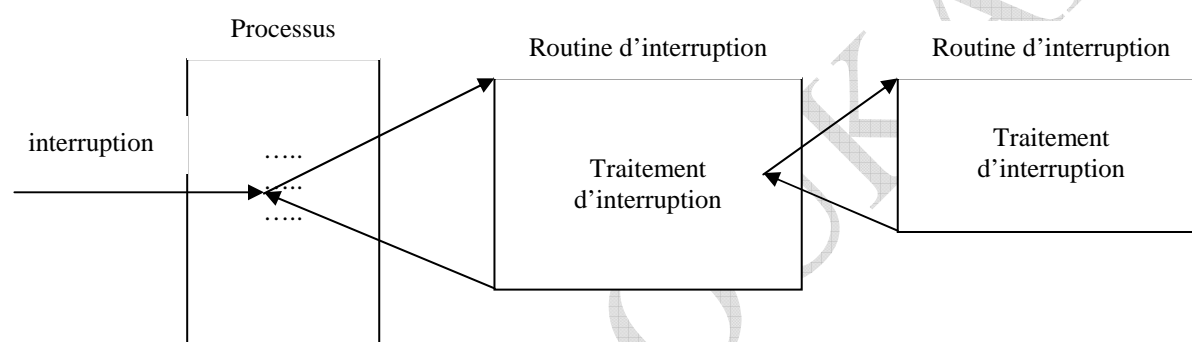


Fig. 5.4 Interruptions en cascade

Il est possible à un processus superviseur de *masquer* une interruption. Masquer une interruption revient à retarder son effet temporairement. Elle peut être *démasquée* après.

D'autre part, un niveau d'interruption peut être *désarmé*. Une interruption désarmée est une interruption dont l'effet est annulé.

## 5.2 ARCHITECTURE MATERIELLE :

### 5.2.1 Les contrôleurs :

Un périphérique d'E/S contient en fait deux parties : un appareil (clavier, écran, disque, ...) et un *contrôleur* de périphérique. Le contrôleur sert d'interface entre le périphérique et le processeur : il reçoit les requêtes du processeur et les transforme en commandes pour le périphérique, et réciproquement, il envoie les requêtes du périphérique au processeur. Il gère également les transferts entre les processeurs et le périphérique. Les principales requêtes qu'un processeur envoie à un périphérique sont « lecture » et « écriture ». Selon les périphériques ces requêtes peuvent correspondre à des commandes simples ou complexes.

On considère qu'il y a deux grandes catégories de périphériques : les périphériques par caractères et les périphériques par blocs. Dans un périphérique par blocs, on ne peut accéder à l'information que par blocs et chaque bloc possède une adresse (exemple : le disque). Dans un périphérique par caractère, on accède bien sûr à l'information caractère par caractère (exemple : le clavier) mais on ne

peut spécifier une adresse ni rechercher une information : on reçoit ou on envoie simplement un flux de caractères.

### 5.2.2 Les drivers :

Chaque contrôleur de périphérique a des commandes spécifiques. Comme un concepteur de SE ne peut inclure dans son logiciel l'ensemble des commandes de l'ensemble des périphériques, il existe différentes couches logicielles pour réaliser l'interface entre le SE et les périphériques. Du point de vue du SE, les périphériques n'ont besoin que d'être lus ou écrits, ce qui constitue l'interface de plus haut niveau. Ensuite, la façon d'accéder à un périphérique diffère suivant qu'il s'agit d'un périphérique par blocs (plusieurs blocs adressables indépendamment) ou par caractères (flux de caractères) ; il existe donc une interface de bas niveau pour ces deux types de périphériques. Enfin, la couche logicielle de plus bas niveau est spécifique au périphérique : elle traduit des commandes générales des périphériques par blocs ou par caractères en commandes spécifiques au périphérique cible. Cette dernière couche logicielle s'appelle un driver ; elle est généralement fournie par le constructeur du périphérique et est insérée dans le SE. A l'origine, le SE ne contient donc qu'une vue abstraite du système matériel.

## 5.3 LES DIFFERENTS MODES D'ENTREES/SORTIES PHYSIQUES :

Le principe élémentaire mis en œuvre pour l'échange de données entre deux constituants physiques est représenté par la figure suivante. En dehors des données proprement dites, deux liaisons supplémentaires sont nécessaires, pour permettre d'une part à l'émetteur de la donnée de signaler la présence effective de cette donnée sur les fils correspondants, et d'autre part au récepteur de signaler qu'il a lu la donnée.

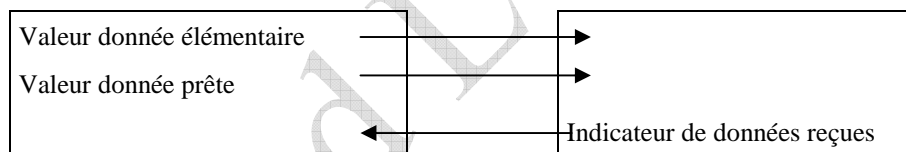


Fig. 5.6 Protocole élémentaire d'entrées-sorties.

A partir de ce schéma élémentaire plusieurs modes d'entrées-sorties ont été proposés dans les systèmes informatiques : les E/S programmées, les E/S par interruption, les E/S avec accès DMA et les E/S avec processeur spécialisé.

### 5.3.1. Les entrées-sorties programmées (avec scrutation)

La façon la plus simple d'assurer la liaison entre le bus et un périphérique, est de faire une simple adaptation des signaux évoqués ci-dessus. On parle alors d'une *interface*. Le processeur adresse directement le périphérique soit par les instructions habituelles d'accès à la mémoire centrale, l'interface jouant alors le rôle d'un (ou de plusieurs) emplacement de mémoire, soit par des instructions spécialisées qui assurent le transfert d'une donnée élémentaire avec un registre ou un emplacement mémoire. Dans tous les cas, le programmeur doit assurer le protocole élémentaire d'échanges évoqué plus haut ; c'est pourquoi on parle d'entrées-sorties programmées.

```

tantque il_y_a_des_données_à_lire faire
    tantque donnée_suivante_non_prête
    faire
    fait; { attente de la donnée}
    lire_la_donnée;

```

```

    traitement_de_la_donnée;
fait

```

Fig. 5.7 Exemple d'entrée-sortie programmée.

Critique : Avec ce mode, le débit est souvent limité à 50 Ko/s (Kilo-octets par seconde). Par ailleurs si le périphérique est lent, le processeur est monopolisé pendant toute la durée de l'échange. Dans ce cas, on ne lit que quelques octets à la fois pour éviter cette monopolisation. Cette forme d'échange était la seule possible dans les premières générations de machines.

### 5.3.2 Les entrées-sorties par interruptions

Pour éviter la monopolisation du processeur pendant toute la durée de l'E/S, une première amélioration a été apportée grâce au mécanisme d'interruption. Pour comprendre ce principe, examinons l'exemple suivant :

On veut faire une lecture sur disque. Pour cela le contrôleur lit un bloc, qui correspond à un ou plusieurs secteurs, bit par bit, jusqu'à ce que le bloc soit entièrement rangé dans son tampon. A ce moment là, le contrôleur génère une interruption. Le processeur peut ainsi lire le contenu du tampon du contrôleur et recopier les données lues en mémoire. Cette lecture peut se faire au moyen d'une boucle qui lit à chaque itération un octet ou un mot du tampon. Cette boucle exécutée par le processeur pour lire un octet après l'autre consomme, malheureusement, beaucoup de temps du processeur.

### 5.3.3 Les entrées-sorties par accès direct à la mémoire

Pour accroître le débit potentiel des entrées-sorties, et diminuer la monopolisation du processeur dont nous avons parlé ci-dessus, une autre solution a été de déporter un peu de fonctionnalité dans le dispositif qui relie le périphérique au bus, de façon à lui permettre de ranger directement les données provenant du périphérique en mémoire dans le cas d'une lecture, ou d'extraire directement ces données de la mémoire dans le cas d'une écriture. C'est ce que l'on appelle l'*accès direct à la mémoire*, ou encore le *vol de cycle*.

### 5.3.4 Les entrées-sorties par processeur spécialisé

L'autre façon de relier un périphérique avec la mémoire est d'utiliser un processeur spécialisé d'entrées-sorties, c'est-à-dire de déléguer plus d'automatisme à ce niveau. Dans le schéma précédent, le processeur principal avait en charge la préparation de tous les dispositifs, accès direct à la mémoire, contrôleur, périphérique, etc..., pour la réalisation de l'opération. L'utilisation d'un processeur spécialisé a pour but de reporter dans ce processeur la prise en charge de cette préparation, mais aussi des opérations plus complexes, telles que par exemple le contrôle et la reprise d'erreurs.

## 5.4 LES ENTREES/SORTIES SYNCHRONES/ASYNCHRONES :

Suivant le mode de transfert utilisé, le mode d'entrée-sortie peut être synchrone ou asynchrone :

- Une opération d'E/S est dite *synchrone* si le processeur doit attendre la fin de l'opération d'E/S pour continuer son traitement ; on dit aussi que dans ce cas le transfert est bloquant.
- Une opération d'E/S est dite *asynchrone* si le processeur peut lancer l'opération et accomplir d'autres tâches en parallèle. Il sera informé de la fin de l'opération d'ES par une interruption.

## 5.5 LES PILOTES DE PERIPHERIQUES : DRIVERS

Un pilote (driver) est un programme qui gère un périphérique. Chaque contrôleur possède un ou plusieurs registres de commande. Les pilotes de périphériques envoient ces commandes et vérifient leur bon cheminement. Le pilote de disque, par exemple, est la seule partie du SE qui connaît les registres d'un contrôleur de disque donné et leur utilisation. Il est le seul à connaître les secteurs, les pistes, les cylindres, les têtes, le déplacement du bras, le temps de positionnement des têtes et tous les autres mécanismes qui permettent le bon fonctionnement du disque.

D'une manière générale, un pilote de périphérique doit traiter les requêtes de plus haut niveau qui émanent du logiciel (indépendant du matériel) situé au dessus de lui.

La première étape pour un pilote consiste à traduire une requête d'E/S en termes concrets. Un pilote de disque doit, par exemple, déterminer où se trouve le bloc requis et vérifier si le moteur du disque tourne, si le bras est bien positionné, .... En résumé, il doit déterminer les opérations que le contrôleur doit exécuter ainsi que l'ordre dans lequel il faut les effectuer. Après cette étape, le pilote remplit les registres du contrôleur pour faire exécuter ces commandes.

## 5.6 LA MEMOIRE SECONDAIRE :

Les mémoires secondaires, dont les disques, ont trois avantages majeurs par rapport à la mémoire centrale :

- La capacité de stockage est beaucoup plus grande.
- Le prix de l'octet est beaucoup plus faible.
- Les informations ne sont pas perdues lorsqu'on éteint l'ordinateur.

Une caractéristique intéressante des disques est que le pilote peut effectuer des recherches sur plus d'un disque à la fois. On les qualifie de recherches simultanées (overlapped seeks). Le contrôleur peut lancer une recherche sur un disque alors qu'il attend des résultats en provenance d'un autre. De nombreux contrôleurs peuvent aussi lire ou écrire des données sur un disque alors qu'ils recherchent des données sur un ou plusieurs autres. Mis aucun ne peut lire ou écrire sur deux disques à la fois. Le contrôleur doit, lors d'une lecture ou d'une écriture, déplacer les octets toutes les microsecondes. Un transfert consomme donc pratiquement toute la puissance de calcul d'un contrôleur. Les recherches simultanées peuvent réduire de manière significative le temps d'accès moyen.

D'une manière générale, on peut dire que le temps de lecture ou d'écriture d'un bloc de disque dépend de 3 facteurs :

- *Temps de recherche* ; il s'agit du temps qu'il faut pour positionner le bras sur le bon cylindre.
- *Délai de rotation* ; c'est le temps qu'il faut pour positionner le secteur requis sur la tête.
- *Temps de transfert réel*

Le temps de recherche est le plus important pour la plupart des disques ; sa réduction améliore sensiblement les performances du système.

Si le pilote du disque n'accepte qu'une requête à la fois et exécute les requêtes séquentiellement, c'est à dire selon la méthode Premier Arrivé Premier Servi (FCFS), il n'est pas possible de réduire de manière significative le temps de recherche. Il existe néanmoins une autre stratégie quand le disque est utilisé intensivement. Le disque peut, par exemple, accepter d'autres requêtes pendant que le bras se positionne en réponse à une première requête. De nombreux pilotes utilisent une table indexée par les numéros des cylindres, où chaque entrée pointe sur le premier élément d'une liste chaînée des différentes requêtes qui concernent un cylindre.

## 5.7 ORDONNANCEMENT DES REQUÊTES DU DISQUE :

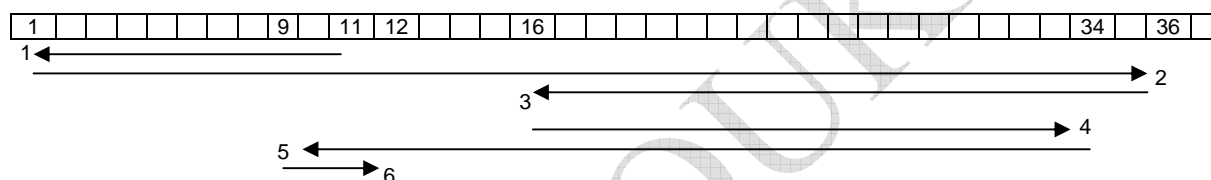
Il existe plusieurs algorithmes d'ordonnement des requêtes du disque : FCFS, SSF, SCAN , ... etc.

### 6.6.1 Algorithme FCFS

Cet algorithme exécute les requêtes en fonction de leur ordre d'arrivée. Considérons, par exemple, un disque de 40 cylindres. Une demande de lecture du cylindre 11 arrive. Pendant que la recherche est en cours, le pilote du disque reçoit respectivement de nouvelles requêtes qui concernent les cylindres 1, 36, 16, 34, 9 et 12. Elles sont placées dans la table des requêtes en attente qui contient la liste chaînée pour les différents cylindres.

A la fin de la requête courante (concernant le cylindre 11), le pilote doit choisir la requête suivante, c'est à dire 1, puis 36, puis 16, et ainsi de suite.

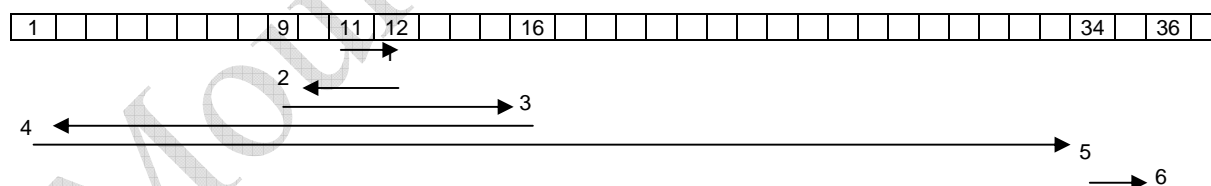
Cet algorithme requiert que le bras se déplace de 10, 35, 20, 18, 25 et 3 cylindres, soit un total de 111 cylindres. Le schéma suivant illustre le mouvement du bras du disque induit par l'application de cet algorithme (en supposant que la position actuelle est le cylindre 11).



L'inconvénient de cet algorithme est le nombre de cylindres parcourus trop important, ce qui le rend peu efficace.

### 6.6.2 Algorithme SSF (Shortest Seek First) :

Cet algorithme consiste à permettre au pilote de choisir la requête qui concerne le cylindre le plus proche de la position actuelle de manière à réduire le temps de recherche. L'ordre des cylindre est donc : 12, 9, 16, 1, 34, 36.



Le nombre de cylindres parcourus est donc :  $1+3+7+15+33+2=61$

#### Inconvénient de cet algorithme :

Si après s'être placé sur le cylindre 16, une nouvelle requête arrive pour le cylindre 8, cette dernière aura la priorité sur celle du cylindre 11. Une nouvelle requête pour le cylindre 13 sera, elle aussi, traitée avant celle du cylindre 1. Si le disque est utilisé intensément, le bras aura tendance à rester au centre du disque. Les cylindres extrêmes ne seront accédés que lorsque la moyenne des requêtes se déplacera, c'est à dire quand il n'y aura plus de requêtes qui concernent les cylindres du milieu du disque. Les requêtes qui portent sur des cylindres éloignés du milieu du disque risquent, de ce fait, de ne pas être traités rapidement.

### 6.6.3 Algorithme Scan (Ascenseur) :

La gestion d'un ascenseur dans un immeuble ressemble à celle du bras du disque. Les requêtes arrivent continuellement pour demander aléatoirement l'ascenseur aux différents étages (cylindres). Le processeur qui contrôle l'ascenseur mémorise facilement l'ordre des appels puis les traite selon la technique suivante : Il se déplace dans un certain sens tant qu'il y a des appels à traiter dans ce sens, puis ils changent de sens. Le logiciel doit mémoriser un seul bit : le sens du déplacement, vers le bas ou vers le haut. A la fin d'une requête, le pilote du disque ou de l'ascenseur vérifie ce bit. S'il est à haut, le bras (ou la cabine) se place au niveau immédiatement plus élevé, indiqué par la première requête en attente. S'il n'y a pas de requêtes en attente, le bit de sens est inversé et la recherche se fait dans l'autre sens.

Le nombre de cylindres parcourus pour la même liste de requêtes examinée précédemment est :  $1+4+18+2+27+8=60$ .

