LES MEMOIRES

Dans ce chapitre, nous abordons la constitution des mémoires.

Les fils électriques représentés dans les différents schémas représentent une donnée numérique, qui n'a comme durée de vie que le temps pendant lequel les tensions pour cette opération sont générées : lors de l'opération suivante, la valeur obtenue sera perdue, car remplacée par la nouvelle valeur. Nous aurons donc besoin de stocker les valeurs obtenues, afin de s'en "souvenir" le moment venu. Aussi aurons-nous besoin de "mémoire".
Lorsqu'on conduit une voiture, notre mémoire fait intervenir deux types de souvenirs :

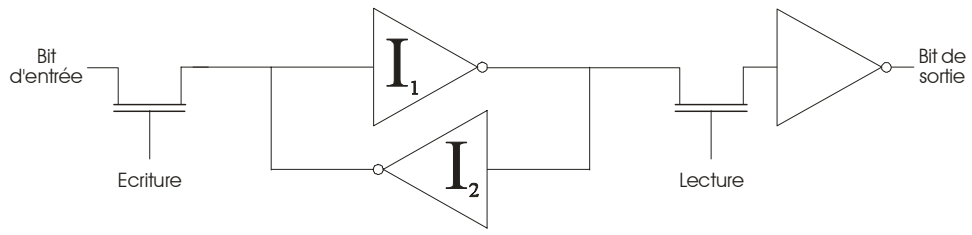
- ceux liés au trajet à effectuer : la vitesse limite de la portion de route, la position de la voiture qui nous suit, la distance à la prochaine station-service, le rapport de vitesse enclenché, etc… Ces informations sont en constant renouvellement, non seulement d'un trajet à l'autre, mais également en cours de route. De plus, elles sont oubliées dès que le trajet prend fin. Ce type d'information est stocké dans un ordinateur en *mémoire vive*.
- ceux qui sont indépendants du trajet, et liés à la conduite à proprement parler : comment passer une vitesse, la signification des panneaux, le code de la route, etc... Ces informations seront utilisées quel que soit le trajet, et constituent le "mode d'emploi" du conducteur. Ce type d'information est stocké dans un ordinateur en *mémoire morte*.

La mémoire vive

La mémoire vive est un circuit capable de :

- se mettre dans l'état (0 ou 1) de son entrée lorsqu'on le lui demande,
- ne pas modifier son état dans les autres moments, quel que soit l'état de son entrée,
- transférer son état en sortie lorsqu'on le lui demande.

Il faut alors un circuit qui entretienne lui-même son entrée pour maintenir sa sortie, même sans sollicitation extérieure. Voyez le circuit suivant :

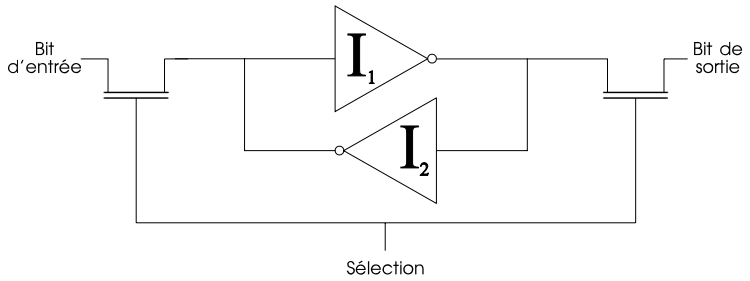


Ainsi :

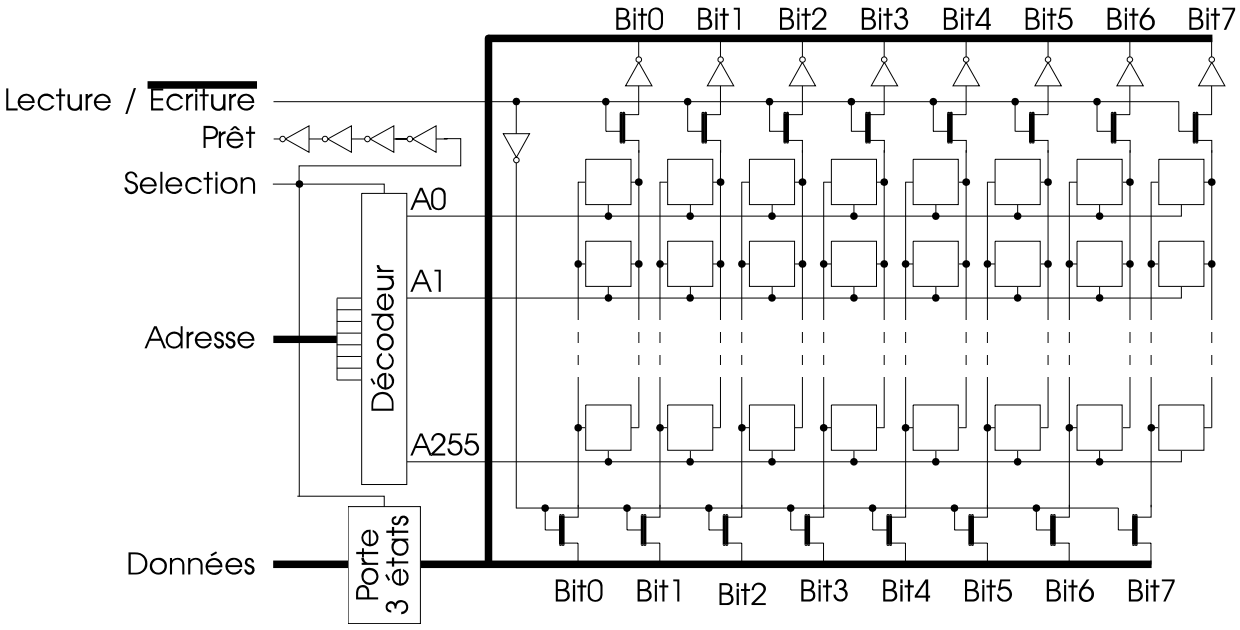
- Quand "écriture" est à 1, la valeur du bit d'entrée est transmise à I_1, qui transmet l'inverse à I_2, qui alimente I_1 de la valeur du bit d'entrée, et ainsi de suite. Ainsi, si "écriture" revient à 0, le transistor devient bloqué : c'est un interrupteur ouvert. Les valeurs d'entrée et de sortie de I_1 et I_2 sont conservées, quelle que soit la valeur du bit d'entrée.
- Il suffit de mettre "lecture" à 1 pour récupérer la valeur du bit d'entrée.
- Il suffit de mettre "écriture" à 1 pour réactualiser l'état de la mémoire en fonction de la valeur du bit d'entrée. Vu la structure d'un inverseur, il n'est pas gênant de forcer la sortie de I_2.

Le schéma ne représente pas les alimentations en courant, nécessaires au fonctionnement des transistors des inverseurs, et qui fournissent l'énergie pour le maintien de la mémoire.

Afin d'avoir une mémoire de plusieurs octets, nous utiliserons un montage très légèrement différent :



Ce montage constitue une cellule mémoire à deux entrées et une sortie, dont on peut assembler un grand nombre (1 carré = une telle cellule mémoire) pour réaliser notre mémoire à plusieurs octets (256 par exemple) :



La modification légère que nous avons apportée à notre cellule de mémoire n'avait pour but que de déporter les transistors de *Lecture / ~~Ecriture~~*, et de pouvoir sélectionner d'un seul coup les 8 cellules du même octet.
Le signal *Prêt* a pour but, lors d'une opération de lecture, de signaler le moment où les données attendues sont effectivement sur le bus de données (et lors d'une opération d'écriture, le moment où les données sont effectivement enregistrées), c'est-à-dire le moment où tous les transistors (notamment ceux câblés en cascade) ont commuté. En effet, lorsque les signaux *Sélection* et *Lecture* sont à 1, le contenu de la mémoire à l'adresse spécifiée n'est pas immédiatement sur le bus de données : un transistor met toujours un certain temps pour commuter de 0 à 1 ou de 1 à 0. Ici, le signal le plus long passe par le décodeur, puis par une cellule-mémoire, puis par un transistor commandé par le signal *Lecture*, puis par l'inverseur, puis par la porte 3 états. Le nombre d'inverseurs (nécessairement pair) générant le signal *Prêt* est défini en fonction du retard qu'il faut imposer à la génération de ce signal (le schéma en indique 4 pour exemple, mais c'est insuffisant face au signal le plus long).

De manière générale, un signal noté X signifie qu'il produit l'action quand il est à 1, un signal noté ~~X~~ produit l'action quand il est à 0. Le signal *Lecture / ~~Ecriture~~* (souvent noté R/~~W~~ pour Read / Write) signifie que la lecture est active lorsque le signal est à 1, et que l'écriture est active lorsque le signal est à 0.
Le signal *Sélection* s'appelle souvent ~~CS~~ (Chip Select), ou ~~CE~~ (Chip Enable). Comme l'indique sa notation barrée, il est souvent inversé : la mémoire est activée lorsque le signal est à 0.
Le signal *Prêt* s'appelle souvent RDY (ReaDY).
Le schéma pédagogique ci-dessus réalise une mémoire de 256 octets. Néanmoins, les mémoires sont généralement des mémoires de bits, qu'on assemble en parallèle pour former une mémoire d'octets (chaque mémoire de bits étant consacrée à un des bits à mémoriser).

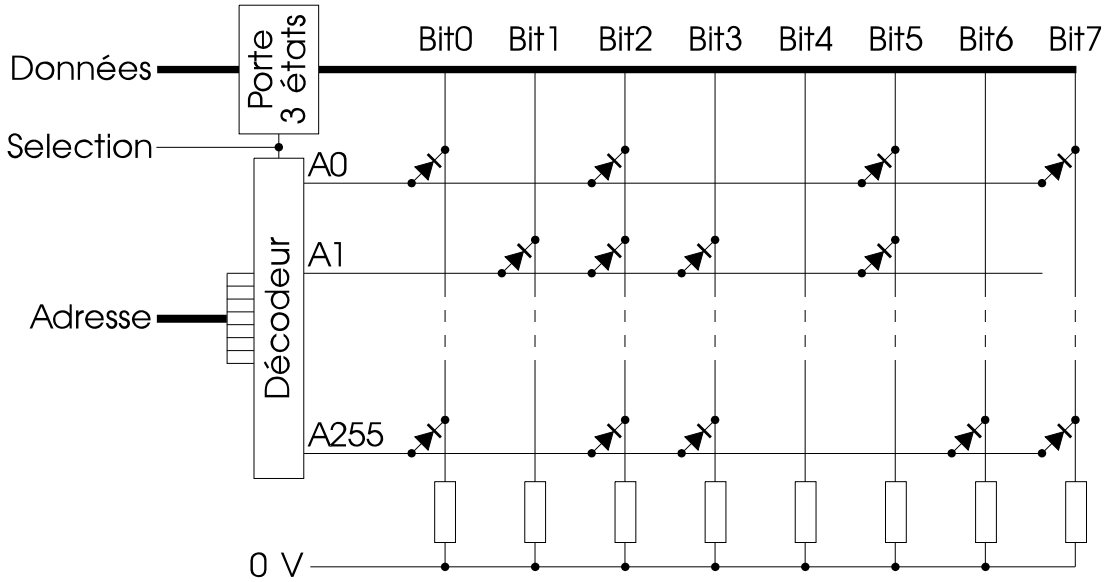
La mémoire présentée ici est une mémoire dite *statique*, car son état ne varie pas dans le temps si aucune action n'est entreprise. De plus, elle a des temps de réponse rapides. Ce type de mémoire onéreux est utilisé pour la mémoire inscrite au sein du microprocesseur (mais de plus en plus aussi pour la mémoire RAM), directement en

relation avec l'UAL, dont on appelle les octets des registres. Aussi, afin de disposer de plus grandes quantités de mémoire, des composants distincts du microprocesseur assurent la même fonction de mémoire pour un coût moindre (mais également une moindre rapidité). Ils sont dotés de mémoire *dynamique* : son principe est de charger ou non un condensateur (qui remplace les deux inverseurs de la cellule-mémoire statique) pour stocker un bit (plutôt que de saturer ou non un transistor). L'inconvénient est qu'un condensateur présente toujours un courant de fuite, et se décharge donc. Il est alors nécessaire d'utiliser un circuit auxiliaire qui va scruter chaque condensateur, pour le recharger s'il contient un 1, avant qu'on n'y voie un faux 0. Ce traitement est appelé *rafraîchissement*. Aussi, lorsqu'on parle communément et sans autre précision de la *mémoire vive* d'un micro-ordinateur (ou *MEV* pour MEmoire Vive, ou *RAM* pour Random Access Memory), il s'agit parfois de cette mémoire de "plus grande" capacité, séparée du microprocesseur (les composants prenant place sur des barrettes amovibles, dites *barrettes SIMM*, pour les PC) contrairement aux registres.

L'inconvénient de la mémoire vive (registres ou RAM) est que les informations qu'elle contient sont perdues si on coupe son alimentation. Il faut donc également une mémoire "gravée dans la pierre" pour avoir dès la mise sous tension un "grimoire" qui contient le mode d'emploi de l'ordinateur. C'est le rôle assuré par la mémoire morte.

Mémoire morte

La mémoire morte étant immuable une fois qu'elle a été construite, il n'est pas question d'aller y écrire par la suite.
Après la mémoire vive, la constitution d'une mémoire morte est un jeu d'enfant : il suffit, dans le circuit de la mémoire vive, que chaque fil d'adresse soit connecté (par une diode) ou non à chaque fil de donnée (selon la valeur que le constructeur a voulu inscrire dans ce bit à cette adresse), ce dernier étant raccordé à la masse via une résistance. En pratique :



La mémoire morte est appelée *MEM* (MEmoire Morte), ou *ROM* (Read Only Memory). Elle comporte également un signal *Prêt* (non figuré ici).
Nous verrons plus tard les autres types de mémoire : mémoire cache, mémoire de masse, mémoire virtuelle, mémoire d'éléphant…

Taille des mémoires

L'unité de mesure de la taille d'une mémoire est généralement l'octet (plus rarement le bit). Aussi, les mémoires actuelles offrant de grandes capacités, on est amener à utiliser des multiples de l'octet : le ko (kilo-octet), le Mo (méga-octet), le Go (giga-octet), le To (téra-octet)...
La signification de ces multiples n'est pas tout à fait identique au sens commun. En effet, la taille des mémoires est directement fonction du nombre de bits permettant leur adressage. Une mémoire 64 Ko nécessite 16 bits pour son adressage. Or, 16 bits permettent d'adresser 65536 octets (2^{16}), qu'il serait dommage de ne pas utiliser puisqu'on a tout le matériel. La valeur du kilo n'est donc pas exactement 1 000, mais $65536/64 = 2^{10} = 1\,024$.

De la même manière :

- le méga $= 2^{20} = 1\ 048\ 576$
- le giga $= 2^{30} = 1\ 073\ 741\ 824$
- le téra $= 2^{40} = 1\ 099\ 511\ 627\ 776$ (environ !).