

## 1. Présentation d'un système standard à microprocesseur.

### 1.1. Introduction.

La donnée fondamentale sur laquelle travaille un ordinateur est le bit. Les états 0 et 1 peuvent être associés à une absence ou une présence de tension.

### 1.2. Le microprocesseur.

Le processeur est à la base de toutes les opérations arithmétiques et logiques sur les données binaires.

#### 1.2.1. Ses éléments internes.

- Horloge : rythme la cadence de travail. Plus la fréquence est grande, plus le  $\mu P$  effectuera d'instruction à la seconde (MIPS : millions d'instructions par seconde).
- Unité arithmétique et logique (UAL) : responsables des calculs sur les données. Une des caractéristiques est la taille de la donnée sur laquelle les calculs vont porter. En fonction de la taille, on parlera de  $\mu P$  16, 32 ou 64 bits.
- Registres internes : lorsque le  $\mu P$  traite les données, il stocke les données temporairement dans des registres internes. Voici les plus importants :
  - *Registre accumulateur* : stocke les données et les résultats des opérations arithmétiques et logiques.
  - *Registre d'état* : contient les indicateurs sur le résultat de la dernière opération exécutée (nombre positif, négatif, nul, ...).
  - *Registre ordinal* ou *programm counter* : contient l'adresse de la prochaine instruction à traiter.
- Bus externes : la taille du bus de données est exprimée en bits. C'est la taille maximale de la donnée que le  $\mu P$  peut lire en un seul cycle dans la mémoire.

#### 1.2.2. Les bus externes.

Bus : ensemble des pistes servant à faire transiter des informations binaires de même nature. L'échange de la donnée entre la mémoire et le  $\mu P$  se fait par le bus de données. Sa taille est en général égale à la taille de la donnée que l'UAL peut traiter en un seul cycle. Ce n'est pas toujours le cas :

- Taille du bus plus importante pour augmenter la vitesse de transfert de données.
- Taille du bus de donnée plus petite pour diminuer le coût.

Lorsque la mémoire traite une donnée, elle doit savoir s'il s'agit d'une lecture ou d'une écriture. Le  $\mu P$  fournit alors un signal read/write par le bus de contrôle ou de bus de commande.

#### 1.2.3. L'architecture.

##### 1.2.3.1 Le pipelining.

Pipelining : ensemble d'unités effectuant un travail à la chaîne pour traiter des instructions.

Lorsqu'on augmente la profondeur du pipeline, on augmente le nombre d'unités et on réduit donc le travail que chacune doit effectuer. Le pipeline doit être rempli de façon optimale. Une instruction de saut ou l'attente d'une donnée arrête le fonctionnement du pipeline. L'organisation se fait par l'unité de contrôle du processeur et le compilateur.

Un processeur scalaire possède plusieurs unités d'exécution. Il peut alors appliquer la même opération à un ensemble de données variables. On parle d'architecture SIMD (single instruction multiple data).

À partir du 486, on a des  $\mu P$  avec une cadence d'horloge interne, cadencant les échanges entre unités, différente de la cadence permettant d'assurer les échanges sur le bus extérieur. Il faut alors utiliser une unité d'interface de bus permettant d'adapter les différences de cadence.

##### 1.2.3.2 L'exécution dynamique.

L'exécution dynamique repose sur l'utilisation simultanée de :

- Prédiction de branchement : consiste à deviner l'emplacement de la prochaine instruction à exécuter et à la charger dans le pipeline. En effet, un saut oblige le processeur à recharger ses pipelines et le programme perd l'accélération due au pipeline.
- Analyse de flux : permet au processeur de modifier l'ordre d'entrée des instructions pour optimiser le taux d'occupation.

#### 1.2.3.3 L'architecture CISC.

Traduction : « ordinateur à jeu d'instructions complexes ». Ces processeurs peuvent exécutées des instructions complexes qui sont directement câblées sur leurs circuits électroniques pour gagner en rapidité d'exécution sur ces instructions.

L'inconvénient est le prix à la fabrication (plus de fonctions imprimées sur le silicium). De plus, les instructions peuvent parfois prendre plus d'un cycle d'horloge, ce qui les rend lentes à l'exécution car un processeur basé sur l'architecture RISC ne peut traiter qu'une instruction à la fois.

#### 1.2.3.4 L'architecture RISC.

Traduction : « ordinateur à jeu d'instruction réduit ». Pas d'instructions supplémentaires directement câblées. Donc, il faut utiliser des programmes ayant des instructions simples. Cela se traduit par une programmation plus difficile et un compilateur plus puissant.

Les quelques instructions qui ne peuvent pas être décrites en instructions simples sont quand même directement imprimées sans alourdir la fabrication.

L'avantage est le coût réduit et, les instructions étant simples, elles sont exécutées en un seul cycle d'horloge. De tels processeurs sont capables de traiter plusieurs instructions simultanément en parallèle.

CCL : la diminution de la complexité matérielle est compensée par un compilateur très évolué.

#### 1.2.4. Les interruptions matérielles.

Deux cas de figures :

- Méthode scrutation : le  $\mu P$  scrute à intervalle régulier l'ensemble des périphériques par l'intermédiaire du registre d'état. Simple d'un point de vue matériel mais si aucun périphérique n'est demandeur, c'est un gaspillage de temps.
- Le périphérique demandeur envoie une demande vers le  $\mu P$  qui interrompt son travail en terminant l'instruction en cours et exécute le code en fonction du périphérique. C'est plus complexe d'un point de vue matériel car elle nécessite des bornes sur le  $\mu P$  permettant aux périphériques d'envoyer des demandes, ce sont les bornes d'interruption. Deux types de bornes :
  - Borne d'interruption masquable, désactivable par programmation et identifié par INT.
  - Borne d'interruption non masquable, identifié par NMI.

Il y a une gestion des priorités sur un circuit interne (PIC).

#### 1.2.5. Les sockets.

Socket : type de support du  $\mu P$ .

Avec les 486, apparaît le ZIF (Zéro Insertion Force).

En fonction de la taille du bus, de la cadence d'horloge et parfois des tensions utilisées, les sockets évoluent.

Analyse des différents Pentium :

- Fréquence du bus externe n'a pas beaucoup augmenté (100M -> 133M).
- Taille du bus de données a augmenté, + utilisation Quad Pumped.
- Fréquence du bus interne a beaucoup augmenté (60M -> 1,5G et +).
- Intégration meilleure.

#### 1.3. Le contrôleur d'interruptions.

### 1.3.1. Introduction.

Le  $\mu$ P possède une ou plusieurs bornes d'interruption qui permettent au périphérique de demander un service au  $\mu$ P. Lorsque le  $\mu$ P reçoit un tel signal, il termine son instruction en cours et se branche sur du code appelé 'routine d'interruption' (ISR : Interrupt Routine Service).

On trouve plus de périphériques demandeur que d'entrées sur le  $\mu$ P, c'est pour cela que se trouve sur la carte mère un contrôleur d'interruption effectuant une gestion centralisée des demandes. De plus, cette gestion comprend une gestion des priorités.

### 1.3.2. Interruption et architecture Intel.

Exception : appelée quand une erreur se produit.

Vecteurs d'interruptions : contient l'adresse mémoire à laquelle se trouve le code qui doit être exécuté.

Les  $\mu$ P de type Intel x86 utilisent 256 interruptions, la plupart étant logicielles. Ces  $\mu$ P ont une table des vecteurs d'interruptions qui comprend les adresses de toutes les routines de service d'interruption (codées sur 4 octets -> table = 1024 octets).

Les ordinateurs jusqu'au 286 étaient équipés d'un seul Pic permettant la gestion de 7 interruptions matérielles, l'IRQ2 étant déjà réservé pour la mise en cascade d'un deuxième contrôleur. Le 2<sup>ème</sup> Pic fut intégré sur le type AT.

### 1.3.3. Le PIC.

Deux modes de gestion des priorités :

- Rotation automatique : lorsque le périphérique a été servi, il reçoit la priorité la plus basse. Au pire, il attendra 7 'tours'.
- Rotation spécifique : le programmeur peut changer l'ordre des priorités.

Dans les Pc, c'est la 2<sup>ème</sup> solution qui est choisie. Ce mode est programmé lors de l'initialisation de la carte mère par le BIOS. L'IRQ7 reçoit la priorité la plus basse et l'IRQ0 la plus haute (+ petite valeur => + grande priorité).

Le Pic dispose de registres internes pour la gestion des priorités. Voici les plus importants :

- IRR : Interrupt Request Register.
- IMR : Interrupt Mask Register.
- ISR : In Service Register.

Procédé d'une demande d'interruption par un périphérique :

- Les entrées passent au travers du registre de masquage qui détermine si l'entrée est masquée ou pas.
- Si elle est masquée, la demande sur cette ligne n'est pas prise en compte.
- Si elle n'est pas masquée, la demande est enregistrée dans le registre de demande d'interruption jusqu'à ce qu'elle soit servie (en fonction des priorités).
- Le Pic envoie une demande INT au  $\mu$ P en activant la borne INT du  $\mu$ P.
- Le  $\mu$ P termine son instruction en cours et accepte la demande en activant sa borne INTA.
- Le Pic positionne alors le bit correspondant à l'interruption dans le registre ISR à '1' et le bit correspondant dans le registre IRR à '0'.
- Le  $\mu$ P envoie un deuxième signal INTA indiquant au Pic qu'il doit déposer sur le bus de données le numéro de l'interruption logicielle qui doit être servie (trois bits de poids faible = numéro de l'interruption matérielle ; cinq bits de poids fort = valeur préprogrammée lors de l'initialisation du Pic).
- La routine d'interruption envoie un signal EOI (End Of Interrupt) au Pic qui remplace le bit correspondant à l'interruption servie à '0'.

### 1.3.4. Partage et interruptions.

Les périphériques étant de plus en plus nombreux, Windows permet une gestion logicielle des interruptions. Si plusieurs périphériques se partagent la même interruption, on peut passer en revue leur registre d'état de façon logicielle. Le bus d'extension PCI permet un partage des interruptions de façon matérielle.

#### 1.4. Le contrôleur de DMA.

##### 1.4.1. Introduction.

Sur une carte mère, des périphériques doivent transférer des données que le  $\mu$ P doit traiter, ces données sont placées en mémoire selon deux méthodes :

- I/O programmé (PIO).
- Accès direct à la mémoire (DMA).

##### 1.4.2. Entrées/sorties programmées.

Cette méthode se réfère à l'utilisation d'instructions d'entrées/sorties qui seront exécutées par le  $\mu$ P pour effectuer le transfert des données entre la mémoire et les registres du périphérique.

Avantage : simplicité de la mise en œuvre. Le seul matériel supplémentaire nécessaire est souvent un circuit de 'Wait State' pour ralentir le  $\mu$ P lors de ses accès ou pour le synchroniser avec le périphérique.

Inconvénient : le  $\mu$ P est occupé pendant toute la durée du transfert pour une tâche relativement simple.

##### 1.4.3. L'accès direct en mémoire.

Pour effectuer un transfert, le périphérique doit utiliser les bus du  $\mu$ P. Le contrôle du bus est assuré par un circuit appelé contrôleur de DMA (DMAC). Ce circuit peut effectuer toutes les opérations nécessaires pour le transfert en un seul cycle d'horloge.

DMAC travaille selon 2 modes :

- Par vol de cycle : le DAMC prend le contrôle des bus pour chaque octet transféré et rend le contrôle au  $\mu$ P.
- En rafale : des blocs de données sont transférés avant de rendre le contrôle des bus au  $\mu$ P.

Le choix dépend de la rapidité à laquelle les données sont fournies par le périphérique ET du fait qu'une application peut ou non être déconnectée du  $\mu$ P pendant la durée du transfert.

##### 1.4.4. DMA sur les PC XT.

Les slots d'extensions ISA ne permettent des transferts de données que de 8 bits.

Différentes phases caractérisant un transfert DMA :

- Le périphérique active la borne DREQn (n = numéro du canal utilisé) du DMAC.
- Le DMAC active la borne HOLD (maintenue jusqu'à ce que le transfert soit terminé) du  $\mu$ P.
- Le  $\mu$ P termine son instruction et active sa borne HOLDA pour indiquer au DMAC qu'il met ses bus en haute impédance.
- Le DMAC dépose l'adresse mémoire sur le bus d'adresse, active MEMR (Memory Read) et IOW (Input/Output Write) ou MEMW (Memory Write) et IOR (Input/Output Read) et enfin, active le signal d'acquiescement de DMA (DACKn) vers le périphérique.
- Le périphérique répond au DACKn en lisant ou écrivant la donnée sur le bus de donnée.

##### 1.4.5. DMA sur les PC AT.

*Bus mastering* = *DMA first party* : transfert de disque en mode PIO ou transfert DMA.

Avec les Pc AT, apparaît un 2<sup>ème</sup> DMAC en cascade sur le 1<sup>er</sup>. Les Pc AT sont équipés d'un connecteur additionnel pour porte les slots d'extensions à 16 bits. La vitesse maximale de travail de ce type de DMAC est liée à la cadence d'horloge du 1<sup>er</sup> Pc XT.

Avec les Pc AT est aussi apparu un nouveau contrôleur de disque dur. Avant, il se trouvait sur la carte mère. Pour les Pc AT, il se trouve sur le disque dur et le câble le reliant à la carte mère est considéré comme une extension du bus système, on parle alors de bus ATA ou d'IDE (Integred Drive Electronics).

L'utilisation d'un canal DMA bridé pour des raisons de compatibilité ne fait plus l'affaire car les vitesses de rotation de disque augmentent. Les ordinateurs ont maintenant des disques dont les transferts se font en mode PIO ou transfert DMA 'first party' sur le bus PCI = bus mastering.

1.4.6. Le bus mastering ou DMA 'first party'.

L'augmentation du débit de transfert s'explique par une diminution du temps de cycle nécessaire pour effectuer un transfert. L'utilisation de 2 flancs d'horloge pour effectuer un transfert peut aussi augmenter le débit. Le passage aux normes UDMA66 et UDMA 100 s'est effectué en augmentant la cadence d'horloge.

Attention : un cycle de transfert n'est pas nécessairement égal à un cycle d'horloge.

2. **Classifications des mémoires.**

*L'explication des mémoires par les schémas des transistors : voir syllabus pages 16 -> 20.*

2.1. Introduction.

Volatilité d'une mémoire : conservation ou non des données si l'alimentation disparaît.

Les mémoires peuvent être classées en fonction de la volatilité des données et du fait qu'il faut ou non une cadence d'horloge pour la synchroniser.

2.2. Les mémoires asynchrones.

2.2.1. Les mémoires mortes.

2.2.1.1. Les mémoires ROM.

Read-Only Memory : mémoire en lecture seule.

Ces mémoires sont non volatiles. Elles ne sont pas programmables directement par l'utilisateur final, les données y sont placées lors de la fabrication.

2.2.1.2. Les mémoires PROM.

Ces mémoires sont non volatiles, elles sont en lecture seule et ne sont programmables qu'UNE seule fois. Lors de la fabrication, des diodes sont placées à toutes les intersections, la programmation consiste à détruire les diodes aux endroits voulus. Ces mémoires sont également appelées OTP (One Time Programmable).

2.2.1.3. Les mémoires EPROM.

Ce sont elles aussi des mémoires non volatiles mais elles sont effaçables et programmables. Elles se basent sur l'utilisation de transistors FAMOS (pages 17 et 18). Pour effacer une telle mémoire, on soumet la petite fenêtre de quartz présente sur la mémoire à des UV. La programmation nécessite un niveau de tension qu'on ne retrouve pas sur les cartes mères, il faut donc l'enlever de son support (ce qui est un inconvénient). Le nombre de cycles effacement/programmation est 1000.

2.2.1.4. Les mémoires EEPROM ou E<sup>2</sup>PROM.

L'effacement d'une telle mémoire est électrique par l'effet tunnel. On a baissé le niveau de tension pour éviter les inconvénients d'une EPROM.

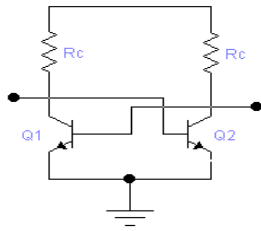
Les mémoires FLASH sont des E<sup>2</sup>PROM.

2.2.2. Les mémoires vives (mémoires volatiles).

Lorsque la tension disparaît, ces mémoires perdent leurs données.

2.2.2.1. Les mémoires statiques (SRAM).

Ce sont des mémoires volatiles accessibles en lecture/écriture.



Principe :  
Si le courant base/émetteur est nul  
=> le transistor est bloqué.

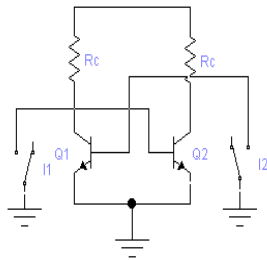
Si le courant base/émetteur est différent de 0 => le transistor est passant.

Exemple : Si Q1 est passant, alors  $V_C$  de Q1 = 0 => courant base/émetteur de Q2 = 0 et Q2 est donc bloqué. C'est donc un état de fonctionnement stable. Grâce à la symétrie, on peut dire :

Q2 ON => Q1 OFF  
Q1 ON => Q2 OFF

Comment passer d'un état stable à un autre ?

On ajoute des interrupteurs en parallèle sur chaque transistor.



Etat de départ : Q1 ON => Q2 OFF.

On ferme l'interrupteur en parallèle sur le transistor bloqué,  $V_C$  de Q2 = 0 => courant base/émetteur de Q1 = 0 => Q1 devient bloqué => courant base/émetteur de Q2 est différent de 0 => Q2 est passant.

On associe à chaque état stable une valeur logique.

#### 2.2.2.2. Les mémoires dynamiques (DRAM).

Temps d'accès : temps qui sépare la réception par la mémoire du signal RAS et la fourniture de la donnée.

Mémoire volatile dont le fonctionnement repose sur la capacité parasite grille/substrat d'un transistor MOS. Cette cellule est confrontée à la perte de charge de la capacité de stockage.

On peut voir cette perte sous deux aspects :

- Une capacité n'est pas parfaite et va se décharger naturellement plus ou moins rapidement.
- Lorsqu'on va accéder à la cellule mémoire, la capacité  $C_S$  va se décharger dans la capacité de colonne  $C_C$  et comme  $C_C$  sera plus grande que  $C_S$ , la charge de  $C_S$  va s'en trouver réduite.

Un rafraîchissement est nécessaire toutes les 2 ms. Les mémoires DRAM se présentent sous un tableau de cellules. Cette table est adressée par un décodeur de lignes et un décodeur de colonnes. Pour gagner de la place, le bus d'adresse est multiplexé c-à-d les mêmes serviront soit pour recevoir l'adresse de la ligne ou l'adresse de la colonne. Chaque adresse sera validée par les deux bornes RAS (Row Address Strobe) et CAS (Column Address Strobe).

Accès type à cette mémoire :

- L'adresse de la ligne est fournie et le signal RAS est activé. L'activation provoque le chargement de la ligne dans les amplificateurs sensitifs et la mémorisation de l'adresse dans un buffer.
- L'adresse de la colonne est fournie et lorsque le signal CAS est activé, l'amplificateur sensitif fournira sa donnée sur le buffer de sortie de la mémoire.

En mode FPM (Fast Page Mode), lorsqu'on a fourni l'adresse d'une ligne, on peut, pour les accès successifs suivants, ne fournir que l'adresse de la colonne.

#### 2.3. Les mémoires synchrones.

Cela ne concerne que les mémoires vives.

### 3. Etude de la structure d'un disque dur.

#### 3.1. Structure physique.

Tout disque dur est constitué de plateaux, divisés en pistes.

Plateaux ( 0 -> N = la tête de lecture) > pistes ( 0 (+extérieure) -> N) >secteurs (1 -> N) = 512 octets.

L'ensemble des pistes sur l'ensemble des plateaux est appelé cylindre. C'est le système de représentation CHS (Cylindre, Head, Secteur). Une autre représentation : LBA (Logic Block Array) : numérotation linéaire des secteurs.

*Capacité d'un disque* = #cylindres x #secteurs x #têtes x 512 octets.

Le nombre de secteurs par piste et la vitesse influent sur le débit de transfert des données.

*Débit* = #secteurs sur une piste x 512 octets x 7200 tours / 60 secondes.

#### 3.2. La gestion du système de fichiers.

##### 3.2.1. Introduction.

La table d'allocation de fichiers et le répertoire racine servent à retrouver l'ensemble des secteurs sur lesquels une donnée est enregistrée.

##### 3.2.2. La table d'allocation de fichiers.

*Cluster* : regroupement logique d'un certain nombre de secteurs.

*FAT* : table présente sur les disques durs et contenant les numéros des clusters qui sont occupés par des données.

Comme un secteur contient 512 octets, on adaptera le nombre de secteurs par cluster en fonction de la taille du disque. Un cluster partiellement occupé est considéré comme totalement utilisé donc on disposera jamais plus de 64 secteurs par cluster.

Le numéro qui suit le nom FAT sur le nombre de bit sur lequel est codé le numéro de cluster.

Exemple : FAT12 : permet de numérotter au plus 4096 clusters ( $2^{12}$ ).

Avec Windows 95, apparaît la FAT32. Pour des disques supérieurs à 512Mo, Microsoft recommande la FAT32.

##### 3.2.3. Le répertoire racine.

La FAT contient les informations sur les clusters occupés. Le répertoire racine contient, lui, les informations sur les fichiers qui sont stockés sur le disque dur (nom du fichier, extension, attribut, date et heure, taille et le numéro du premier cluster occupé). La taille du répertoire racine ne change pas en fonction du nombre de fichiers stockés. Pour la FAT16, on a 512 entrées (une fois qu'elles sont toutes utilisées, un message d'erreur dit : « disque plein »).