périphériques d'E/S est souvent intimement liée à leur fonctionnement interne. Dans les trois prochaines sections, nous ferons un petit rappel des relations qui existent entre matériel d'E/S et leur programmation. Considérez-le comme une révision et un complément de l'introduction que nous avons faite à la section 1.4.

5.1.1 Les périphériques d'E/S

Les périphériques d'E/S peuvent être divisés en deux catégories du point de vue des informations manipulées : les **périphériques d'E/S par blocs** (*block devices*) et les **périphériques d'E/S de caractères** (*character devices*). Un périphérique d'E/S par blocs stocke les informations dans des blocs de taille fixe, chaque bloc possédant sa propre adresse. La taille courante des blocs s'étend de 512 à 32 768 octets. La propriété essentielle d'un périphérique d'E/S par blocs est qu'il permet d'écrire et de lire chaque bloc indépendamment des autres. Les disques durs, les CD-ROM et les clés USB représentent les périphériques d'E/S par blocs les plus courants.

La limite entre les périphériques dont les blocs sont adressables et ceux qui ne le sont pas n'est pas très bien définie. Chacun s'accorde à dire qu'un disque est un périphérique avec blocs adressables puisque, quel que soit l'emplacement du bras, il est toujours possible de se déplacer vers un autre cylindre puis d'attendre que le bloc demandé passe sous la tête de lecture/écriture. Prenons à présent l'exemple d'un lecteur de bandes destiné aux sauvegardes. Les bandes contiennent une suite de blocs. Si le lecteur de bandes reçoit la commande de lire le bloc *N*, il peut toujours rembobiner la bande et aller en avant jusqu'à trouver le bloc *N*. Cette opération est analogue à un déplacement du bras sur un disque, mais elle est beaucoup plus longue. En outre, on ne peut pas toujours réécrire un bloc qui se trouve au milieu de la bande. Même s'il est possible d'utiliser des bandes comme périphériques par blocs à accès aléatoire, c'est une déformation de leur fonction et on ne les exploite normalement pas de cette manière.

Un périphérique d'E/S de caractères représente l'autre type de périphérique d'E/S. Il reçoit ou fournit un flot de caractères, sans aucune structure de blocs. Il n'est pas adressable et ne possède aucune fonction de recherche. Les imprimantes, les interfaces réseau, les souris et la plupart des autres périphériques que l'on ne peut apparenter à des disques peuvent être considérés comme des périphériques d'E/S de caractères.

Toutefois, cette classification n'est pas parfaite : certains périphériques ne correspondent pas à ce schéma. Les horloges, par exemple, ne sont pas adressables par blocs, pas plus qu'elles ne génèrent ou n'acceptent des flots de caractères. Elles servent simplement à provoquer des interruptions à des intervalles définis. Les écrans à mémoire ne correspondent pas davantage au modèle. Pourtant, la classification en périphériques d'E/S par blocs/périphériques d'E/S de caractères suffit généralement pour servir de base à l'interaction entre le logiciel du système d'exploitation et un périphérique d'E/S autonome. Le système de fichiers, par exemple, traite uniquement avec les périphériques d'E/S par blocs génériques et laisse la partie dépendante du périphérique au logiciel de niveau inférieur.

Les périphériques présentent des vitesses de fonctionnement et des débits de transfert de données très différents, d'où une pression considérable sur le logiciel qui doit gérer avec des performances correctes une grande variété de périphériques. La figure 5.1 montre les débits de quelques périphériques courants. La majorité de ces périphériques devient de plus en plus rapide avec le temps.

Figure 5.1 • Débits de certains périphériques courants, réseaux et bus.

Périphérique	Débit de données
Clavier	10 octets/s
Souris	100 octets/s
Numériseur	400 Ko/s
Caméscope numérique	3,5 Mo/s
Réseau sans fil 802.11g	6,75 Mo/s
CD-ROM 52x	7,8 Mo/s
Ethernet rapide	12,5 Mo/s
Carte mémoire « Compact Flash »	40 Mo/s
FireWire (IEEE 1394)	50 Mo/s
USB 2.0	60 Mo/s
Réseau SONET OC-12	78 Mo/s
SCSI Ultra 2	80 Mo/s
Ethernet Gigabit	125 Mo/s
Disque dur SATA	300 Mo/s
Bande Ultrium	320 Mo/s
Bus PCI	528 Mo/s

5.1.2 Le contrôleur de périphérique

Les périphériques d'E/S sont généralement constitués d'un composant mécanique et d'un composant électronique. Il est souvent possible de séparer les deux parties pour permettre une conception modulaire et générique. Le composant électronique s'appelle un **contrôleur de périphérique** (*device controller*) ou adaptateur. Sur les ordinateurs personnels, il prend souvent la forme d'une puce sur la carte mère ou d'un circuit imprimé que l'on peut insérer dans un connecteur d'extension (carte PCI). Le composant mécanique est le périphérique lui-même, dont la disposition est illustrée par la figure 1.6.

La carte du contrôleur est généralement équipée d'un connecteur sur lequel est branché un câble que l'on connecte au périphérique. De nombreux contrôleurs sont en mesure de gérer 2, 4, voire 8 périphériques identiques. Si l'interface entre le contrôleur et le périphérique est normalisée, soit en respectant une norme internationale (ANSI, IEEE, ISO) ou une norme de fait, des sociétés peuvent fabriquer des contrôleurs ou des périphériques qui respectent cette interface. De nombreuses sociétés, par exemple, fabriquent des disques durs qui utilisent l'interface IDE, SATA, SCSI, USB ou FireWire (IEEE 1394).

L'interface entre le contrôleur et le périphérique est souvent de très bas niveau. Un disque, par exemple, peut être formaté avec 10 000 secteurs de 512 octets par piste. Le lecteur produit en réalité un flot binaire série, qui commence par un **synchroniseur initial** ou **préambule** (*preamble*), suivi des 4 096 bits d'un secteur, et se termine par une somme de contrôle appelée **code de correction d'erreurs** (ECC, *Error-Correcting Code*). Le préambule est écrit au moment du formatage du disque. Il contient le numéro du cylindre et du secteur, la taille du secteur et d'autres données similaires, ainsi que les informations de synchronisation.

Le travail du contrôleur consiste à convertir le flot binaire série en un bloc d'octets et à apporter toute correction qui s'impose en cas d'erreur. Le bloc d'octets est assemblé, bit par bit, dans une mémoire tampon qui se trouve au sein du contrôleur. Une fois que la somme de contrôle a été vérifiée et que le bloc a été déclaré sans erreur, il peut être copié dans la mémoire principale.

Le contrôleur d'écran (par exemple, un moniteur vidéo) fonctionne lui aussi comme un périphérique utilisant un flot binaire en série de bas niveau. Il lit les octets de la mémoire qui contiennent les caractères à afficher et génère les signaux employés pour la modulation du tube cathodique (CRT, *Cathode Ray Tube*) et l'affichage à l'écran. En outre, le contrôleur génère les signaux de balayage horizontal, et vertical, de même que les synchronisations de retour ligne et vertical. Sans contrôleur vidéo, le programmeur du système d'exploitation devrait programmer explicitement le balayage analogique du tube. Grâce au contrôleur vidéo, le système d'exploitation initialise le contrôleur avec quelques paramètres, tels que le nombre de caractères et de pixels par ligne et le nombre de lignes par écran, et il laisse la transmission et la gestion de l'affichage à l'écran sous la responsabilité du contrôleur. Les écrans plats à matrice active (TFT, *Thin Film Transistor*) sont différents mais sont aussi compliqués.

5.1.3 Les E/S projetées en mémoire

Chaque contrôleur possède quelques registres, appelés registres de contrôle ou registres de commande, qui servent à la communication avec le processeur. En écrivant dans ces registres, le système d'exploitation demande au périphérique de lui délivrer des données, d'en accepter, de se mettre sous ou hors tension lui-même ou encore d'effectuer une action donnée. En lisant ces registres, le système d'exploitation peut connaître l'état du périphérique, savoir s'il est prêt à accepter une nouvelle commande, etc.

En plus des registres de contrôle, de nombreux périphériques sont équipés de mémoire tampon pour les données que le système d'exploitation peut lire ou écrire. Par exemple, les pixels affichés à l'écran d'un ordinateur proviennent souvent de la mémoire vidéo, qui n'est autre qu'un tampon de données, dans lequel les programmes ou le système d'exploitation peuvent écrire.

Il faut alors savoir comment le processeur communique avec les registres de contrôle et les mémoires tampon de données du périphérique. Il existe deux possibilités. Dans le premier cas, chaque registre de contrôle se voit assigner un numéro de **port d'E/S** (*I/O port*), un entier de 8 ou 16 bits. L'ensemble de tous les ports d'E/S forme l'**espace d'E/S** (*I/O port space*), qui est un espace protégé c'est-à-dire qu'un programme utilisateur ne peut pas y accéder (seul le système d'exploitation peut y accéder). En utilisant une instruction d'E/S spéciale comme

```
IN REG,PORT,
```

le processeur lit dans le registre de contrôle `PORT` et stocke le résultat dans le registre `REG` du processeur. De même, avec

```
OUT PORT,REG
```

le processeur écrit le contenu du registre `REG` dans un registre du contrôle. La majorité des ordinateurs récents, comme presque tous les mainframes, tels l'IBM 360 et ses successeurs, fonctionnent ainsi.

Dans cette configuration, les espaces d'adressage de la mémoire et des E/S sont distincts, comme l'illustre la figure 5.2(a). Ainsi, les instructions

```
IN R0,4
```

et

```
MOV R0,4
```

sont complètement différentes dans cette conception. La première instruction lit le contenu du port d'E/S 4 et le place en `R0` alors que la seconde lit le contenu du mot 4 de la mémoire et le place dans `R0`. Les valeurs `4` de ces exemples se réfèrent à des espaces d'adressage différents et sans aucun rapport entre eux.

La seconde approche, introduite avec le PDP-11, consiste à projeter tous les registres de contrôle dans l'espace mémoire, comme le montre la figure 5.2(b). Chaque registre de contrôle se voit attribuer une adresse mémoire unique à laquelle aucune mémoire n'est assignée. Ce système est appelé **E/S projetées ou mappées en mémoire** (*memory-mapped I/O*). En général, les adresses assignées se trouvent au sommet de l'espace d'adressage. La figure 5.2(c) montre un schéma hybride, dans lequel les mémoires tampon de données sont projetées en mémoire et des ports d'E/S séparés pour les registres de contrôle. Le Pentium exploite cette architecture. Les adresses de 640 Ko à 1 Mo sont réservées aux mémoires tampon de données des périphériques dans les ordinateurs compatibles IBM PC, en plus des ports d'E/S de 0 à 64 Ko.

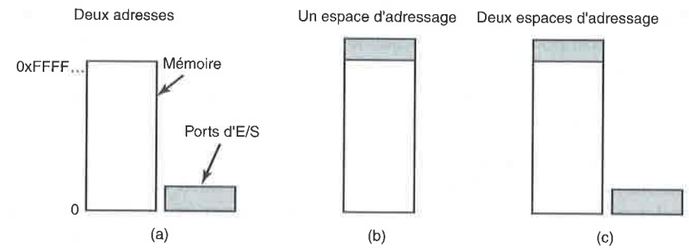

Figure 5.2 • (a) Espaces mémoire et E/S séparés. (b) E/S projetées en mémoire. (c) Système hybride.

Comment fonctionnent ces architectures ? Dans tous les cas, lorsque le processeur veut lire un mot à partir de la mémoire ou d'un port d'E/S, il place l'adresse nécessaire sur les lignes d'adressage du bus, puis revendique un signal `READ` sur la ligne de contrôle du bus. Un second signal précise s'il s'agit d'une adresse dans l'espace mémoire ou dans l'espace d'E/S. Dans le cas de l'espace mémoire, la mémoire répond à la requête. Dans le cas de l'espace d'E/S, c'est le périphérique d'E/S qui répond à la requête. Si on dispose uniquement de l'espace mémoire [comme dans la figure 5.2(b)], chaque module mémoire et chaque périphérique d'E/S comparent les lignes d'adressage à la plage d'adresses qu'ils desservent. Si l'adresse se trouve dans cette plage, ils répondent à la requête. Dans la mesure où une adresse n'est jamais assignée simultanément à la mémoire et à un périphérique d'E/S, il n'existe aucune ambiguïté ni aucun conflit.

Les deux schémas d'adressage possèdent des avantages et des inconvénients. Commençons par les avantages des E/S projetées en mémoire. Si des instructions d'E/S spéciales sont nécessaires pour lire et écrire dans les registres de contrôle du périphérique, leur accès demande l'usage de code assembleur puisqu'il n'existe aucun moyen d'exécuter une instruction `IN` ou `OUT` en C ou C++. L'appel d'une telle procédure surcharge le contrôle des E/S. Par contre, avec les E/S projetées en mémoire, les registres de contrôle sont de simples variables en mémoire et peuvent être adressés en C, à l'instar de toute autre variable. Ainsi, avec les E/S projetées en mémoire, un pilote de périphérique d'E/S peut être entièrement écrit en C. Sans E/S projetées en mémoire, il faut du code assembleur.

De plus, avec les E/S projetées en mémoire, aucun mécanisme de protection spécial n'est nécessaire pour empêcher les processus utilisateur d'effectuer des E/S. Il suffit que le système d'exploitation s'abstienne de placer dans l'espace d'adressage virtuel d'un utilisateur la portion de l'espace d'adressage contenant les registres de contrôle. En outre, si les registres de contrôle de chaque périphérique sont placés dans une page différente de l'espace d'adressage, le système d'exploitation peut donner à un utilisateur le contrôle de certains périphériques et pas à d'autres, en incluant simplement les pages nécessaires dans son tableau de pages. Une telle configuration permet de placer différents pilotes de périphériques dans des espaces d'adressage différents, ce qui réduit la taille du noyau et évite les interférences entre les différents pilotes.

Enfin, avec les E/S projetées en mémoire, chaque instruction qui peut faire référence à la mémoire est également en mesure de faire référence à des registres de contrôle. Par exemple, s'il y a une instruction `TEST` qui teste si un mot mémoire est égal à 0, elle peut également servir à tester si un registre de contrôle est égal à 0, qui peut être le signal indiquant que le périphérique est inactif et qu'il est prêt à accepter une nouvelle commande. Le code assembleur peut prendre la forme suivante :

```
LOOP:   TEST PORT_4       // vérifie que le port 4 est à 0
        BEQ READY         // si oui, état prêt
        BRANCH LOOP       // sinon, attendre
READY:
```

S'il n'y a pas d'E/S projetées en mémoire, le registre de contrôle doit d'abord être lu dans le processeur, puis testé, ce qui nécessite deux instructions au lieu d'une. Dans le cas de la boucle qui précède, il faut ajouter une quatrième instruction, ce qui réduit légèrement la sensibilité de la détection de l'état inactif d'un périphérique.

Dans un ordinateur, presque tout implique des échanges entre unités fonctionnelles, ce qui est également le cas ici. Les E/S projetées en mémoire possèdent par ailleurs des inconvénients. D'abord, la plupart des ordinateurs actuels disposent d'une forme de mise en cache de mots mémoire. Il serait désastreux de placer dans le cache le registre de contrôle d'un périphérique. Prenons l'exemple de la boucle du code assembleur précédent en présence d'un cache. La première référence à `PORT_4` le place en cache. Les références suivantes prennent la valeur du cache sans demander l'état du périphérique. Ensuite, lorsque le périphérique est prêt, le logiciel n'a aucun moyen de le savoir. Et la boucle continue indéfiniment.

Pour éviter ce type de situation avec des E/S projetées en mémoire, le matériel doit pouvoir désactiver la mise en cache sélectivement, par page par exemple. Cette fonctionnalité ajoute une certaine complexité au matériel et au système d'exploitation, qui doit gérer la mise en cache sélective.

Ensuite, s'il n'y a qu'un espace d'adressage, tous les modules mémoire et tous les périphériques d'E/S doivent examiner toutes les adresses mémoire pour savoir qui doit répondre. Si l'ordinateur ne possède qu'un bus, comme dans la figure 5.3(a), la consultation de chaque adresse par tous est simple.

Toutefois, les ordinateurs personnels modernes tendent à posséder un bus mémoire à haut débit dédié [voir figure 5.3(b)], propriété que l'on retrouve aussi dans les mainframes. Ce bus est destiné à optimiser les performances de la mémoire, sans imposer de contraintes particulières aux périphériques d'E/S lents. Les systèmes Pentium, par exemple, peuvent avoir plusieurs bus externes (mémoire, PCI, SCSI, USB, ISA), comme le montre la figure 1.12.

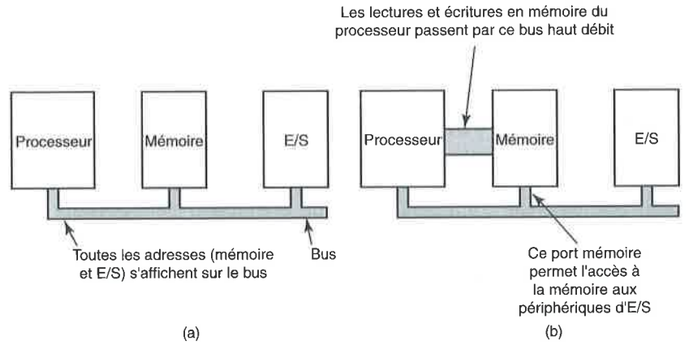

Figure 5.3 • (a) Une architecture à bus unique. (b) Une architecture à deux bus mémoire.

Le problème que pose un bus mémoire séparé sur les ordinateurs ayant des E/S projetées en mémoire est que les périphériques d'E/S ne disposent d'aucun moyen de voir les adresses mémoire lorsqu'elles passent par le bus mémoire ; ils ne peuvent donc pas y répondre. En outre, il faut prendre des mesures spéciales pour faire fonctionner sur un système équipé de plusieurs bus les E/S projetées en mémoire. L'une des solutions consiste à envoyer d'abord toutes les références mémoire à la mémoire. Si celle-ci ne répond pas, le processeur essaie les autres bus. Cette conception peut fonctionner, mais elle augmente la complexité du matériel.

Une deuxième solution consiste à placer un fureteur (*snooping device*) sur le bus mémoire pour transmettre toutes les adresses présentées aux périphériques d'E/S potentiellement intéressés. Le problème, dans ce cas, est que les périphériques d'E/S ne sont pas tous en mesure de traiter les requêtes à la même vitesse que la mémoire.

La troisième possibilité, utilisée dans la configuration du Pentium de la figure 1.12, consiste à filtrer les adresses dans la puce de gestion du bus PCI. Cette puce peut contenir des registres de plages préchargés au moment du démarrage de la machine. Par exemple, les adresses de 640 Ko à 1 Mo peuvent être marquées comme faisant partie de la plage non destinée à la mémoire. Les adresses qui entrent dans l'une des plages marquées comme non destinées à la mémoire sont transférées au bus PCI et non à la mémoire. Cette configuration pose néanmoins un problème : celui de savoir, au moment du démarrage, quelles adresses mémoire ne sont pas réellement des adresses mémoire. En résumé, chaque configuration présente des avantages et des inconvénients, et les compromis sont inévitables.

5.1.4 L'accès direct à la mémoire (DMA)

Que le processeur possède ou non des E/S projetées en mémoire, il doit adresser les contrôleurs de périphérique pour échanger des données avec eux. Le processeur peut demander les données au contrôleur d'E/S octet par octet, mais il gaspille ainsi le

temps du processeur. On fait donc souvent appel à une configuration différente et plus performante, appelée **accès direct à la mémoire** (DMA, *Direct Memory Access*). Le système d'exploitation peut exploiter le DMA uniquement si le matériel est équipé d'un contrôleur DMA, ce qui est le cas de la majorité des systèmes. Il arrive que le contrôleur soit intégré aux contrôleurs de disques ou à d'autres contrôleurs, mais une telle conception demande un contrôleur DMA distinct pour chaque périphérique. En général, il n'existe qu'un seul contrôleur DMA (par exemple, sur la carte mère) pour réguler les transferts, souvent simultanés, vers plusieurs périphériques.

Quel que soit son emplacement, le contrôleur DMA a accès au bus système sans être tributaire du processeur, comme l'illustre la figure 5.4. Il contient plusieurs registres dans lesquels le processeur peut lire et écrire, dont un registre d'adresses mémoire, un registre pour le nombre d'octets et un ou plusieurs registres de contrôle. Ces derniers spécifient le port d'E/S à employer, la direction du transfert (lecture du périphérique d'E/S ou écriture dans le périphérique d'E/S), l'unité de transfert (un octet ou un mot à la fois) et le nombre d'octets à transférer par rafale.

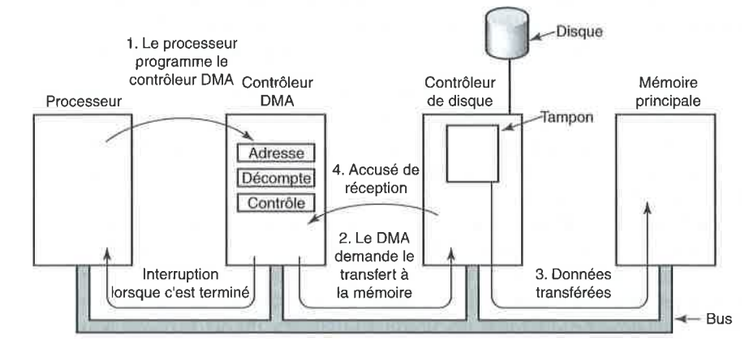

Figure 5.4 • Fonctionnement d'un transfert DMA.

Pour expliquer le fonctionnement du DMA, commençons par étudier de quelle manière se font les lectures de disque quand il n'y a pas de DMA. Tout d'abord, le contrôleur lit le bloc (soit un ou plusieurs secteurs) du disque en série, bit par bit, jusqu'à ce que tout le bloc se trouve dans une mémoire tampon interne. Ensuite, il calcule la somme de contrôle pour vérifier qu'aucune erreur de lecture ne s'est produite. Le contrôleur déclenche alors une interruption. Lorsque le système d'exploitation prend la main, il lit le bloc du disque contenu dans la mémoire tampon du contrôleur, un octet ou un mot à la fois, en exécutant une boucle. À chaque itération de boucle, un octet ou un mot d'un registre du contrôleur est lu et stocké en mémoire principale.

Si l'on fait appel au DMA, la procédure est différente. Pour commencer, le processeur programme le contrôleur DMA en paramétrant ses registres pour qu'il sache quoi