Mémoires vives

Dans les trois chapitres précédents nous avons étudié la plupart des briques de base utilisées pour la conception d'ordinateurs. Nous allons maintenant monter d'un niveau et aborder les principes de réalisation des principaux blocs fonctionnels à partir de ces briques. Nous commençons (c'est un choix arbitraire) par la mémoire vive. La première partie de ce chapitre sera consacrée à son organisation matérielle. Dans la seconde nous ébaucherons son organisation logique, dont l'étude doit être approfondie dans le cours sur les systèmes d'exploitation.

V.1 Introduction

Nous savons que dans un ordinateur toutes les informations : valeur numérique, instruction, adresse, symbole (chiffre, lettre,…) etc… sont manipulées sous une forme binaire. Ces informations doivent en général être conservées pendant un certain temps pour permettre leur exploitation. Ce rôle est dévolu aux mémoires chargées de conserver programmes, données provenant de l'extérieur, résultats intermédiaires, données à transférer à l'extérieur, etc. Nous avons déjà rencontré les registres de mémorisation, mais ceux-ci ne sont pas adaptés aux grandes capacités de stockage.

Il faut pour cela des mémoires à lecture et écriture ou mémoires vives, qui permettent d'enregistrer une information, de la conserver et de la restituer. Ces mémoires sont, d'autre part, à accès aléatoire (RAM : Random Acces Memory) c'est-à-dire que le temps d'accès à l'information est indépendant de sa place en mémoire. Cette appellation, d'origine historique, est toujours synonyme de mémoire vive. Bien que très répandue cette appellation n'est plus suffisante car tous les circuits à semi-conducteur sont aujourd'hui à accès aléatoire. L'accès séquentiel ne porte plus que sur les mémoires magnétiques (disques ou bandes). Par contre, une mémoire vive est volatile : la conservation de son contenu nécessite la permanence de son alimentation électrique.

L'information élémentaire, ou bit (binary digit), est mémorisée dans une cellule ou point mémoire. Nous étudierons plus loin les deux principales technologies utilisées pour réaliser une cellule. Ces cellules sont groupées en mots de n bits, c'est-à-dire que les n bits sont traités (écrits ou lus) simultanément. On ne peut pas modifier un seul bit, il faut transférer le mot dans un registre, modifier le bit puis réécrire le mot en mémoire. Par ailleurs, les cellules sont arrangées en bloc mémoire.

Extérieurement, et en ne tenant compte que des signaux logiques, un bloc mémoire peut être représenté comme sur la figure 1. Pour pouvoir identifier individuellement chaque mot on utilise k lignes d'adresse. La taille d'un bloc mémoire est donc 2^k, le premier mot se situant à l'adresse 0 et le dernier à l'adresse 2^k - 1. Une ligne de commande (R/W) indique si la mémoire est accédée en écriture (l'information doit être mémorisée) ou en lecture (l'information doit être

restituée). Sur ce schéma on distingue deux canaux de n lignes en entrée et en sortie, mais dans d'autres cas les accès en entrée et en sortie peuvent être confondus en un seul canal bidirectionnel. Nous verrons l'intérêt de la ligne de validation ou de sélection du bloc (CS) un peu plus loin.

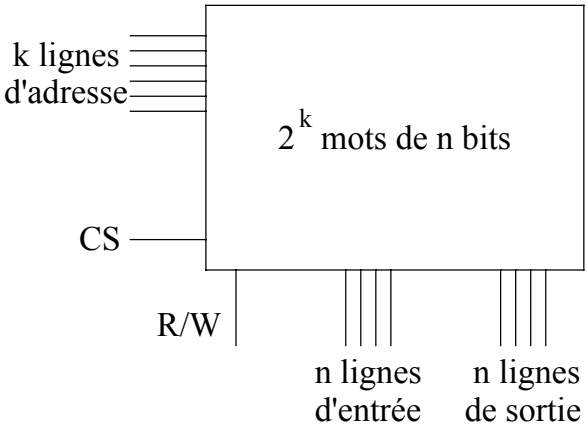


Figure 1

V.2 Adressage bidimensionnel ou matriciel

L'organisation des cellules à l'intérieur d'un bloc la plus simple à imaginer correspond au schéma suivant, chaque ligne correspond à un mot de n bits :

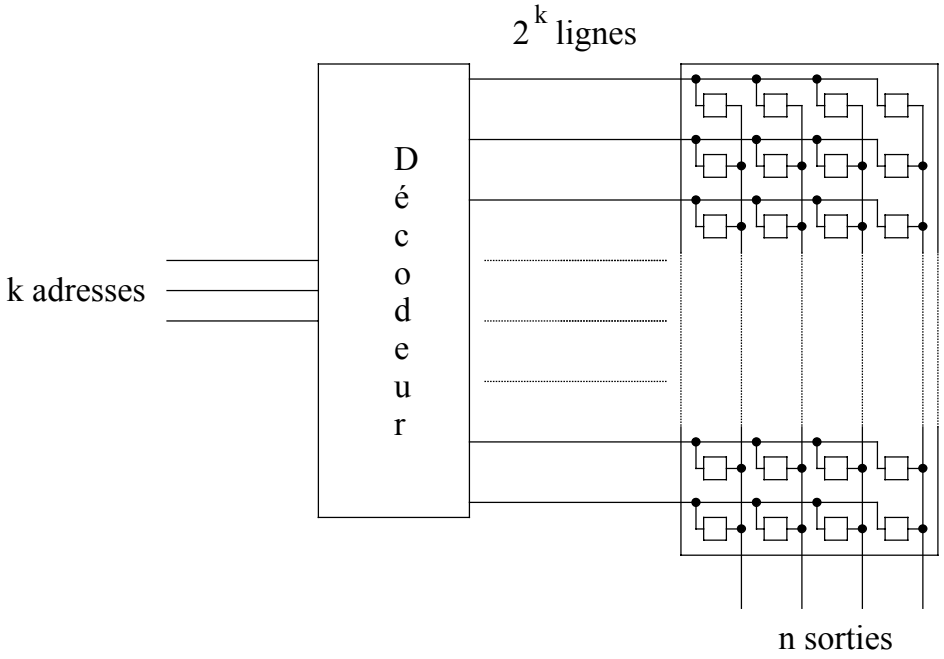


Figure 2

Pour simplifier la figure, chaque cellule y est matérialisée avec uniquement une ligne de sélection et une ligne de sortie. Si la ligne de sélection est à "0", la cellule est "isolée" de la sortie. Si la ligne de sélection est à "1", l'information mémorisée se retrouve sur la ligne de sortie.

La figure correspond au mécanisme de lecture, mais le principe est également valable en écriture. En fonction de l'adresse, le décodeur active une des 2^k lignes. Ainsi seules les cellules correspondant à l'adresse demandée sont sélectionnées et l'information mémorisée est alors disponible en sortie. Cette architecture très simple n'est pas la plus économique en terme de nombre de portes.

Considérons par exemple une mémoire contenant 512 mots de 4 bits soient 2048 bits. C'est-à-dire k = 9 et n = 4. Il faut 512 portes ET pour réaliser le décodeur. Une économie importante peut être obtenue en organisant la mémoire en une matrice de 64 lignes et 32 colonnes (2048 = 64 x 32). Chacune de ces 64 lignes peut être sélectionnée en utilisant 6 bits d'adresse. Comme nous ne voulons que des mots de 4 bits, il faut encore utiliser 4 multiplexeurs chacun sélectionnant une ligne sur 8. Les 3 derniers bits d'adresse sont affectés à ces multiplexeurs. Cette architecture est dénommée adressage X-Y ou bidimensionnel. Dans ce cas 64 portes ET sont nécessaires pour le décodeur et 9 portes (8 ET et 1 OU) pour chacun des multiplexeurs, soit au total 64 + (9 x 4) = 100 portes. Cela représente cinq fois moins de portes.

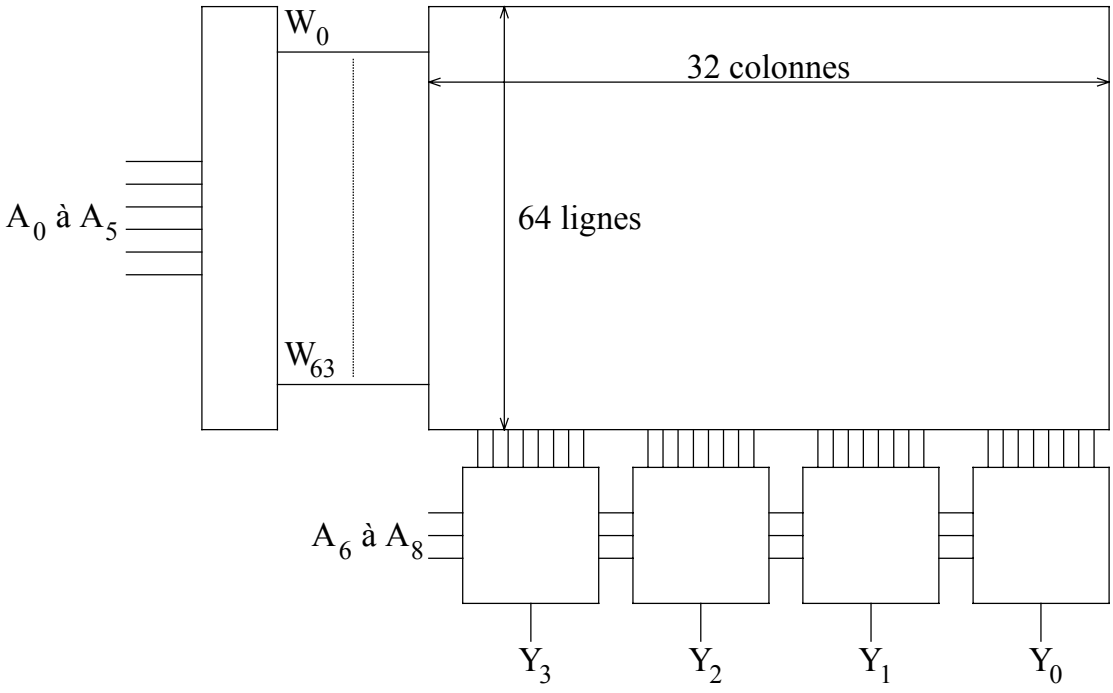


Figure 3

L'organisation matricielle des blocs mémoires permet également d'éviter des pistes trop longues pour la distribution des différents signaux aux cellules. Les constantes de temps et les pertes de charge sont ainsi réduites.

Nous n'avons considéré que deux cas de figure pour l'organisation matricielle du bloc mémoire : 512 lignes de 4 colonnes et 64 lignes de 32 colonnes. D'autres variantes sont évidemment possibles, pour certaines le nombre de portes nécessaires est résumé dans le tableau suivant. Nous constatons que le minimum se situe pour une organisation "carrée", pour laquelle les nombres de lignes et de colonnes sont égaux ou diffèrent d'un facteur 2.

nb lignes	nb colonnes	Décodeur	Multiplexeur	Total
512	4	512	0	512
256	8	256	3	268
128	16	128	5	148
64	32	64	9	100
32	64	32	17	100
16	128	16	33	148
8	256	8	65	268

Table 1

Très souvent, les blocs mémoires comportent autant de lignes que de colonnes. Les mêmes lignes d'adresse peuvent alors être utilisées pour identifier successivement la ligne puis la colonne. Cela permet de réduire le nombre de broches de connexion, donc l'encombrement et le coût des circuits. Cependant cela demande environ deux fois plus de temps par transmettre l'adresse complète. Par contre, si on cherche à accéder à des informations stockées dans une même ligne il peut être possible de définir une fois la ligne, puis pour chaque mot de n'avoir à envoyer que l'adresse de la colonne. Il faut alors au moins un signal de commande supplémentaire pour indiquer que l'adresse correspond à une ligne ou une colonne.

Parmi les caractéristiques d'une mémoire nous trouvons la capacité et le format. La capacité représente le nombre total de bits et le format correspond à la longueur des mots. Le nombre de bits d'adresse k définit le nombre total de mots de la mémoire, si n est le nombre de bits par mot, la capacité de la mémoire est donnée par :

$$\text{Capacité} = 2^k \text{ mots} = 2^k \text{ x n bits}$$

Cette capacité est exprimée en multiple de 1024 ou kilo. La table suivante résume la valeur des autres préfixes utilisés pour exprimer les capacités des mémoires :

Symbole	Préfixe	Capacité		
1 k	(kilo)	2^{10}	=	1024
1 M	(méga)	2^{20}	=	1048576
1 G	(giga)	2^{30}	=	1073741824
1 T	(tera)	2^{40}	=	1099511627776

Table 2

La figure 4, présente une organisation logique possible pour une mémoire de 128 mots de 8 bits. Ici chaque mot est stocké dans une case de 8 bits, tel un registre. Cette case reçoit en entrée 8 lignes de données et une ligne de chargement. Elle dispose de 8 lignes de sortie fournissant le contenu du registre. Chacune de ces lignes est commandée par une porte "3 états". Ces cases sont organisées en une matrice de 32 lignes et 4 colonnes. Les 7 bits d'adresse sont séparés en deux groupes, 5 bits pour identifier la ligne et 2 bits pour la colonne.

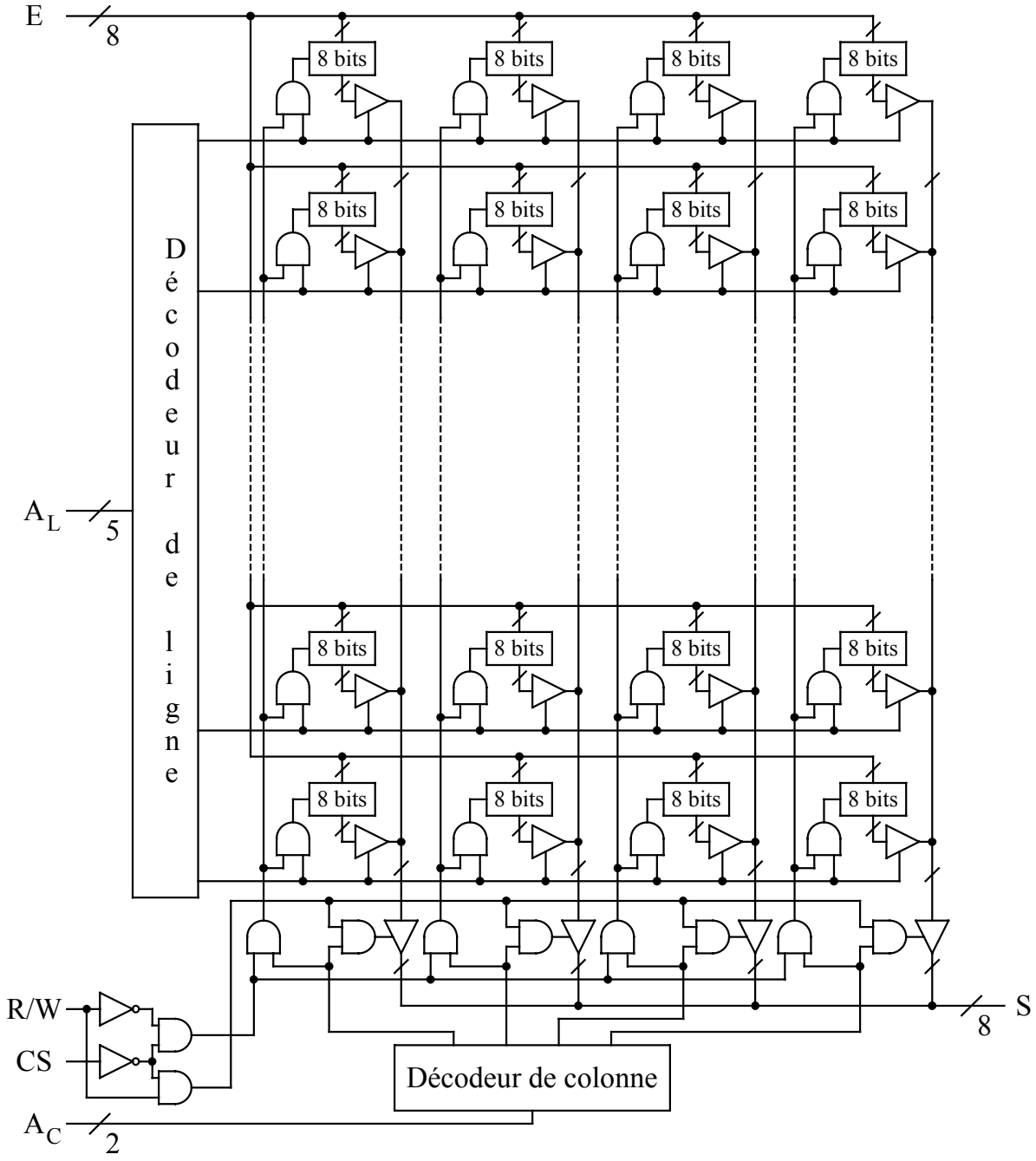


Figure 4

Le signal de sélection du boîtier (CS) valide les fonctions d'écriture et de lecture. Le boîtier est bloqué pour CS = 1. Lorsque CS = 0, une fonction d'écriture correspond à R/W = 0 et une fonction de lecture à R/W = 1.

En écriture, le mot à charger doit être présent sur l'entrée E du circuit. Ces données sont distribuées simultanément sur toutes les cases de 8 bits. La ligne désignée par l'adresse A_L est à 1. Le signal de chargement est transmis à la seule colonne identifiée par l'adresse A_C. Seul le registre à l'intersection de cette ligne et de cette colonne est donc chargé.

En lecture, les quatre cases de la ligne sélectionnée fournissent leur contenu sur les quatre bus verticaux. Une seule des quatre portes "3 états", au bas du schéma, est connectée à la sortie S du boîtier. Cette porte "3 états" fournit une amplification des signaux.

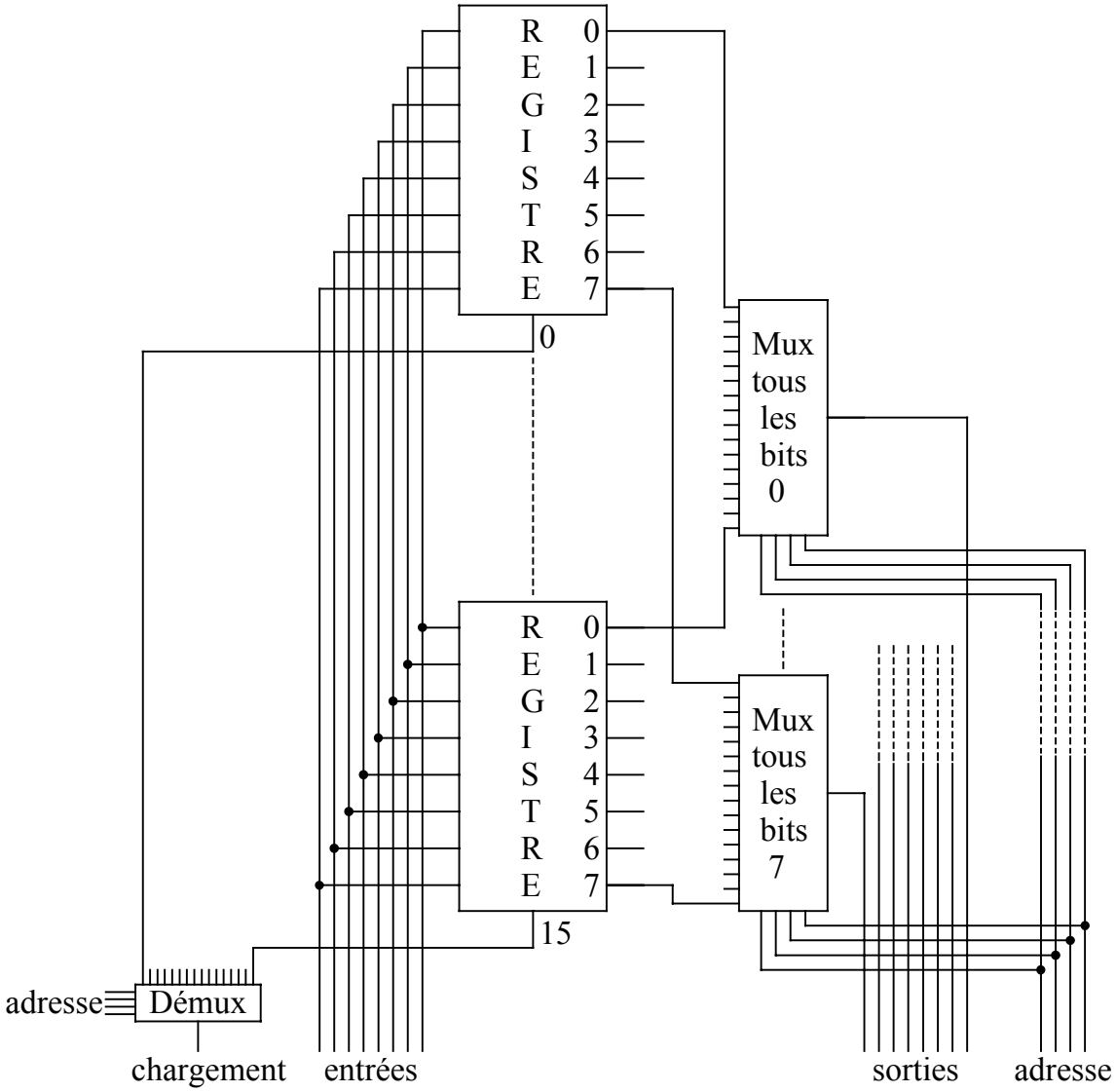


Figure 5

La figure 5 illustre une autre organisation de la lecture. Pour alléger le dessin, nous avons considéré une mémoire de 16 mots de 8 bits. Le principe du chargement est comparable au scénario précédent : les données sont distribuées sur tous les registres, mais un seul reçoit le signal de chargement. Par contre la sortie fait appel à 8 multiplexeurs. Chacun recevant les bits de poids identiques en provenance des 16 registres. L'adresse est distribuée sur tous ces multiplexeurs, et on trouve donc en sortie tous les bits du registre identifié.

V.3 Assemblage de blocs mémoires

Les techniques d'intégration ne permettent pas d'obtenir des boîtiers ayant des capacités ou des formats suffisants pour toutes les applications. Il est alors nécessaire d'associer plusieurs boîtiers pour augmenter la longueur des mots ou le nombre de mots. L'association de plusieurs blocs peut permettre d'améliorer les performances temporelles de la mémoire en faisant fonctionner plusieurs blocs en parallèle.

V.3.a Augmentation de la longueur des mots

La figure suivante montre qu'il est aisé d'associer deux boîtiers de 2^k mots de n bits pour obtenir un bloc de 2^k mots de 2 n bits. L'adressage doit être appliqué simultanément aux deux circuits, l'un fournissant les n bits de bas poids et l'autre les n bits de haut poids.

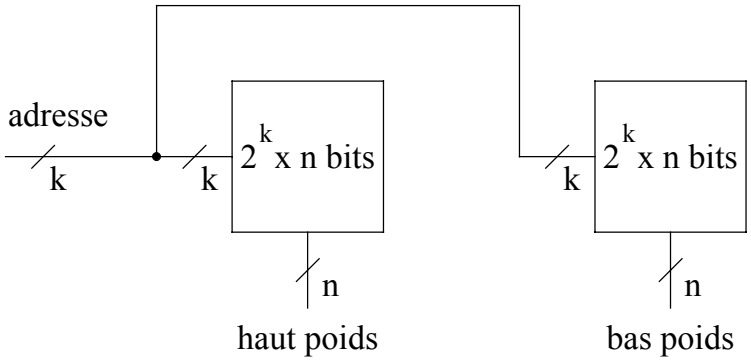


Figure 6

V.3.b Augmentation du nombre de mots

De même la figure suivante montre la réalisation d'un bloc de 4 x 2^k mots de n bits à l'aide de 4 boîtiers de 2^k x n bits. Il nous faut k+2 lignes d'adresse. Les k bits de bas poids de l'adresse sont appliqués simultanément sur les 4 boîtiers. Les deux bits de haut poids attaquent un décodeur à quatre sorties. Chacune de ces quatre lignes permet de sélectionner un boîtier (entrée de validation du boîtier : CS). Un seul boîtier est alors connecté aux lignes de sortie.

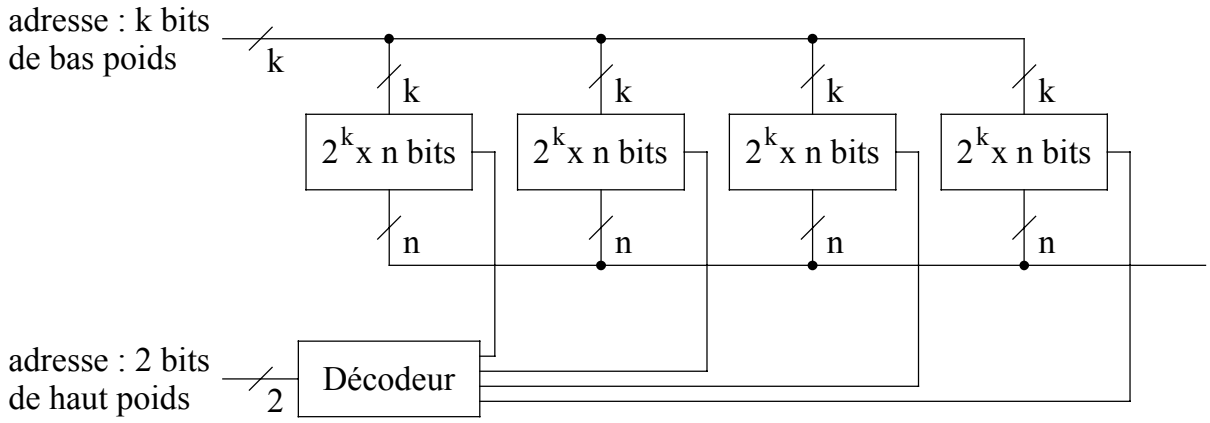
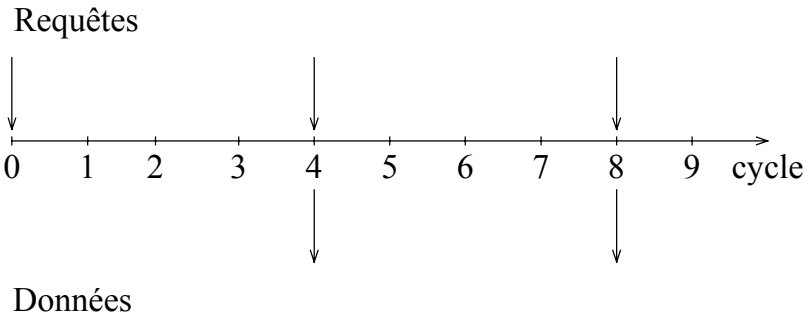


Figure 7

V.3.c Entrelacement

Une mémoire entrelacée à n voies est constituée de n blocs. Le bloc numéroté i, avec $i \in [0, n-1]$, contient toutes les cellules dont les adresses sont égales à i modulo n (add = k n + i). De cette manière deux mots à des adresses consécutives sont rangés dans deux blocs différents. Cette organisation permet de réduire le temps d'accès à la mémoire lors de la lecture ou de l'écriture par groupe de mots.

Supposons par exemple qu'une unité centrale, de temps de cycle δt, veuille lire un ensemble de N mots à des adresses consécutives. Dans une mémoire non entrelacée cette opération demande un temps $\Delta t_1 = N \,.\, \delta t_a$, où δt_a est le temps d'accès à la mémoire, après réception de la première requête par l'unité de gestion de la mémoire (fig. 8).

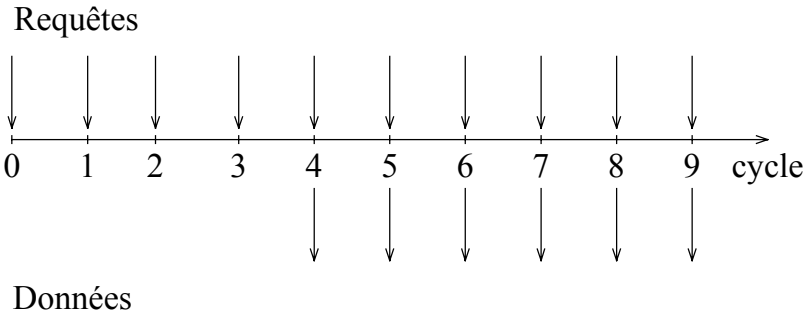


Mémoire non entrelacée

Figure 8

Dans une mémoire entrelacée à n voies, telle que $\delta t_a \leq n\, \delta t$, la première requête est transmise par l'unité de gestion de la mémoire au bloc i. La seconde requête, au cycle d'horloge

suivant, est transmise au bloc i+1, et ainsi de suite. Chacune des n premières requêtes est ainsi transmise à un bloc différent. Ces blocs fonctionnent en parallèle. A l'instant δt_a mesuré après la première requête la première la donnée est disponible. Elle est collectée par l'unité centrale au $n^{ième}$ cycle. La deuxième est disponible un cycle (δt) plus tard, et ainsi de suite (fig. 9). Les données suivantes pourront être récupérées à chaque cycle. Il faut donc un temps $\Delta t_2 = (n - 1 + N) . \delta t$ pour lire les N mots. Il faut attendre $n - 1$ cycles sans recevoir aucune donnée.



Mémoire entrelacée à 4 voies

Figure 9

Cette méthode, comparable au traitement en pipe-line des instructions que nous étudierons dans le chapitre consacré à l'unité centrale, est d'autant plus efficace que très souvent on doit accéder à des informations (instructions ou données dans des structures) à des adresses consécutives. Les groupes de mots lus sont ensuite stockés dans une mémoire cache (cf. plus loin).

V.4 Réduction du nombre de broches

Les fabricants de mémoires cherchent à réduire le nombre de broches des boîtiers, pour augmenter la densité d'implantation sur les circuits imprimés. Deux méthodes peuvent être utilisées pour réduire le brochage des signaux logiques.

Il est possible de diviser par deux le nombre de broches pour l'adressage en chargeant successivement l'adresse de ligne puis l'adresse de colonne (fig. 10). La mémoire a alors une organisation "carrée", avec autant de lignes que de colonnes. Deux signaux (RAS : Row Address Strobe et CAS : Column Address Strobe) sont nécessaires pour charger ces deux adresses dans deux registres internes (RAR : Row Address Register et CAR : Column Address Register). Le contenu de ces registres est exploité respectivement par les décodeurs de ligne et de colonne. Pour accéder à plusieurs mots d'une même ligne, il n'est pas nécessaire de charger l'adresse de la ligne à chaque requête, il suffit de modifier l'adresse de colonne, on parle alors d'accès en mode page.

On peut également tirer profit de ce qu'on n'effectue jamais une lecture et une écriture simultanément. Il est donc possible d'utiliser les mêmes broches en lecture et en écriture. Pour cela on fait appel à des porte "3 états" (fig. 11).

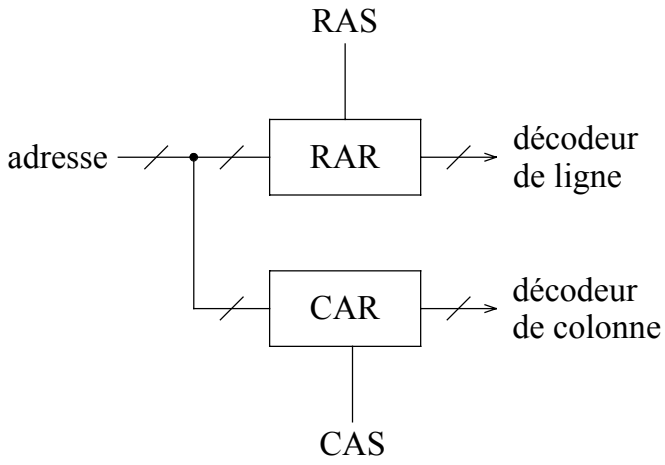


Figure 10

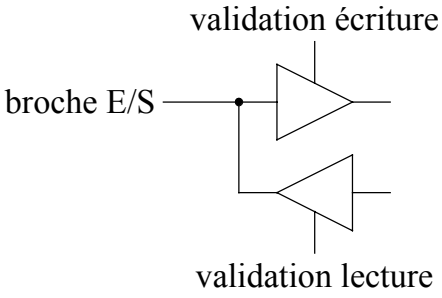


Figure 11

V.5. Les deux familles de RAM

Pour réaliser le point mémoire on dispose de deux techniques. On peut utiliser des bascules. Selon que sont utilisées des bascules du type D ou R-S, il faut une ou deux lignes pour amener le bit à charger. Avec une seule ligne il faut un inverseur par point mémoire. Les bascules garantissent la mémorisation de l'information aussi longtemps que l'alimentation électrique est maintenue sur la mémoire. Ces mémoires sont dites RAM statiques (SRAM).

Dans les RAM dynamiques (DRAM) l'élément de mémorisation est constitué par un condensateur et un transistor à effet de champ (généralement réalisé en technique MOS). Ce condensateur joue le rôle d'un interrupteur commandé. L'information est mémorisée sous la forme d'une charge électrique stockée dans le condensateur. Cette technique permet une plus grande densité d'intégration, car un point mémoire nécessite environ deux à quatre fois moins de place que dans une mémoire statique. Par contre, du fait des courants de fuite le condensateur a tendance à se décharger. C'est pourquoi les RAM dynamiques doivent être rafraîchies régulièrement pour entretenir la mémorisation : il s'agit de lire l'information avant qu'elle n'ait

totalement disparu et de la recharger. Par ailleurs, la lecture étant destructive il est également nécessaire de restaurer la charge électrique à la fin de l'opération.

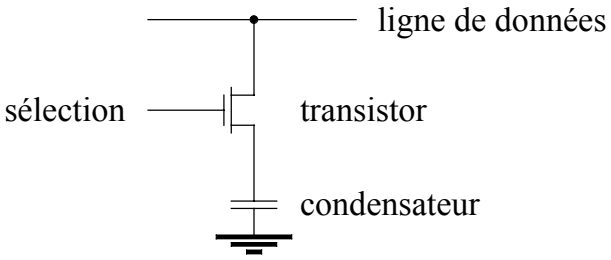


Figure 12

Ce rafraîchissement indispensable a plusieurs conséquences. Tout d'abord il complique la gestion des mémoires dynamiques car il faut tenir compte des actions de rafraîchissement qui sont prioritaires. D'autre part, la durée de ces actions augmente le temps d'accès aux informations. Le temps d'attente des données est variable selon que la lecture est interrompue ou non par des opérations de rafraîchissement et la quantité de cellules à restaurer. Il faut donc se placer dans le cas le plus défavorable pour déterminer le temps d'accès à utiliser en pratique.

En général les mémoires dynamiques, qui offrent une plus grande densité d'information et un coût par bit plus faible, sont utilisées pour la mémoire centrale, alors que les mémoires statiques, plus rapides, sont utilisées pour les caches.

V.6 Cycles de fonctionnement des mémoires

Deux paramètres caractérisent la vitesse d'une mémoire. Le temps d'accès (t_a) représente le temps qui sépare une demande de lecture de l'obtention de l'information. Le temps de cycle (t_c) correspond à l'intervalle de temps minimum qui sépare deux demandes successives en lecture ou écriture. La fréquence maximum de fonctionnement est égale à l'inverse du temps de cycle. Un autre paramètre est la bande passante qui caractérise le débit maximum en bits par seconde. En accès aléatoire cette bande passante est égale à :

$$B = \frac{n}{t_c}$$

où n est le nombre de bits transférés par cycle. Mais pour des données consécutives, certaines architectures (entrelacement, etc.) et certains modes de lecture (rafales, etc.) peuvent permettre d'approcher la limite :

$$B_{max} = \frac{n}{t_a}$$

V.6.a Consommation d'une mémoire

Les constructeurs cherchent à limiter la consommation des boîtiers de mémoire qui peuvent être nombreux dans un système. Pour certaines mémoires on peut distinguer trois états avec chacun un niveau différent de consommation. L'état actif correspond au cas d'une opération de lecture ou d'écriture. La puissance dissipée dépend du débit. Dans l'état inactif ou passif, la mémoire alimentée normalement n'est pas sélectionnée. Les commandes de lecture et écriture envoyées sur un boîtier non sélectionné n'agissent pas. Les boîtiers qui consomment moins dans cet état sont dits "automatic power down". Dans l'état de veille, la mémoire est alimentée sous tension réduite avec des batteries. Elle ne peut être ni lue, ni écrite. Elle se contente de conserver son information : un circuit interne avec horloge et compteur permet un rafraîchissement automatique régulier de toutes les cellules sans intervention externe. Cet état permet de réduire la consommation.

V.6.b Cycle de lecture

C'est le plus simple des cycles. La procédure, schématisée sur la figure 13, consiste à :

- Etablir l'adresse;
- Afficher la fonction de lecture;
- Sélectionner le ou les boîtiers nécessaires;
- Après un certain délai correspondant au temps d'accès, l'information apparaît sur la sortie qui passe en basse impédance. L'information reste présente jusqu'à la fin du cycle.

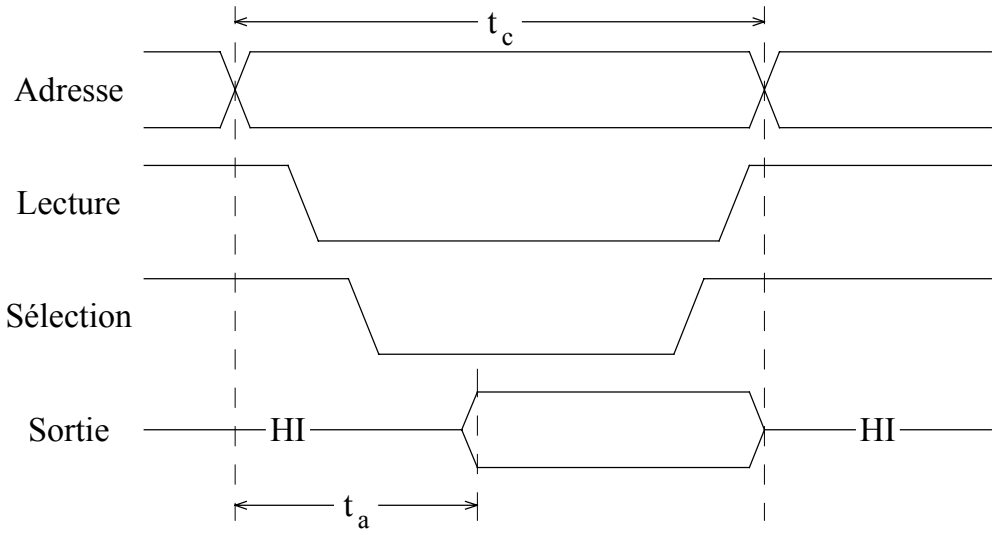


Figure 13

Il s'agit uniquement d'un chronogramme de principe. La polarité, la durée de stabilité de chaque signal, ainsi que les retards relatifs (ou phases) entre signaux sont précisés par le constructeur.

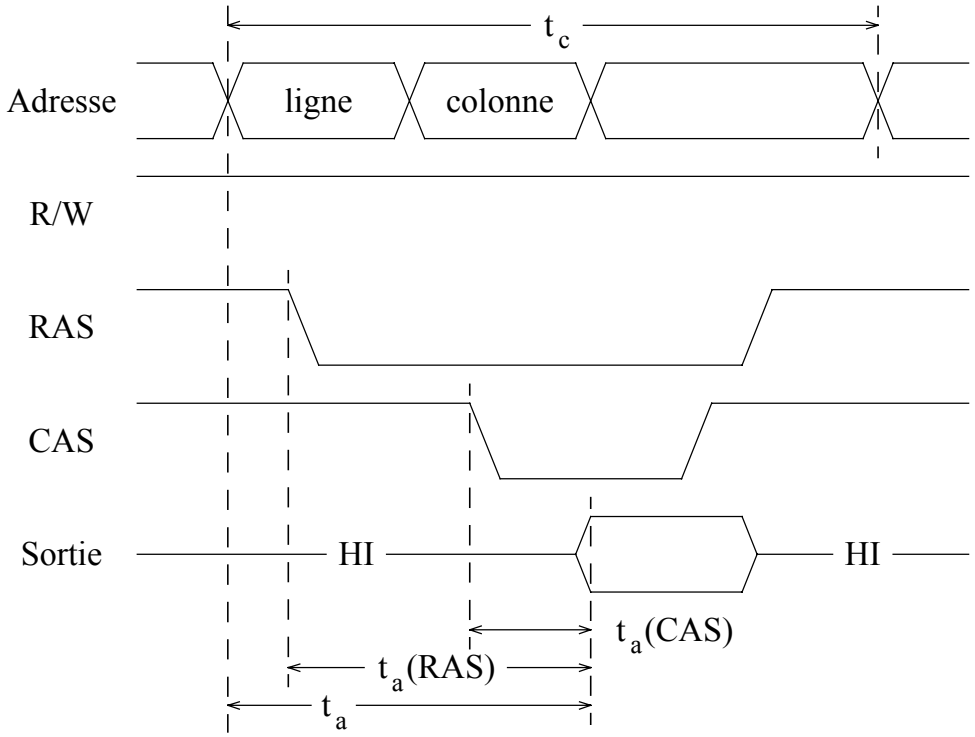


Figure 14

Lorsque les adresses de ligne et de colonne sont multiplexées celles-ci doivent être stabilisées avant les signaux de chargement RAS et CAS respectivement. Le temps d'accès t_a est mesuré à partir du début de l'opération, mais on distingue deux autres temps d'accès mesurés à partir des fronts descendants des signaux RAS et CAS. Dans l'exemple illustré, la ligne R/W à 1 indique une fonction de lecture.

V.6.c Cycle d'écriture

La procédure d'écriture consiste à :

- Etablir l'adresse;
- Sélectionner le ou les boîtiers nécessaires;
- Etablir la donnée sur l'entrée;
- Etablir une impulsion d'écriture.

L'adresse doit être stabilisée avant la sélection et les données doivent être stabilisées avant le signal de chargement.

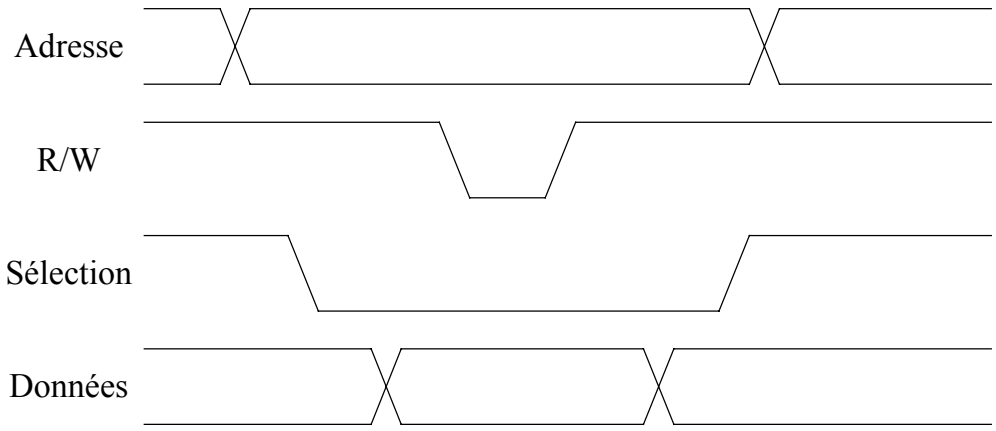


Figure 15

V.6.d Cycle de lecture-modification-écriture

Ce cycle permet de lire une donnée à une certaine adresse, de modifier la donnée et de l'inscrire à la même adresse. Le même résultat pourrait être obtenu avec deux cycles successifs de lecture et d'écriture, mais plus lentement car il faut alors présenter l'adresse deux fois. Il faut que les lignes d'entrée et de sortie soient distinctes. La procédure consiste à :

- Etablir l'adresse;
- Sélectionner le boîtier correspondant;
- Afficher la fonction lecture;
- La donnée apparaît sur la sortie au bout du temps d'accès;
- Le système modifie la donnée qu'il applique sur l'entrée;
- Etablir une impulsion d'écriture.

V.6.e Lecture ou écriture en mode page

Le mode page concerne les mémoires avec multiplexage des adresses ligne et colonne. Au cours de l'exécution d'un programme, l'accès aux informations (instructions ou données) se fait plus souvent à des adresses consécutives qu'à des adresses isolées. L'accès en mode page permet alors d'améliorer les temps d'accès. En effet lorsque l'adresse de la ligne (ou page) reste identique, il n'y a pas lieu de la présenter à nouveau. Dans un cycle en mode page, l'adresse de ligne est chargée une fois, chaque accès est ensuite défini par l'adresse de colonne. Le temps du premier accès est équivalent à celui d'un accès aléatoire. Il est appelé latence. Par contre les accès suivants sont plus rapides. La figure suivante illustre une séquence de trois accès comprenant une écriture intercalée entre deux lectures.

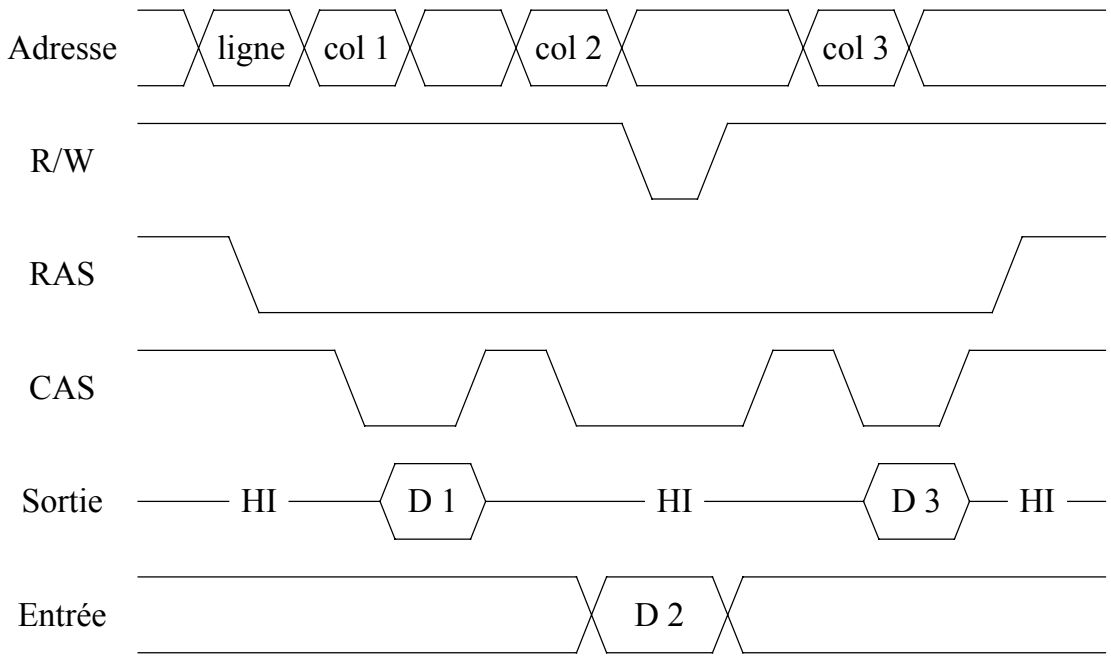


Figure 16

V.6.f Rafraîchissement des mémoires dynamiques

Pour la plupart des mémoires dynamiques les adresses sont multiplexées. A chaque accès, en lecture ou en écriture, chaque bit de la ligne désignée est transféré en bas de colonne dans une bascule. Ce transfert est initié par le front descendant du signal RAS. Les opérations effectives de lecture ou d'écriture se font au niveau de ces registres. Ceux-ci sont ensuite chargés dans la ligne d'origine sur le front montant du signal RAS. En procédant ainsi on rafraîchit la ligne adressée à chaque accès. Cela ne garantit pas un rafraîchissement suffisant de l'ensemble de la mémoire. Il est possible de déclencher le même mécanisme de rafraîchissement d'une ligne sans faire aucun accès. Il suffit de préciser l'adresse de la ligne à rafraîchir puis d'activer une impulsion d'une durée définie sur la ligne RAS. Par ailleurs les mémoires vives contiennent aussi un pointeur interne sur la prochaine ligne à rafraîchir. Il est utilisé en activant la ligne CAS avant la ligne RAS.

Le mécanisme de rafraîchissement des divers blocs mémoires est confié à un contrôleur. C'est également ce contrôleur qui se charge de multiplexer les adresses.

V.7 Mémoire synchrone

Tous les accès étudiés jusqu'à présent sont de type asynchrone. Ils peuvent intervenir à tout instant et leur durée est fixée par les composants de la mémoire.

Ces dernières années le fossé entre la vitesse de fonctionnement des processeurs et la vitesse d'accès des mémoires n'a fait que croître. La mémoire constitue aujourd'hui le goulot d'étranglement des architectures. Les gains en performance sur la vitesse intrinsèque des composants sont relativement faibles. Les fabricants de mémoires jouent sur l'organisation logique des accès. Nous en avons déjà vu un exemple avec le mode page, dont de nombreuses variantes ont été mises en œuvre.

Un autre mode permet de réduire le temps nécessaire à l'adressage des colonnes. Il s'agit du mode d'accès en rafale (burst) qui concerne plusieurs colonnes consécutives. Un compteur interne permet d'éviter le temps nécessaire à l'établissement des signaux d'adresse et à leur chargement. Il suffit donc de préciser initialement l'adresse de départ et le nombre de colonnes. Dans un composant asynchrone l'incrémentation de ce compteur est provoqué par des cycles du signal CAS.

Mais il est également possible de faire appel à un signal d'horloge. Nous obtenons alors une mémoire synchrone (SDRAM pour Synchronous Dynamic Random Acces Memory). Cette solution a pour avantage de simplifier la gestion de la mémoire. Ce qui permet un léger gain de vitesse, mais aussi allège la conception du contrôleur. Il est ainsi possible d'avoir un accès à chaque cycle d'horloge. Par contre le temps d'accès pour une seule donnée n'est pas inférieur à celui d'une mémoire asynchrone. Il peut même être plus important.

Pour parvenir à suivre une fréquence intéressante les fabricants mettent également en œuvre des techniques déjà rencontrées dans ce chapitre. Par exemple, à l'intérieur d'un même circuit la mémoire est organisée en deux blocs (ou plus) qui peuvent être entrelacés. Cela permet également l'ouverture simultanée de deux pages. Il est alors possible de passer d'une page à une autre sans ralentir la vitesse d'accès.

V.8 Gestion de la mémoire centrale

Au début de l'informatique les mémoires étaient chères et de petites tailles. A cette époque les programmeurs passaient l'essentiel de leur temps à optimiser leur programme en fonction de la taille mémoire. On a ensuite pu utiliser une mémoire secondaire : bandes, tambours et disques. Le programmateur divisait alors son programme en un certain nombre de morceaux appelés branches (overlays), chaque branche étant de taille inférieure à celle de la mémoire. Pour exécuter un programme on chargeait la première branche puis la seconde et ainsi de suite. Le programmeur était totalement responsable de la découpe du programme et de la gestion de la migration des branches entre la mémoire principale et la mémoire secondaire.

En 1961 des chercheurs de Manchester ont proposé une méthode qui permettait de rendre transparent au programmeur le mécanisme de gestion des branches. Au début des années 1970 cette technique, dite de mémoire virtuelle, devint disponible sur la plupart des ordinateurs. L'idée

préconisée par le groupe de Manchester consiste à distinguer les notions d'espace d'adressage et d'emplacement mémoire. Ce paragraphe doit être développé dans un cours sur les systèmes d'exploitation. Je n'introduirai ici que les principales notions, pour étudier leur mise en œuvre au niveau du matériel.

V.8.a Unité d'adressage

L'unité d'adressage est la plus petite quantité de mémoire qui sépare deux adresses consécutives. Dans toutes les machines, par convention, on assigne à la première position de mémoire l'adresse 0, à la deuxième l'adresse 1 et à la $N^{ème}$ l'adresse N-1. Une machine peut être adressable par octet (byte) ou par mot de 16, 20, 32, 60, 64, … bits. Dans une machine adressable par octet, un mot de 32 bits est constitué de 4 octets consécutifs. Mais les constructeurs ont adopté deux conventions différentes pour ranger les octets à l'intérieur d'un mot. Dans la première convention l'octet qui contient les 8 bits de poids faible du mot est rangé à l'adresse la plus basse. Il est rangé à l'adresse la plus haute dans le second cas. Ces deux conventions sont parfois référencées par les vocables de big-endian et little-endian respectivement. Aucune de ces deux conventions n'est intrinsèquement meilleure que l'autre. Cela n'a d'importance que lors d'un échange de données entre deux machines respectant des conventions différentes.

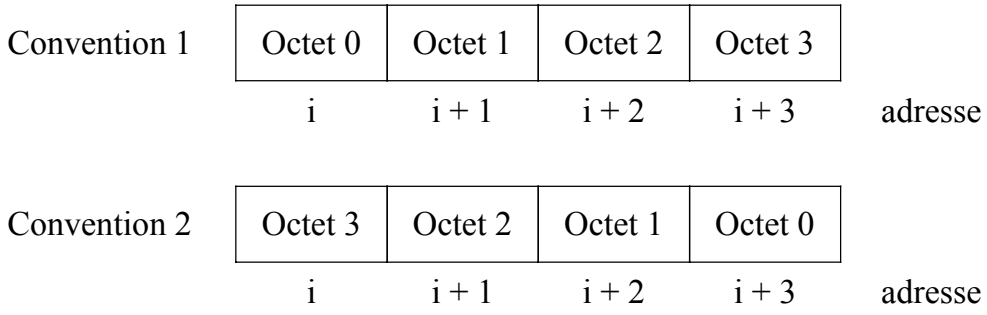


Figure 17

Il peut exister d'autres restrictions sur la façon dont une machine peut manipuler sa mémoire. Il peut par exemple être impossible de faire démarrer un mot de 32 bits à n'importe quelle adresse. L'adresse du premier octet doit être paire, voire même un multiple de 4.

V.8.b Espace d'adressage

L'espace virtuel est le nombre de cellules mémoires qu'un programme peut adresser directement. Ce nombre dépend du nombre de bits utilisés pour manipuler les adresses au niveau des instructions. La mémoire physique disponible sur la machine peut avoir une taille différente de l'espace virtuel disponible pour chaque programme.

L'espace physique peut être plus grand que l'espace virtuel. Dans cette configuration il est possible de charger plusieurs programmes à la fois. Lorsque le contrôle de l'unité centrale passe d'un programme à l'autre, l'exécution peut s'enchaîner rapidement sans attendre le chargement du nouveau programme depuis le disque. Comme l'unité centrale accède à la mémoire par des adresses comptant un plus grand nombre de bits que les adresses manipulées par le programme il faut être capable de "compléter" celles-ci. Cela peut se faire grâce aux registres de base dont nous parlerons plus loin.

L'espace physique peut être plus petit que l'espace virtuel. Dans ce cas il faut disposer d'un mécanisme capable de charger, à un moment donné, les parties du programme nécessaires à son exécution. Ce mécanisme est également utilisé dans des machines pour lequel l'espace physique est plus grand que l'espace virtuel. Il permet en effet un partage plus efficace de la mémoire entre plusieurs programmes concurrents.

V.8.c Principe de la séparation et de l'assemblage des adresses

Le principe de séparation/assemblage des adresses est le fondement de toutes les techniques de gestion de la mémoire centrale. Une adresse peut toujours être vue comme l'assemblage d'un numéro de bloc et d'un déplacement à l'intérieur de ce bloc :

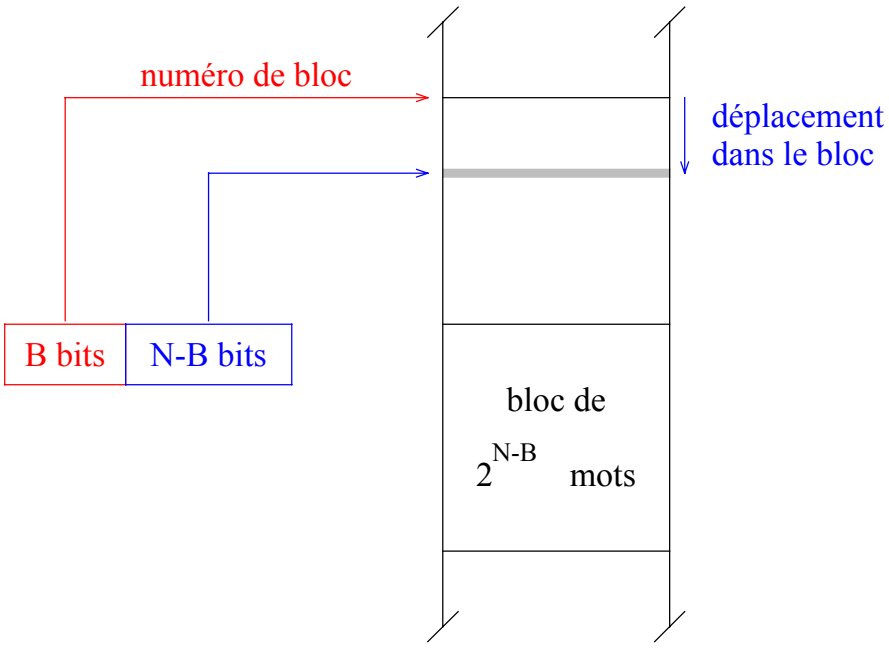


Figure 18

Une adresse de N bits permet de repérer 2^N cases. On peut la séparer en un numéro de bloc, codé sur B bits (les bits de poids fort), et un numéro de case dans le bloc sur N-B bits. Chaque bloc comptera 2^{N-B} cases et il y aura 2^B blocs, donc toujours 2^N cases au total.

Prenons un exemple numérique très simple pour illustrer ce principe. Considérons une mémoire comptant 16 cases, numérotées de 0 à 15. Les adresses permettant d'identifier les mots dans cette mémoire ont 4 bits. Nous nous intéressons à la case d'adresse 13 (marquée **** sur la figure). En binaire nous avons $13 = (1101)_2$. Si nous divisons la mémoire en deux blocs de huit mots, cette case 13 se situe à l'adresse 5 du bloc numéro 1 ($6^{ème}$ mot du $2^{ème}$ bloc). Le numéro de bloc est donné par le bit de haut poids (à gauche) de l'adresse. Le déplacement dans le bloc est donné par les trois bits de droite : $(101)_2 = 5$. Nous pouvons également diviser la mémoire en quatre blocs de quatre mots. Le numéro de bloc est alors donné par les deux bits de gauche $(11)_2 = 3$ et le déplacement par les deux bits de droite $(01)_2 = 1$. La case 13 est en effet à l'adresse 1 dans le bloc 3.

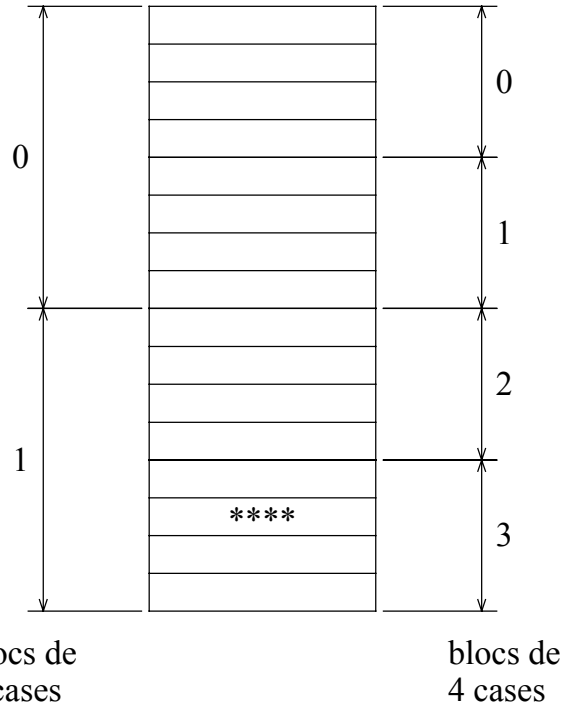


Figure 19

Ce principe est facile à mettre en œuvre au niveau du matériel : les B lignes du bus d'adresse correspondant aux bits de poids fort sont connectées à un registre de B bits et les N – B autres lignes sont commandées par un autre registre de N – B bits. Ces registres peuvent être chargés indépendamment par l'unité de calcul des adresses effectives.

V.8.d Registre de base et pagination

Pour optimiser l'utilisation de tout ordinateur il est rapidement devenu indispensable de pouvoir charger tout programme n'importe où en mémoire. Il faut donc un mécanisme permettant à partir des adresses virtuelles de calculer les adresses physiques. Une des méthodes les plus utilisées est celle des registres de base qui existent dans presque tous les ordinateurs. Le mécanisme est le suivant :

- Les programmes sont "relogeables" et toutes les adresses connues par le programmeur sont virtuelles, relatives au début du programme à l'adresse 0;
- Un registre spécial de l'unité centrale, appelé registre de base, contient l'adresse physique du début du programme dans la mémoire;
- Chaque fois que l'on fait une référence à la mémoire, on ajoute à l'adresse virtuelle trouvée dans le programme le contenu du registre de base.

Un registre de base permet également de construire des adresses physiques comptant un nombre différent de bits que les adresses virtuelles.

Pour obtenir une plus grande souplesse dans la gestion de la mémoire centrale, l'espace virtuel de chaque programme est découpé en blocs, tous de taille identique, les pages. L'espace physique est découpé en blocs (ou pages physiques) de même taille que les pages de l'espace virtuel. La longueur des pages est toujours une puissance de 2 ce qui facilite l'obtention de l'adresse physique par assemblage binaire. Le mécanisme de gestion qui permet de passer d'une adresse virtuelle à une adresse physique dans un système de pagination passe par la table des pages. Les propriétés de la table de pages sont les suivantes :

- Il y a une table des pages pour chaque programme;
- La table des pages contient une entrée pour chaque page virtuelle du programme;
- Le début de la table des pages est chargé dans un registre : le registre de base de la table de pages (RBTP);
- Chaque entrée de la table peut contenir :
  - . des bits de contrôle :
    - page présente en mémoire/page absente;
    - page modifiée/non modifiée;
    - autorisation d'accès en écriture;
    - compteur d'utilisation;
  - . deux adresses :
    - position de la page en mémoire secondaire (en général un disque : unité, face, piste et secteur);
    - numéro du bloc de mémoire physique contenant la page, si la page virtuelle est présente en mémoire.

Le registre de base et le registre de base de la table de pages font partie du contexte du programme. La mise à jour de la table des pages est assurée par le système. L'exploitation de cette table pour le calcul de l'adresse physique peut être effectuée par le système, mais cela consomme une partie du temps CPU disponible. Pour libérer le système d'exploitation cette gestion peut être confiée à un circuit spécialisé : unité de gestion de la mémoire (MMU : Memory Management Unit) intercalée entre l'unité centrale et la mémoire.

Le mécanisme de gestion de la mémoire peut se décomposer de la manière suivante :

1) L'adresse virtuelle est séparée en deux parties :
   - numéro de page virtuelle.
   - déplacement dans la page : identique dans la page physique;

2) On ajoute le contenu du registre RBTP au numéro de page virtuelle obtenu lors de l'étape 1. Cela nous donne l'adresse de l'entrée correspondante dans la table des pages.

3) On teste le bit de présence :
   a) La page est présente. Le numéro du bloc physique contenant la page est chargé dans le registre de base. L'adresse physique est calculée en assemblant le registre de base au déplacement dans la page obtenu en 1 (fig. 20).
   b) La page est absente. Il y a faute de page. L'adresse de la page en mémoire secondaire permet de chercher la page sur disque et de la charger en mémoire. La tâche en incombe au système d'exploitation, qui alloue l'unité centrale à un autre programme pendant ce temps. Après chargement le numéro du bloc physique contenant la page est placé dans la table des pages. On est alors ramené au cas 3a.

Pour accélérer la phase 3.a il est possible d'utiliser une mémoire associative (cf. prochain paragraphe) utilisant le numéro de page virtuelle comme étiquette et fournissant en sortie le numéro de bloc physique. Cette mémoire associative ne contient alors que les pages en mémoire. Si la recherche est infructueuse le système utilise la table des pages.

Pour pouvoir charger une page en mémoire il faut disposer d'un bloc physique libre. Si ce n'est pas le cas il faut décharger une page (en première approximation effacer le bit indiquant que la page est en mémoire et effacer le numéro du bloc physique correspondant). Plusieurs stratégies sont possibles : aléatoire, FIFO ou LRU (cf : Systèmes d'exploitation). Une solution consiste à remplacer la page la moins récemment utilisée (LRU : Least Recently Used). C'est le rôle du compteur associé à chaque entrée de la table des pages. Il indique la plus récente référence à la page. D'autre part, si la page a été modifiée il faut la sauvegarder en mémoire secondaire avant de libérer l'emplacement physique.

Le programme est écrit comme s'il pouvait être totalement contenu dans la mémoire principale, même si ce n'est pas le cas. A la limite, le programmeur n'a pas besoin de savoir qu'il y a une mémoire virtuelle : tout se passe comme si l'ordinateur était doté d'une énorme mémoire principale. Le mécanisme de pagination est transparent, sauf au niveau des performances qui dépendent de la charge du système.

Une autre fonction de l'unité de gestion de la mémoire consiste à vérifier la cohérence des attributs d'accès. Chaque tâche possède des attributs : mode utilisateur, mode superviseur, etc… Il en est de même pour les zones mémoires. On peut avoir :

- Lecture uniquement;
- Superviseur uniquement;

- Zone programme;
- Zone donnée;
- etc…

Si l'accès viole les attributs de la cible, la MMU doit interrompre la requête mémoire et engendrer une condition d'erreur ou une interruption particulière appelée exception.

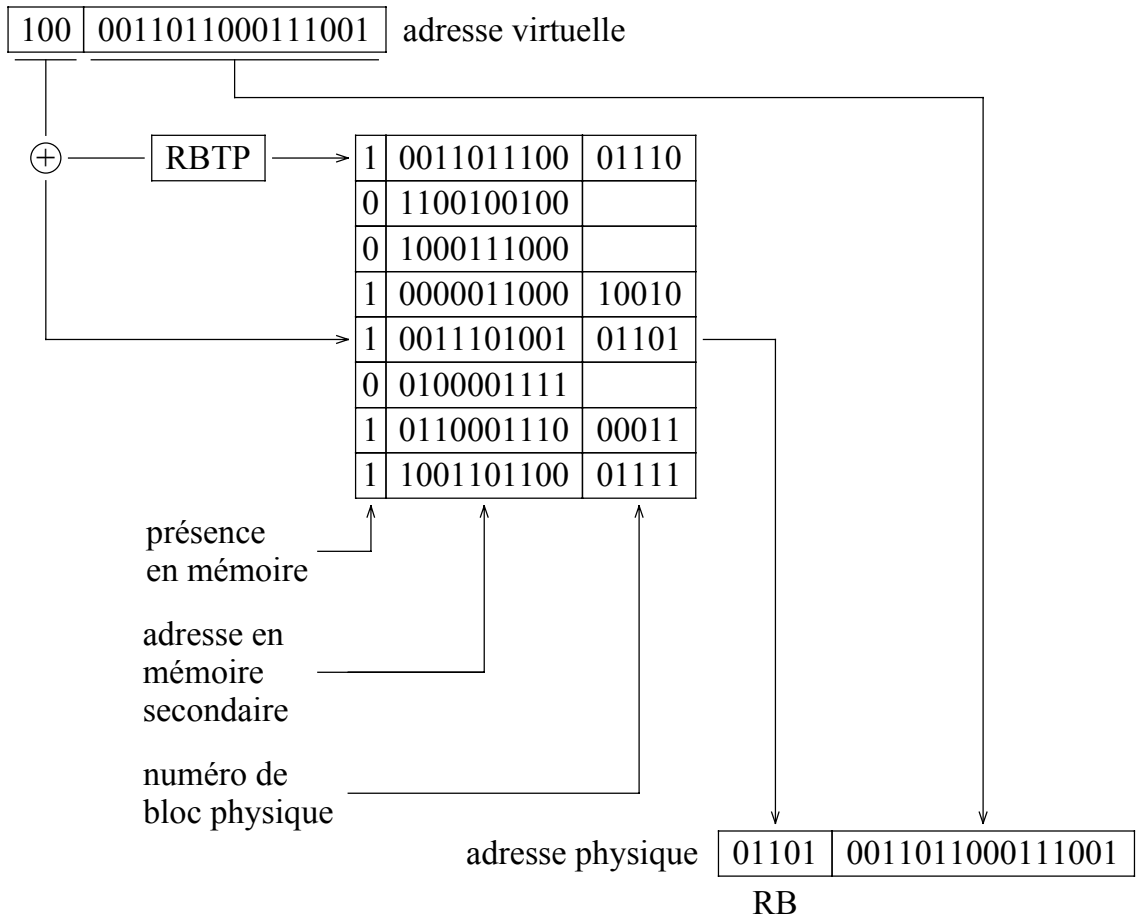


Figure 20

V.9 Mémoire associative

Une mémoire associative, ou mémoire adressable par le contenu, permet de chercher si une information, appelée descripteur ou clef, est contenue dans la mémoire et, dans cette hypothèse de fournir une information associée. Le descripteur est comparé simultanément à tous les mots de la mémoire associative. La figure 21 présente le schéma de principe d'une mémoire associative. La figure 22 explicite les cellules de base des deux matrices. La matrice 2 peut être très semblable à l'organisation des cellules dans une mémoire vive classique. Par contre la matrice 1 remplace le décodage des lignes.

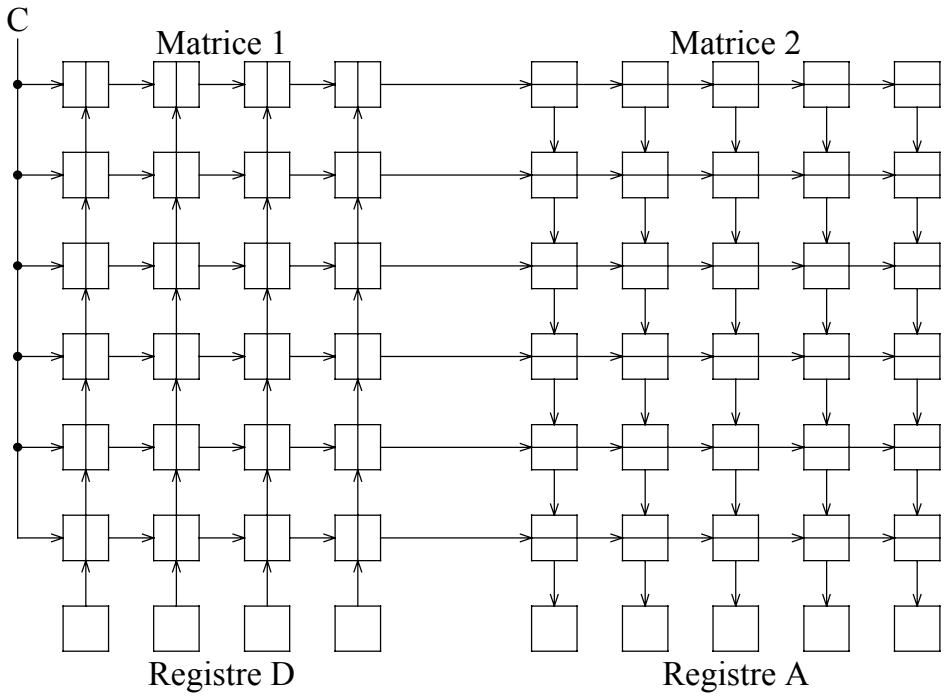


Figure 21

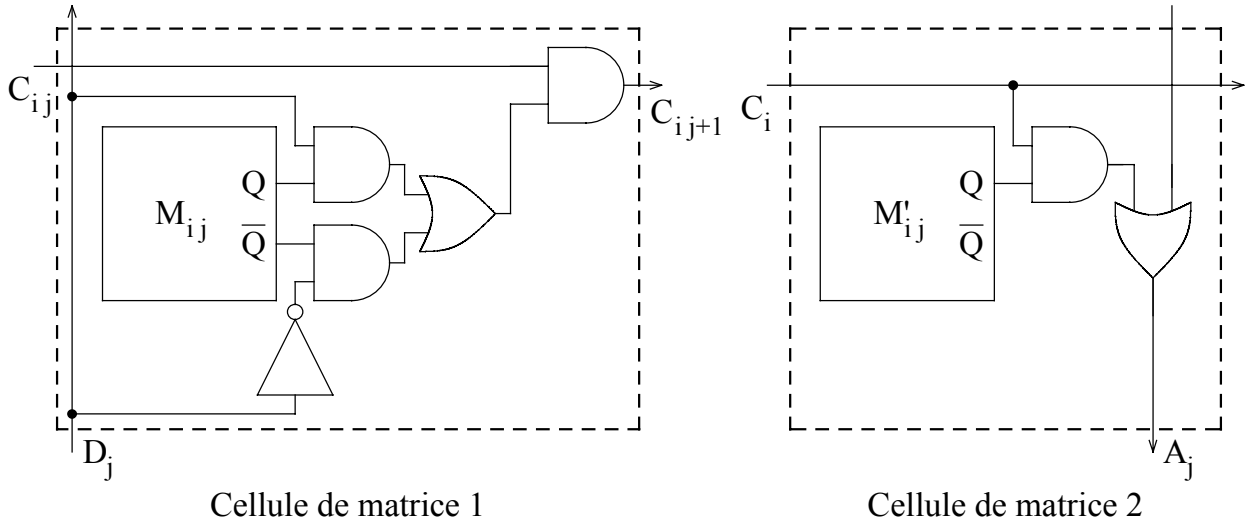


Figure 22

La recherche en mémoire suppose le chargement préalable du descripteur dans le registre D. Ensuite toutes les opérations s'exécutent en parallèle sur tous les mots, sous l'effet d'une seule impulsion C. Une cellule de la matrice 1 transmet l'impulsion C si le contenu du point mémoire est identique au bit correspondant du registre D placé sur la même verticale. Dans le cas contraire

l'impulsion C est arrêtée. En effet à la sortie de la cellule i (et à l'entrée de la cellule i+1) nous avons :

$$C_{i\,j+1} = C_{i\,j} \bullet (D_j \bullet Q_{i\,j} + \overline{D}_j \bullet \overline{Q}_{i\,j})$$

Si on suppose qu'il y a dans la matrice 1 au plus une information identique à celle du descripteur, il sort de la matrice 1 au plus une impulsion C_i qui désigne le mot de la matrice 2 à charger dans le registre A.

Une mémoire associative est souvent utilisée pour la gestion de la mémoire centrale. Les mémoires caches constituent un autre exemple d'utilisation des mémoires associatives.

V.10 Mémoire cache ou antémémoire

Le concept de mémoire virtuelle paginée peut être vu comme un cache, la mémoire physique constituant un cache pour la mémoire secondaire. D'une façon générale, on appelle cache tout dispositif matériel ou logiciel qui stocke dans une zone d'accès rapide une copie de données en petite quantité choisies parmi des données qui sont stockées dans une zone d'accès plus lent.

Dans le cas de la pagination, la mémoire primaire est la mémoire physique et la mémoire secondaire est l'espace sur disque. La mémoire physique ne contient, à un instant donné, que les seules pages du programme utilisées récemment. Dans le cas d'un cache-mémoire, la mémoire primaire est une mémoire rapide, la mémoire secondaire est la mémoire principale de la machine. Le cache-mémoire ne contient, à un instant donné, que les copies de petits blocs de mots de la mémoire physique récemment référencés. Il y a cependant une nuance. Pour la pagination le temps nécessaire au chargement d'une page à partir du disque est long. Ce mécanisme est géré par le système et le programme en cours perd le contrôle de l'unité centrale. Pour l'antémémoire la gestion se fait uniquement au niveau du matériel et le programme en cours conserve la main.

On utilise pour les caches des mémoires statiques très rapides. La mémoire cache permet de combler en partie la différence entre les vitesses de l'unité centrale et de la mémoire. On trouve très souvent deux niveaux de mémoire cache. Le premier niveau est intégré au processeur et dispose d'un temps d'accès équivalent au cycle de fonctionnement de celui-ci. Le second niveau est de plus grande taille mais plus lent. Une mémoire cache intégrée au processeur, ou très proche, est basée sur une mémoire associative.

Les statistiques ont montré que 80 à 90 % des programmes se trouvent dans des boucles. C'est pourquoi une mémoire cache peut obtenir un taux de réussite, probabilité de trouver l'instruction dans le cache, de 70 à 90 %. L'emploi des mémoires caches est essentiel dans les architectures RISC. Certains processeurs disposent de deux caches pour les instructions et les données (architecture de Harvard).

Même en dehors d'une boucle, comme le traitement des instructions est séquentiel on a intérêt à transférer les instructions par blocs. On appelle un tel bloc une ligne, équivalente à la page entre mémoire et disque.

La figure 23 illustre le schéma de principe d'une architecture avec hiérarchisation de la mémoire. L'intégration sur une même plaquette silicium s'étend peu à peu. Elle peut aller aujourd'hui jusqu'au cache de niveau 2.

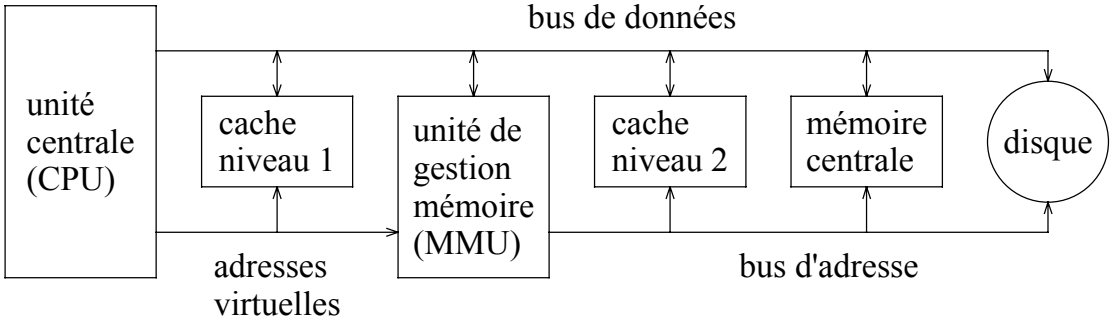


Figure 23

V.11 Mémoire file ou FIFO

Il existe un autre type de mémoire vive : la file d'attente ou FIFO. Cette abréviation signifie First In First Out, soit premier entré premier sorti.

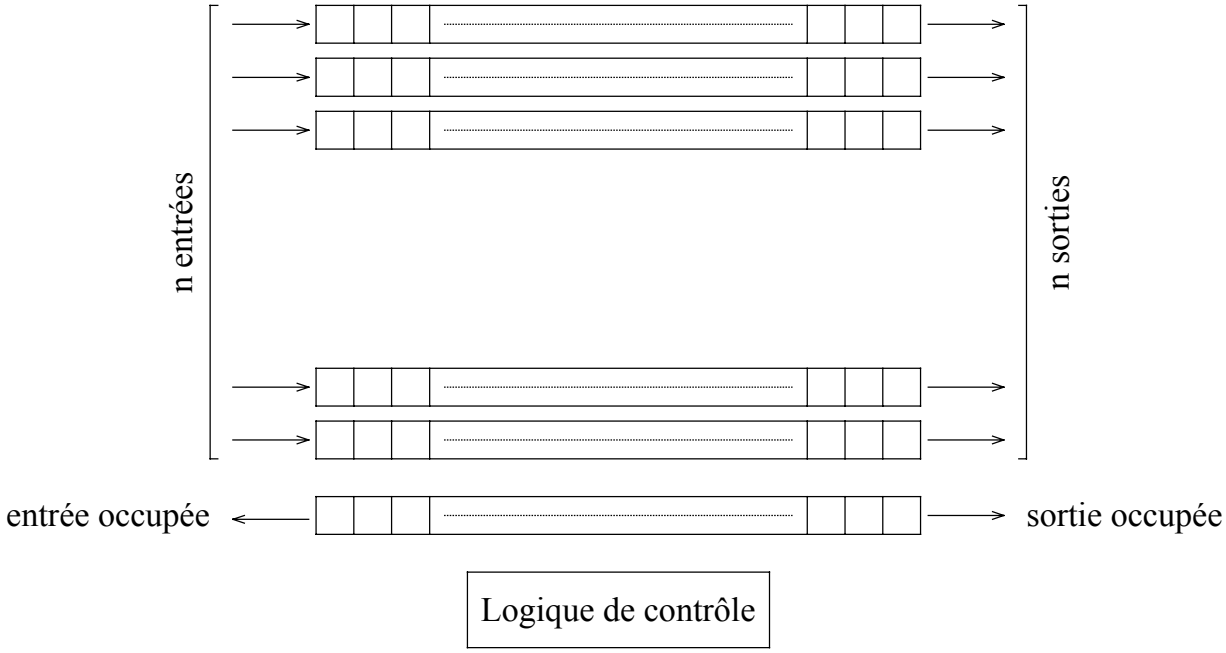


Figure 24

Une file dispose de deux bus de données distincts en entrée et en sortie. L'accès n'est pas aléatoire, l'ordre en sortie est identique à celui en entrée. Il n'y a pas d'adressage. L'utilisateur peut écrire dans la FIFO si le premier étage est libre et lire la FIFO que si le dernier étage est occupé. Une file est constituée de n registres à décalage comptant chacun m cases. Le nombre n correspond à la largeur des mots, alors que m est la profondeur de la file ou sa capacité. Un registre interne indique l'état (libre ou occupé) de chacun des étages. La logique de contrôle décale automatiquement chaque étage occupé vers un étage libre.