

1) Veillez décrire les éléments internes et externes dans l'architecture d'un microprocesseur standard.

Le micro processeur qui est à la base de toutes les opérations arithmétique et logique du micro-ordinateur est composé de :

- Une horloge interne qui cadence son travail, plus la fréquence est élevée plus le microprocesseur effectue d'opération par secondes
- Une unité arithmétique et logique (ALU) qui sera responsable des calculs sur les données. Une caractéristique importante est la taille des données que l'ALU traite en un cycle d'horloge, en fonction de cette taille on parlera de microprocesseur 16 bits, 32 bits ou 64 bits.
- Les registres internes Les registres internes. Lorsque le microprocesseur traite les données (lorsqu'il exécute des instructions), le microprocesseur stocke temporairement les données dans de petites mémoires internes que l'on appelle les registres. Parmi les registres les plus importants, on retrouvera:
 - le registre accumulateur qui stocke les données et les résultats des opérations arithmétiques et logiques.
 - le registre d'état contenant des indicateurs sur le résultat de la dernière instruction exécutée: nombre positif, négatif, nul...
 - le registre ordinal encore appelé PC (programm counter). Ce registre contient l'adresse de la prochaine instruction à traiter.
- Les bus externe, le microprocesseur a souvent besoin d'aller chercher des données en mémoire et en déposer pour les utiliser plus tard. Pour que les données puissent transiter ainsi entre la mémoire et le microprocesseur la carte mère possède des piste de cuivre, on appel c'est piste les BUS. Généralement au nombre de trois, un bus d'adresse : afin d'accéder à une cellule mémoire le processeur doit fournir une adresse sur le bus d'adresse. La taille du bus d'adresse va donc définir l'espace mémoire adressable et va donc nous renseigner sur la capacité maximale avec laquelle le processeur nous permettra de travailler (Ex un bus de 16 bits nous permet $2^{16} = 65\ 536$ adresse différentes ce qui fait 64 Ko) Les données transiteront à travers le bus de données qui sera généralement égale à la taille de l'ALU (si on a eu ALU sur 32 bits alors généralement le bus de données sera de 32 bits lui aussi, remarque pour des raisons philosophique on pourrait retrouver un bus de données différent de la taille de l'ALU. Enfin pour renseigner si il veut effectuer une opération de lecture ou d'écriture en mémoire le microprocesseur envois un signal de type read/write pour se faire il possède une bus de commande encore appeler bus de contrôle

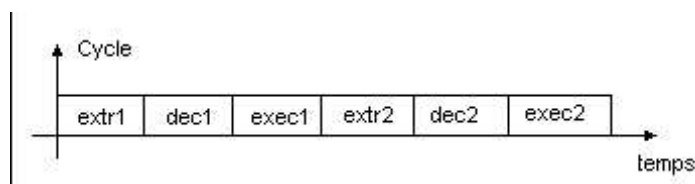
2. veillez décrire les avantages du pipelining dans l'architecture interne des microprocesseurs.

Un pipelining est un ensemble d'unités effectuant un travail à la chaîne dans le but final de traiter les instructions. Lorsque l'on augmente la profondeur du pipeline, on augmente le nombre d'unités, et on réduit donc le travail que chacune doit effectuer.

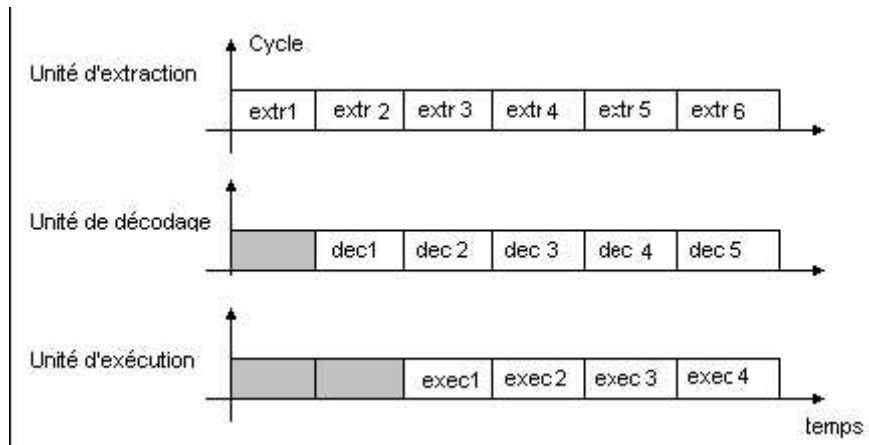
Pour bien comprendre l'utilité de cette architecture, considérant un processeur hypothétique composé de 3 unités :

- l'unité d'extraction qui est responsable de la lecture en mémoire de l'instruction à exécuter.
- l'unité de décodage qui est responsable du décodage de l'instruction lue par l'unité précédente.
- l'unité d'exécution.

Pour un processeur simple, nous obtenons le chronogramme suivant :



Pour un processeur disposant d'une architecture pipeline à 3 unités, nous aurons le chronogramme suivant :



On s'aperçoit que sur un laps de temps identique, un processeur simple a exécuté 2 instructions tandis qu'un processeur disposant d'une architecture pipeline cadencé avec la même horloge en a déjà exécuté 4. On trouve autour de cette architecture la possibilité de travailler avec une cadence d'horloge interne, cadencant les échanges entre unités, différente de la cadence permettant d'assurer les échanges sur le bus extérieur.

3) Lorsque l'on analyse les caractéristiques des différents processeurs Pentium, quelles sont les conclusions que l'on peut tirer sur leur évolution technologique.

Lorsque l'on analyse les caractéristiques des différents processeurs Pentium, on peut en tirer les conclusions suivantes:

1. La fréquence du bus externe n'a que très peu augmenté. Elle était de 100Mhz pour le Pentium I, elle est de 100Mhz pour les premiers Pentium IV pour atteindre maintenant 200MHz pour les dernières versions (266Mhz pour le pentium 4 Extrem Edition). Les principales évolutions sont l'augmentation de la taille du bus de données permettant des débits plus importants et le fait que dans un transfert en rafale entre le processeur et la mémoire, on utilise soit les deux flancs d'horloge pour ainsi doubler le débit soit un artifice de chez Intel appelé Quad pumped et permettant avec la même horloge de quadrupler le débit (4 mots de 64bits) .

2. La fréquence du bus interne a fortement augmenté. D'un Pentium I cadencé à 60Mhz, on arrive à un Pentium IV cadencé à 3.8Ghz. Cette augmentation de la fréquence se traduit par une diminution de la tension d'alimentation. Lorsque l'on s'intéresse aux circuits logiques, on constate qu'à fréquence élevée, une grande partie de la puissance est dissipée par les capacités parasites; on retrouve dans le calcul de la puissance la formule suivante: $C_p \times V_{cc} \times 2 \times \text{Freq}$. Si l'on désire limiter la puissance dissipée tout en augmentant la fréquence, on peut agir sur la tension d'alimentation.

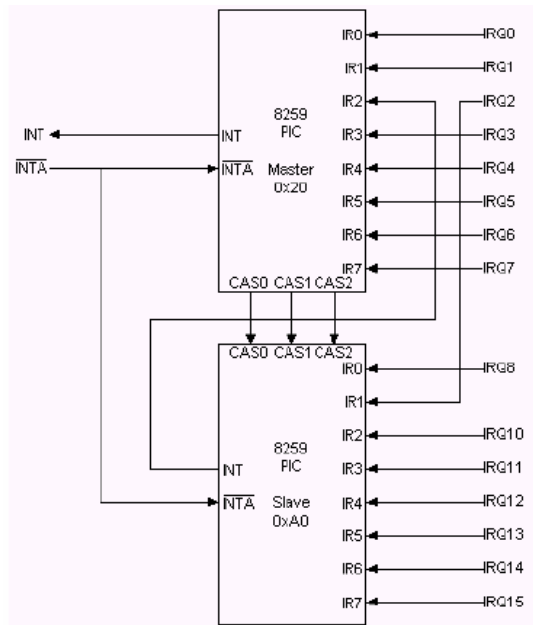
3. L'intégration. Avec ses 42 millions de transistors, le Pentium IV occupe une surface de 217 mm² contre 170 mm² pour le Pentium III. A titre d'information, le 80386 comprenait 275000 transistors pour une surface identique à celle du Pentium IV. La technologie de gravure était de 1.5μ pour le 80386 alors qu'elle est de 0.13μ pour le Pentium IV. Les derniers pentium sortis sur le marché ont actuellement une gravure à 0.09μ.

Question intégration, on atteint 55 Millions de transistors pour une gravure à 0.13 μ et pour le dernier né, 125 millions de transistors pour une surface de 112 mm² dont l'analyse de la structure interne laisse apparaître des zones du processeur non activées dans la version actuelle du processeur. L'arrivée du 0.065μ chez Intel est prévue pour fin 2005 / début 2006.

4. Sur base des registres internes du contrôleur PIC, veuillez décrire à partir d'un exemple commenté, la gestion des priorités des interruptions tant au niveau du PIC que du microprocesseur.

PIC : Programmable Interrupt Controller.

Le microprocesseur dispose d'une ou plusieurs bornes d'interruption. Ces bornes permettent à tout périphérique de demander un 'service' au microprocesseur : lorsque celui-ci reçoit un tel signal, il termine son instruction en cours et se branche sur une partie de code appelée 'routine d'interruption' (ISR : Interrupt Service Routine) permettant de servir le périphérique demandeur. On retrouve généralement plus de périphériques potentiellement demandeurs que d'entrées sur le microprocesseur. C'est la raison pour laquelle on retrouve sur les cartes mère un contrôleur d'interruption permettant d'effectuer une gestion centralisée des demandes avant de les transmettre vers le microprocesseur.



Le composant prévoit deux modes de gestion des priorités:

1. **La rotation automatique:** dans ce mode, lorsque un périphérique a été servi, il reçoit la priorité la plus basse de façon à ce que, dans le pire des cas, un périphérique demandeur n'ait qu'à attendre que les sept autres périphériques aient été servis au plus une seule fois.
2. **La rotation spécifique:** dans ce mode, le programmeur peut changer l'entrée affectée à la priorité la plus basse, les autres priorités étant fixées en fonction de cette dernière.

Dans les PC, c'est la deuxième solution qui a été retenue. Ce mode est programmé lors de l'initialisation de votre carte mère par le BIOS. C'est l'entrée IRQ7 qui reçoit la priorité la plus basse et IRQ0 la priorité la plus élevée. Le fait que le second contrôleur soit mis en cascade sur l'entrée IRQ2 du premier contrôleur, nous aurons dans l'ordre décroissant de priorité les entrées suivantes: IRQ0, IRQ1, IRQ8, IRQ9, IRQ10, IRQ11, IRQ12, IRQ13, IRQ14, IRQ15, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7.

Le PIC dispose de registres internes permettant la gestion des interruptions. Nous retiendrons uniquement les registres suivants:

IRR: (Interrupt request Register)

IMR: (Interrupt Mask Register)

ISR: (In Service Register)

Les huit lignes d'entrée passent au travers du registre de masquage qui va déterminer si une entrée est masquée ou pas. Si elle est masquée, la demande arrivant sur cette ligne ne sera pas prise en compte. Si elle n'est pas masquée, la demande sera enregistrée dans le registre de demande d'interruption qui la maintiendra jusqu'au moment où elle sera servie. En fonction des priorités programmées, le PIC pourra déterminer de quelle interruption il doit s'occuper. Il envoie donc une demande INT vers le processeur en activant la borne INT de ce dernier. Le microprocesseur termine son instruction en cours et va renseigner au PIC qu'il accepte la demande en activant sa borne INTA (Interrupt Acknowledge).

Dès la réception de ce signal, le PIC va positionner le bit correspondant à l'interruption servie dans le registre ISR à '1' et le bit correspondant dans le registre IRR à '0'.

Le microprocesseur va alors envoyer 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. Ce numéro est construit de la façon suivante par le PIC: les trois bits de poids faible correspondent au numéro de l'interruption matérielle tandis que les cinq

bits de poids fort correspondent à une valeur préprogrammée lors de l'initialisation du PIC. Pour les PC, le PIC maître reçoit la valeur 00001 et le PIC esclave la valeur 01110. Si l'on prend l'IRQ matérielle 3 (PIC maître IRQ 3), cela correspondra donc à 00001011 (0B) tandis que si l'on prend l'IRQ matérielle 10 (PIC esclave IRQ 2), cela correspondra à 01110010 (72).

La routine d'interruption doit prévoir l'envoi d'un signal (EOI) vers le PIC qui remplacera le bit correspondant à l'interruption servie à 0.

Analysons maintenant la gestion des priorités:

Partons d'une condition initiale où l'interruption matérielle 4 est servie. Nous retrouverons alors les données suivantes dans les différents registres:

IR	IR	IR	IR	IR	IR	IR	IR
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	0	0	0

Supposons que deux demandes d'interruptions se présentent sur les entrées IR6 et IR2 du PIC. L'IRQ6 étant moins prioritaire que l'interruption 4 en cours de traitement, la demande restera en attente. L'IRQ2 étant plus prioritaire que l'interruption 4 en cours de traitement, elle sera prise en compte par le PIC.

Celui ci enverra donc une demande INT vers le microprocesseur qui répondra par la succession de deux signaux INTA. Les registres contiendront alors les données suivantes:

IR	IR	IR	IR	IR	IR	IR	IR
7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	1	0	0

Il est important de signaler que la routine de l'IRQ4 en cours d'exécution a été interrompue au bénéfice de la routine de l'IRQ2. Lorsque cette routine se termine, elle envoie un EOI (End Of Interrupt) vers le PIC. Celui ci, en analysant son registre ISS sait qu'il s'agit d'un signal de fin d'interruption pour l'IRQ2. Voici le contenu des registres:

IR	IR	IR	IR	IR	IR	IR	IR
7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	1	0	0	0	0

Le processeur ayant terminé l'exécution de la routine de l'IRQ2, il revient à l'exécution du code avant interruption: il s'agissait de la routine de l'IRQ4. Lorsque cette routine se termine, elle envoie un EOI (End Of Interrupt) vers le PIC. Celui ci, en analysant son registre ISS sait qu'il s'agit d'un signal de fin d'interruption pour l'IRQ4. Voici le contenu des registres:

IR	IR	IR	IR	IR	IR	IR	IR
7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	0	0	0	0	0	0	0

Le PIC peut alors prendre en compte le demande de l'IRQ6 puisqu'il n'y a plus aucune autre demande plus prioritaire en cours de traitement.

5) Décrire le principe de fonctionnement de l'IOAPIC (priorités, priorités de tâche, multiprocesseurs, sources d'interruptions)

Tandis que le contrôleur d'interruption standard est conçu pour une utilisation dans des systèmes mono processeur, le contrôleur IOAPIC peut être utilisé indifféremment dans les systèmes mono et multi processeurs. Pour une question de compatibilité, les deux contrôleurs se retrouvent sur la carte mère, les interruptions pouvant être contrôlées par le PIC standard ou par l'IOAPIC dépendant de la façon dont les contrôleurs ont été programmés. Ce sera de la responsabilité du programmeur que le même signal d'interruption ne soit pas pris en charge par les deux contrôleurs. Cette possibilité nous laisse entrevoir l'installation d'anciens systèmes d'exploitation tels que Windows98 sur des cartes mères récentes qui ne gèrera que le PIC tandis que des systèmes d'exploitation tels que Windows 2000 ou XP prendront en charge sur la même carte mère le contrôleur avancé.

Au niveau système, l'APIC est constitué de deux parties : l'une résidant dans le microprocesseur et appelée Local APIC tandis que l'autre se trouve dans le sous système IO et est appelée IOAPIC. Ces deux parties communiquent à travers un bus dédié appelé bus APIC.

Bien que le microprocesseur soit capable de pouvoir gérer 256 sources d'interruption, la partie IO ne prévoit au niveau matériel que la gestion de 24 interruptions matérielles. A chacune de ces 24 entrées est associé un registre de 64 bits permettant de spécifier la polarité du signal (actif à l'état haut ou à l'état bas), la sensibilité du signal sur un flanc ou sur un niveau, la destination et le mode dans lequel l'interruption sera délivrée. Ces registres se trouvent dans une table de redirection et sont utilisés pour translater l'interruption correspondante dans un message inter APIC qui sera délivré sur le bus APIC.

Les priorités: pour les interruptions qui sont délivrées au processeur à travers l'APIC local, chaque interruption a une priorité liée à son numéro de vecteur. L'APIC local utilise cette priorité pour déterminer quand servir cette demande en rapport avec les autres activités du processeur incluant notamment le service des autres interruptions. Pour les vecteurs d'interruption dans la gamme comprise entre 16 et 255, la priorité de l'interruption est déterminée en utilisant la relation suivante :

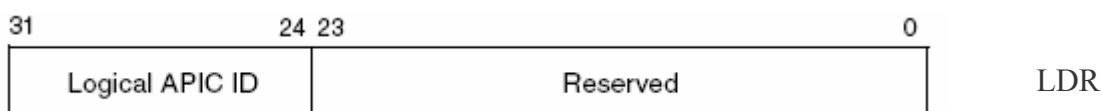
Priorité = vecteur / 16. Le quotient est arrondi à la valeur entière avec la valeur 0 comme priorité la plus basse et 15 la plus élevée. Du fait que les vecteurs 0 à 31 sont réservés pour les exceptions, les priorités des interruptions définies par l'utilisateur s'étendent de 2 à 15.

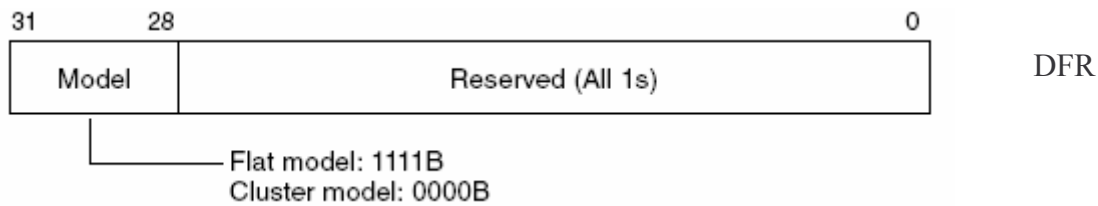
Dans chacun de ces 16 groupes, les quatre bits de poids faible du vecteur seront utilisés pour classer la priorité des interruptions dans un même groupe.

La priorité de tâche: c'est une valeur comprise entre 0 et 15 présente dans le registre de priorité de tâche (TPR) de l'APIC. Cette priorité est configurable par logiciel et permet de définir un seuil de priorité pour les interruptions du processeur. Le processeur servira seulement les interruptions qui ont une priorité plus élevée que celle spécifiée dans le registre TPR : la valeur 0 permettra d'autoriser toutes les interruptions tandis que la valeur 15 les interdira toutes, exceptées les interruptions de type NMI, SMI, INIT, ExtINT.

L'IOAPIC et le multi processeurs. Lorsqu'une interruption arrive sur une des bornes de l'IOAPIC, celui-ci peut transmettre cette demande par le bus APIC vers un processeur spécifique, vers un groupe de processeur ou vers l'ensemble de tous les processeurs. Cette possibilité de configuration est possible grâce aux 64 bits de chaque registre associé à chacune des entrées.

Nous retrouverons pour l'IOAPIC, le champ de destination et le mode de destination. Si le mode de destination est le mode physique, alors le champ de destination contiendra l'identificateur APIC (4 bits) qui permettra de sélectionner un seul composant sur le bus APIC. Si le mode de destination est le mode logique, alors le champ de destination (8 bits) définira un jeu de processeurs, chaque APIC local des processeurs utilisant ses registres DFR (Destination Format Register) et LDR (Logical Destination Register) pour déterminer si il fait partie ou pas du jeu.





Dans le modèle Flat, chaque APIC local effectue un ET logique entre l'identificateur logique APIC et l'adresse de destination du message envoyé par l'IOAPIC. Si une condition logique est vraie, alors l'APIC local va accepter le message. Un broadcast vers tous les APIC locaux peut être effectué en positionnant tous les bits de l'adresse de destination à 1.

Dans le modèle Flat Cluster, l'identificateur logique et l'adresse de destination sont décomposés en deux groupes : les quatre bits de poids fort permettant de coder le numéro du cluster et les quatre bits de poids faible permettant de représenter quatre APIC locaux dans le même cluster, un par bit.

6. Veuillez décrire ce qu'est la DMA et les différentes phases caractérisant un transfert.

DMA : Direct Memory Access.

Sur une carte mère, bon nombre de périphériques doivent transférer des données que le microprocesseur doit ensuite traiter : prenons à titre d'exemple le disque dur, le lecteur de disquette ou la carte réseau. Ces données, avant d'être traitées, sont placées en mémoire visant une des méthodes suivantes : I/O programmé (PIO) ou accès direct à la mémoire (DMA).

On peut aisément comprendre qu'il serait intéressant que le microprocesseur puisse effectuer d'autres tâches alors que d'une façon indépendante le périphérique transférerait ses données vers la mémoire. Pour effectuer un tel transfert, le périphérique doit utiliser les bus du microprocesseur. Le contrôle du bus est assuré pour les périphériques par un circuit supplémentaire que l'on appelle contrôleur de DMA ou encore DMAC. Ce circuit peut effectuer toutes les opérations nécessaires pour le transfert des données (incrémenter de l'adresse mémoire, décrémenter du compteur de transfert, lecture sur le périphérique (INPUT), écriture en mémoire) en un seul cycle de bus. Donc le bus n'est seulement occupé que pendant un cycle pour le transfert d'un octet et le microprocesseur peut continuer à exécuter ses codes.

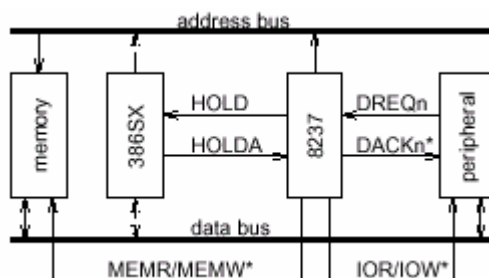
Le contrôleur de DMA peut travailler suivant deux modes :

- mode par vol de cycle dans lequel le contrôleur prend le contrôle des bus pour chaque octet transféré et ensuite rend le contrôle des bus au microprocesseur.
- mode en rafale dans lequel des blocs de données sont transférés avant de rendre le contrôle des bus au microprocesseur.

Le choix dépend de la rapidité avec laquelle les données sont fournies par le périphérique en regard de la bande passante du bus et du fait qu'une application particulière puisse accepter que le microprocesseur soit déconnecté de ses bus pendant la durée du transfert.

Les transferts peuvent être effectués en une étape ou en deux étapes. Dans un transfert en deux étapes, la première étape consiste en la lecture des données sur un port d'un périphérique lors d'un cycle et la deuxième étape en l'écriture de la donnée vers la mémoire lors d'un deuxième cycle. Il est aussi possible pour le contrôleur d'effectuer cette opération simultanément. C'est le mode de fonctionnement utilisé sur les PC.

Le circuit prévoit des priorités sur les entrées de sorte qu'un seul transfert puisse être actif à la fois.



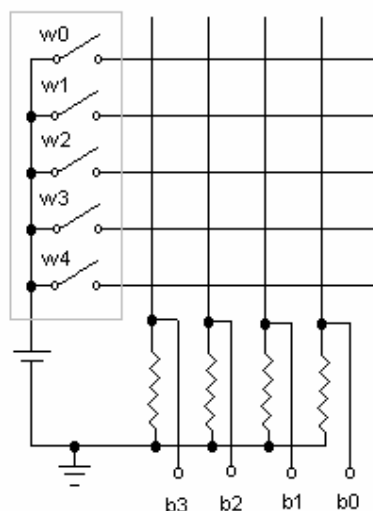
Nous allons nous baser sur le schéma précédent pour décrire les différentes phases caractérisant un transfert DMA.

- Chaque fois qu'un périphérique doit effectuer un transfert, il va activer la borne DREQ_n (n représentant le numéro du canal utilisé) du contrôleur DMA.
- Dès réception de cette demande, le contrôleur va activer la borne HOLD du microprocesseur pour lui indiquer qu'il désire utiliser ses bus.
- Lorsque le microprocesseur reçoit un tel signal, il termine son instruction en cours et active sa sortie HOLDA pour indiquer au contrôleur de DMA qu'il va placer ses bus dans un état haute impédance.
- Le contrôleur va alors déposer l'adresse mémoire sur le bus d'adresse, activer MEMR (Memory Read) et IOW (Input/Output Write) ou MEMW (Memory Write) et IOR (Input/Output Read) et activer le signal d'acquiescement de DMA vers le périphérique.
- Le périphérique répond alors au signal DACK_n en lisant ou en écrivant la donnée sur le bus de données.
- La borne HOLD est maintenue active vers le microprocesseur tant que le transfert n'est pas terminé. Dès le relâchement du signal, le microprocesseur reprend son travail jusqu'à la requête suivante.

7. Expliquer ce que sont les mémoires mortes (ROM et PROM) schéma de base et ses problèmes, rôle des diodes.

Les ROM sont des mémoires dites mémoires morte et non volatiles (read-only memory) cela signifie donc que se sont des mémoires auxquelles on ne peut accéder qu'en lecture et que l'absence de tension d'alimentation les données comprise dans la mémoire ne sont pas perdu.

Schéma de base :



La partie encadrée du schéma représente le décodeur d'adresse qui activera une des 5 ligne de la mémoire en fonction de l'adresse qu'il recevra.

Si Comme nous l'avons expliquer au début du cour de micro-ordinateur nous souhaitons associer un état logique 1 à la présence d'une tension (par exemple 5V) et un état logique 0 à une absence de tension, la cellule représentée ci-dessus serait composé de 4 bits a zéro.

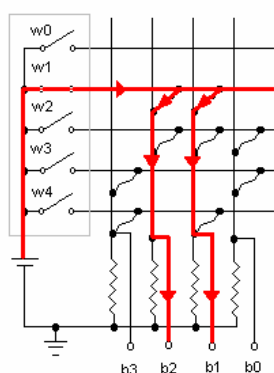
Pour faire apparaître un état logique 1 sur une sortie données, il faudra donc amener le niveau de tension qui se situe sur les lignes (horizontalement) sur les colonnes désirée à l'aide de lien électrique :

Imaginons que l'on veuille obtenir la table de vérité suivante:

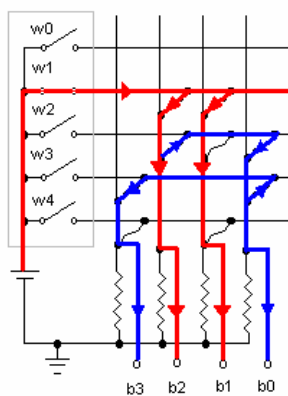
w4	w3	w2	w1	w0	b3	b2	b1	b0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	1	1	0
0	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	0	1
1	0	0	0	0	1	0	1	0

Notre schéma de base deviendrait donc :

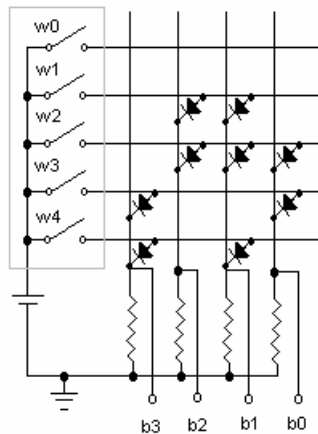
Prenons le cas où l'on voudrait lire la ligne w1.



On se rend compte que les liens que l'on a placés sur les autres ligne fausse notre lecture. C'est pourquoi les liens sont effectués à l'aide de semi conducteur qui consistent dans un sens mais pas dans l'autre.



Le schéma devient donc :



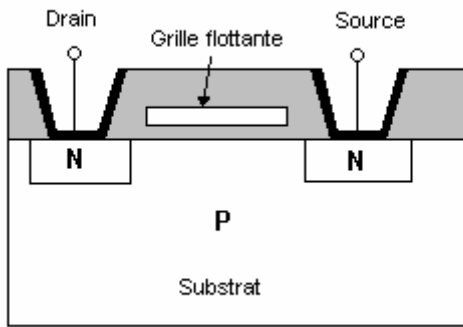
Remarque on utilisera plutôt des transistors au lieu de diode mais le principe reste le même. Les semi conducteurs sont placés à la fabrication, le contenu de la mémoire ne peut pas être changé ce qui explique pourquoi se sont des mémoires mortes.

Les mémoires PROM.

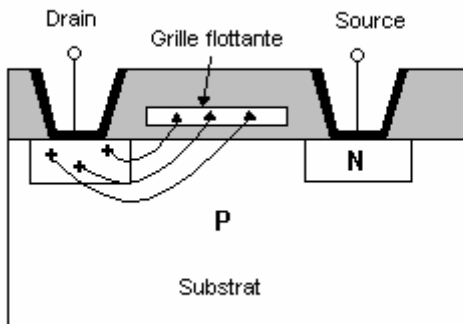
Ces mémoires sont également des mémoires mortes (non volatiles) dont le contenu ne s'efface pas si la tension d'alimentation disparaît. Ces mémoires peuvent être lues mais ne sont programmables qu'une seule fois par l'utilisateur. Le principe est simple: lors de la construction de ces mémoires, on va utiliser le principe des mémoires ROM avec diodes excepté que ces diodes sont placées à toutes les intersections. La lecture d'une telle mémoire non programmée nous fera apparaître tous les bits à l'état logique 1. La programmation consiste donc à détruire les diodes là où l'on désire qu'un 0 logique soit placé dans la cellule mémoire correspondante. Ces mémoires portent également le nom de OTP pour l'abréviation de On Time Programmable.

8. Sur base du schéma du transistor FAMOS, veuillez décrire la façon dont une cellule mémoire élémentaire dans les mémoires EPROM peut se programmer ou s'effacer. Expliquer l'évolution technologique dans le cadre des mémoires EEPROM et l'impact sur les possibilités d'enregistrement et d'effacement.

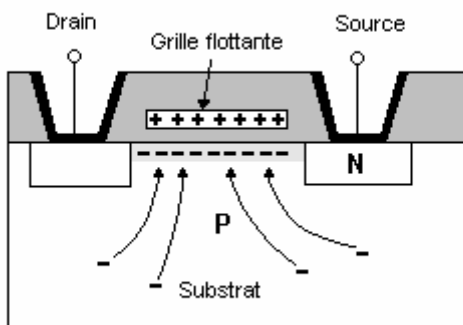
Ces mémoires sont des mémoires dites mortes mais elles sont effaçables et reprogrammables. Pour en comprendre la technique, nous allons analyser l'élément constitutif principal d'une cellule mémoire élémentaire qui est le transistor FAMOS (Floating Gate Avalanche Metal Oxide Silicium). Le schéma de principe d'un transistor FAMOS est le suivant:



Dans une zone P, on retrouve des donneurs d'ion (charges fixes) et des électrons (porteurs majoritaires mobiles) tandis que dans une zone N, on retrouve des accepteurs d'ion et des trous (porteurs majoritaires mobiles). Pour qu'un courant puisse circuler entre le drain et la source, il faut qu'il n'y ait entre ces deux zones que des porteurs majoritaires mobiles de même type. Nous allons décrire le procédé par lequel nous pouvons faire en sorte qu'un courant puisse circuler entre le drain et la source.



Les charges positives prisonnières dans la grille flottante vont pouvoir attirer les porteurs minoritaires présents dans le substrat et l'on va retrouver alors dans la zone P un endroit où la concentration en porteurs positifs sera plus grande que la concentration en porteurs négatifs. On parlera alors de la création d'un canal d'inversion.



Comme on retrouve alors le même type de porteurs mobiles entre le drain et la source, on pourra alors avoir un courant et le transistor est alors considéré comme passant.

Pour rendre le transistor à nouveau bloqué, il suffit de donner aux charges présentes dans la grille flottante suffisamment d'énergie que pour retourner dans le substrat. Cette énergie sera donnée en exposant le transistor à une lumière dans la gamme de l'ultra violet. C'est la raison pour laquelle on retrouve dans les mémoires de type EPROM une fenêtre en quartz présente sur le dessus du boîtier. Cette fenêtre doit être occultée lorsque la mémoire a été programmée pour ainsi éviter tout effacement accidentel.

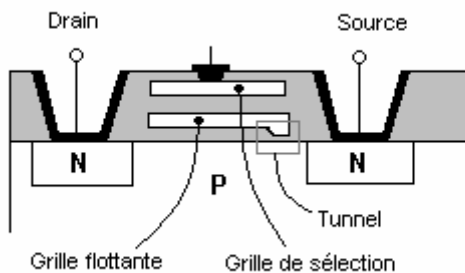
Il faut compter plusieurs minutes pour effacer une EPROM tandis que sa programmation nécessite une tension de l'ordre de 15 à 20 volts pour l'effet d'avalanche et il faut compter quelques μsec par mots pour la programmation. Il est important de signaler que le nombre de cycles (effacements / reprogrammations) est limité et avoisine les 1000 unités.

Expliquer l'évolution technologique dans le cadre des mémoires EEPROM et l'impact sur les possibilités d'enregistrement et d'effacement.

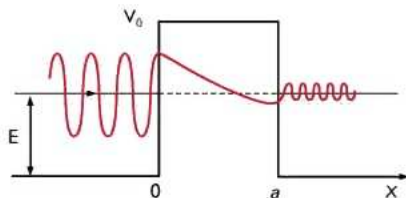
Un des aspects négatifs des mémoires EPROM est le fait que pour en effacer leur contenu, il faut les extraire de leur support en vue de les placer dans un effaceur d'EPROM constitué d'une lampe à ultra violet. La programmation nécessite également l'utilisation d'un niveau de tension que l'on ne retrouve pas sur les cartes mères d'ordinateur, et de ce fait, il faut donc disposer d'un programmeur d'EPROM externe pour cette opération.

L'idée est donc de pouvoir diminuer le niveau de tension nécessaire à la programmation et permettre une autre technique d'effacement.

EEPROM signifie "*Electrically Erasable Programmable ROM*". L'effacement d'une telle mémoire est électrique. Ces mémoires reposent sur l'utilisation d'un transistor FLOTOX. On y retrouve une grille flottante, une grille de sélection et le principe de l'effacement repose sur un effet tunnel que l'on appelle (Fowler-Nordheim tunneling).



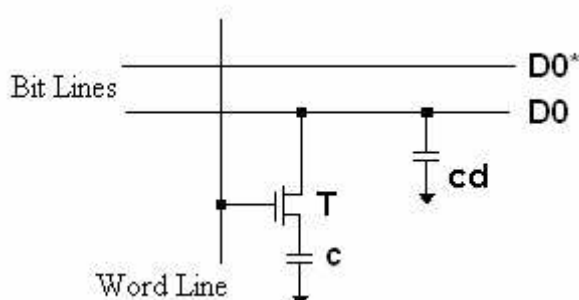
Si l'on considère un électron non plus en tant que particule mais en tant qu'onde, on peut expliquer que même si cet électron n'a pas l'énergie suffisante que pour franchir une barrière de potentiel, il pourra la traverser si cette barrière est mince.



Ce schéma illustre le fait que si la barrière de potentiel est très mince (quelques angströms), l'onde, quoique d'amplitude très faible pourra passer à travers la barrière de potentiel et se propager à nouveau dans la direction OX.

9. Dans le cadre du rafraîchissement des mémoires dynamiques, expliquer le fonctionnement détaillé de l'amplificateur différentiel sensitif (schéma, valeurs, calculs à l'appui).

Une mémoire dynamique est composée d'une capacité comme cellule mémoire élémentaire. Cette capacité va se décharger naturellement ainsi que lors du cycle de lecture.



Soient : V_0 la tension initiale aux bornes de C .
 V_{d0} la tension initiale aux bornes de cd .

Lorsque le transistor de selection est active :

$$V_{\text{final}}(c+cd) = V_0 * C + V_{d0} * cd$$

$$V_{d0} = \frac{1}{2} V_{cc} \text{ avec } V_{cc} = 3,3 \text{ volts}$$

Soit un zero logique alors $V_0 = \text{GND}$

Dans les calculs : $cd = 10 * C$

$$V_{\text{final}}(C+10C) = \frac{1}{2} * V_{cc} * 10C$$

$$V_{\text{final}}(1,65 \text{ volts} * 10/11) = 1,5V$$

Soit un 1 logique alors $V_0 = V_{cc} = 3,3 \text{ volts}$

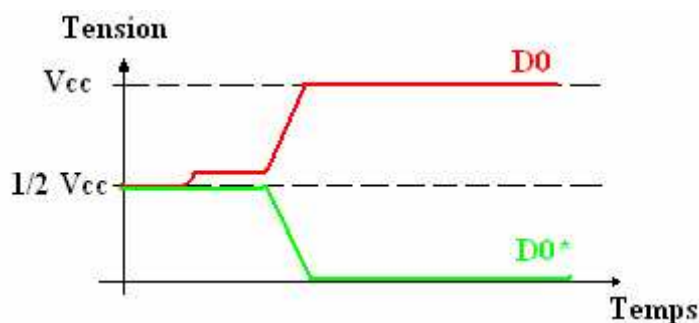
$$V_{\text{final}}(C+10C) = 3,3 * C + 1,65 * 10C$$

$$V_{\text{final}} = 3,3 + 16,5/11 = 1,8V$$

Si nous considérons la tension sur la ligne D_0 comme étant de 1.65, nous pouvons établir le tableau suivant :

Etat logique	Tension V_{d0}	Tension V_{d0}^*	Delta $V_{d0} - V_{d0}^*$
1	1.8V	1.65V	0.15V
0	1.5V	1.65V	-0.15V

La lecture du contenu de la cellule mémoire modifie la charge de la capacité C . Il sera donc important, à la suite de la phase de lecture, de restaurer les charges dans la capacité de stockage en utilisant un amplificateur différentiel sensitif. Cet amplificateur aura la structure simplifiée suivante :

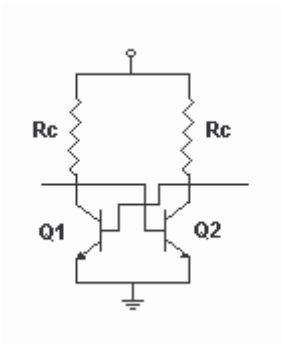


Le chronogramme ci-dessus représente l'évolution des tensions sur les bornes D_0 et D_0^* de l'amplificateur sensitif lorsque une cellule mémoire correspondant à un état logique '1' est lue. La charge de la capacité est donc restaurée à son état d'origine. Lors d'un accès en lecture, on suppose que la tension initiale sur les bornes D_0 et D_0^* est de $\frac{1}{2} V_{cc}$. Il est important que ces lignes de la DRAM soient maintenues à un potentiel de $\frac{1}{2} V_{cc}$ avant toute opération de lecture. C'est ce que l'on appelle le pré chargement de la DRAM. Ce temps de pré chargement apparaît dans le chronogramme d'un cycle de lecture d'une mémoire dynamique.

10. Sur base de son schéma électronique de principe, veuillez décrire le fonctionnement d'une cellule mémoire élémentaire de type SRAM.

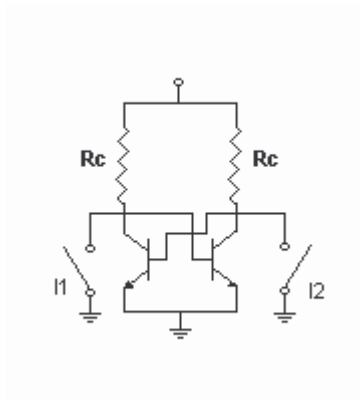
SRAM signifie Static Random access memory. Ces mémoires sont dites volatiles c-à-d que l'absence de tension d'alimentation provoque la perte des données comprises dans ces mémoires. Ces mémoires sont accessibles en lecture/écriture.

Le schéma de base d'une cellule mémoire SRAM est le suivant :



Le principe de ce montage est d'obtenir deux états stables que l'on pourrait assigner à un 1 ou un 0 logique. Analysons le schéma, prenons au départ le transistor Q1 bloqué, puisqu'il est bloqué aucun courant ne le traverse, en revanche un courant de base apparaît et polarise le transistor Q2 qui devient passant. Comme le schéma est parfaitement symétrique on peut affirmer que l'on aurait eu le résultat inverse si nous avions pris Q2 bloqué pour point de départ. Nous avons donc obtenu 2 états stables, mais il faut ajouter des éléments pour nous permettre de passer d'un état à l'autre

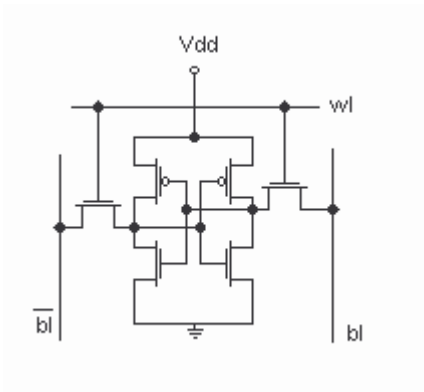
Le schéma devient donc :



Prenons la situation de départ suivante: Q1 est passant, Q2 est bloqué et les deux interrupteurs I1 et I2 sont ouverts.

Fermons alors l'interrupteurs placés en parallèle sur le transistor bloqué. Le potentiel du collecteur de Q2 est tiré à la masse par cet interrupteur et aucun courant ne peut plus circuler dans la base de Q1 qui se bloque. Si Q1 se bloque, un courant peut alors circuler dans la base de Q2 qui devient alors passant ce qui nous autorise alors à rouvrir l'interrupteur I2 et le montage se retrouve alors dans le second état stable.

Les interrupteurs I1 et I2 sont eux mêmes des transistors, mais pour des question d'intégration Pour une question d'intégration, les transistors bipolaires et les résistances sont remplacés par des transistors à effet de champs de type MOS. Nous obtenons finalement le schéma suivant:

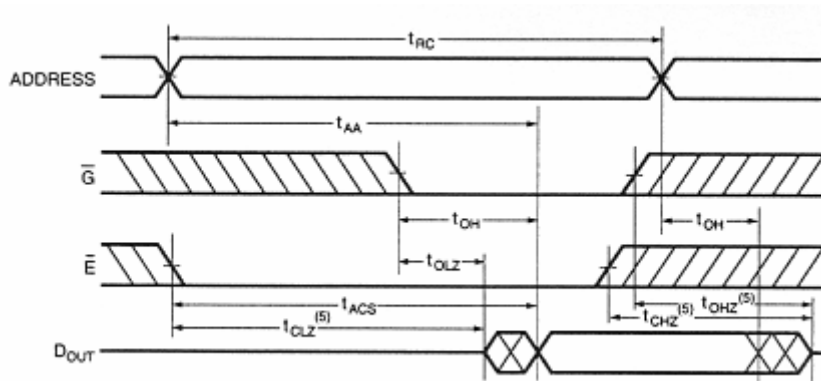


11. Dans le cas d'une mémoire statique, en vous basant sur un chronogramme qui sera fourni, pouvoir définir et calculer ce que sont les temps de cycle, d'accès et de décodage.

Le temps d'accès d'une mémoire est toujours donné en rapport à un cycle de lecture et correspond au temps séparant la réception de l'adresse par la mémoire et la fourniture de la donnée par cette dernière. Dans le chronogramme, nous retrouvons le temps t_{AA} ou $t_{a(A)}$ qui dans le cas de notre mémoire, correspond à 70 nsec et qui va figurer sur le boîtier sous la forme MS62256L-70.

Le temps de cycle d'une mémoire correspond au temps que prend dans le cas de notre chronogramme un cycle complet de lecture et que l'on puisse alors démarrer le cycle suivant. Dans le cas de la mémoire choisie, le temps de cycle est de 70 nsec.

Nous pouvons en déduire que dans le cas des mémoires statiques, le temps de cycle est identique au temps d'accès.



12. Veuillez expliquer par un exemple commenté, l'utilité de la borne Chip Select dans les mémoires statiques.

Pour comprendre l'utilité de la borne 'Chip Select', on considère un microprocesseur ayant un bus d'adresse de 17 bits pour un espace adressable de 128Ko (2^{17}). Si on se limite à l'utilisation des mémoires de 32K x 8, il faudra donc 4 boîtiers pour couvrir les 256Ko.

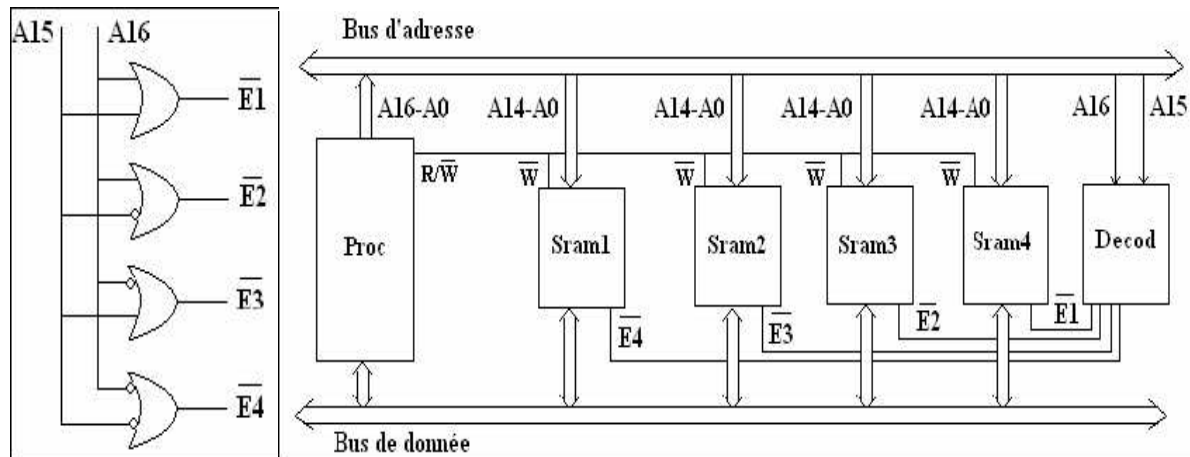
Adresse du microprocesseur (17 bits) $A_{16}-A_0$	Adresse de la mémoire (15 bits)	N° boîtier
00000000000000000 jusque 00111111111111111	000000000000000 jusque 111111111111111	1
01000000000000000 jusque 01111111111111111	000000000000000 jusque 111111111111111	2
10000000000000000 jusque 10111111111111111	000000000000000 jusque 111111111111111	3
11000000000000000 jusque 11111111111111111	000000000000000 jusque 111111111111111	4

Soit l'adresse suivante déposée sur le bus d'adresse par le microprocesseur : **10100111100100101**. L'ensemble des mémoires connectées sur le bus d'adresse ne voient que les 15 bits de poids faible de cette adresse, à savoir **100111100100101** bien que cette demande soit adressée au boîtier numéro 3 et pas aux autres. Il faudra donc sélectionner le boîtier 3 et pas les autres grâce à la borne E de chacune des mémoires.

Adresse du microprocesseur (17 bits)	E boîtier 1	E boîtier 2	E boîtier 3	E boîtier 4
00xxxxxxxxxxxxxxx	0	1	1	1

01xxxxxxxxxxxxxxxx	1	0	1	1
10xxxxxxxxxxxxxxxx	1	1	0	1
11xxxxxxxxxxxxxxxx	1	1	1	0

Nous pouvons donc envisager le schéma logique suivant :



Si l'on tient compte du temps de la propagation des signaux dans le décodeur d'adresse, le signal d'activation du boîtier arrivera après l'adresse. Nous allons analyser le chronogramme pour déterminer s'il y a des limitations dans le temps que le décodeur peut mettre pour traiter l'adresse. Dans le cas la mémoire choisie, le temps t_{ACS} est également de 70 nsec, ce qui nous laisse penser que si le décodage d'adresse prend x nsec pour traiter l'adresse, la donnée ne sera disponible sur le bus de données que $(70+x)$ nsec. Si nous avons choisi de sélectionner les boîtiers en utilisant la borne G au lieu de la borne E, nous aurions alors pu avoir un temps de décodage possible de $t_{AA} - t_{OHc}$ -à-d $70 - 10 = 60$ nsec. Si nous choisissons des portes logiques ayant un temps de propagation de 10 nsec, nous pourrions nous permettre d'avoir 6 portes en cascade entre le signal de sortie et les signaux d'entrée.

13) Veuillez expliquer en quoi consiste l'utilisation des techniques de détection et/ou de correction d'erreur dans les mémoires. (Technique de la parité, technique ECC)

14. Question pouvant comprendre plusieurs sous questions relatives à des définitions dont voici une liste non exhaustive :

- [Expliquer ce qu'est le NetBurst](#)

Netburst est le nom d'une architecture Intel implémentée dans certains Pentium 4. Cette architecture contient une profondeur de pipeline record de 20 niveaux.

- [Expliquer ce que sont les processeurs CISC et RISC](#)

L'architecture CISC (*Complex Instruction Set Computer*, ce qui signifie "ordinateur avec jeu d'instructions complexes") est utilisée par tous les processeurs de type x86, c'est-à-dire les processeurs fabriqués par Intel, AMD, Cyrix, ...

Les processeurs basés sur l'architecture CISC peuvent traiter des instructions complexes, qui sont directement câblées sur leurs circuits électroniques, c'est-à-dire que certaines instructions difficiles à créer à partir des instructions de base sont directement imprimées sur le silicium de la puce afin de gagner en rapidité d'exécution sur ces commandes.

L'inconvénient de ce type d'architecture provient justement du fait que des fonctions supplémentaires sont imprimées sur le silicium, d'où un coût élevé.

D'autre part, les instructions sont de longueurs variables et peuvent parfois prendre plus d'un cycle d'horloge ce qui les rend lentes à l'exécution étant donné qu'un processeur basé sur l'architecture CISC ne peut traiter qu'une instruction à la fois!

Contrairement à l'architecture CISC, un processeur utilisant la technologie RISC (*Reduced Instruction Set Computer*, dont la traduction est "ordinateur à jeu d'instructions réduit") n'a pas de fonctions supplémentaires câblées. Cela impose donc des programmes ayant des instructions simples interprétables par le processeur. Cela se traduit par une programmation plus difficile et un compilateur plus puissant. Cependant vous vous dites qu'il peut exister des instructions qui ne peuvent pas être décrites à partir des instructions simples...

En fait ces instructions sont tellement peu nombreuses qu'il est possible de les câbler directement sur le circuit imprimer sans alourdir de manière dramatique leur fabrication.

L'avantage d'une telle architecture est bien évidemment le coût réduit au niveau de la fabrication des processeurs l'utilisant. De plus, les instructions, étant simples, sont exécutées en un cycle d'horloge, ce qui rend l'exécution des programmes plus rapides qu'avec des processeurs basés sur une architecture CISC.

De plus, de tels processeurs sont capables de traiter plusieurs instructions simultanément en les traitant en parallèle.

Bien sûr, les implications sont nombreuses, mais s'il ne faut en retenir qu'une ce doit être que, sur une machine RISC, la diminution de la complexité matérielle est compensée par un compilateur très évolué.

- [Expliquer ce que sont les mémoires BUFFERED et UNBUFFERED](#)

mémoires BUFFERED et UNBUFFERED :

Si l'on veut comprendre la différence entre ces deux modèles, nous devons parler de la sortance des composants électroniques c à d du nombre d'entrées de composants d'une même technologie que l'on peut mettre sur une même sortie. Dans le cas des modules UNBUFFERED, leur nombre sur une carte mère est limité à quelques uns tandis que dans le cas des modèles REGISTERED/BUFFERED, nous pourrions atteindre une vingtaine de modules. On retrouvera donc ces derniers principalement utilisés sur les cartes mères de serveur où la capacité mémoire offerte est plus importante.

Pour ne pas faire le mauvais choix d'un modèle de mémoire pour une carte mère donnée, ces mémoires sont pourvues de deux encoches dont la position diffère en fonction de la tension d'alimentation et en fonction qu'elle soit ou pas REGISTERED/BUFFERED.

Les Mémoires bufferisées sont généralement plus haute car elle intègre quelques composant en plus permettant d'améliorer la sortance.

- [Expliquer ce qu'est le prefetch](#)

La dernière technologie en matière de mémoire est la DDRII. Pour bien comprendre l'évolution, nous devons intégrer une nouvelle caractéristique dans les mémoires qui est le prefetch. Pour bien comprendre, nous allons reprendre l'architecture d'une mémoire comme composée de trois éléments fondamentaux : la cellule mémoire, le buffer d'entrée/sortie et le bus de données. Dans le cas d'une mémoire de type PC100, l'ensemble de ces trois groupes fonctionne à une cadence de 100 MHz et la cellule mémoire fournit un seul bit.

Pour la mémoire de type DDR I, le fait de pouvoir travailler pour la DDR PC 1600 à une fréquence de 200 MHz en effectuant des transferts sur le flanc montant et sur le flanc descendant (2x100MHz), oblige les constructeurs, du fait des limitations technologiques propres aux mémoires, de travailler avec une double cellule mémoire capable de fournir 2 bits en conservant une cadence de fonctionnement de 100 MHz. Ce nombre de bits transférés par la cellule mémoire élémentaire s'appelle le prefetch.

Comme nous l'avons vu précédemment, plus la fréquence de fonctionnement augmente, plus la dissipation thermique du composant est importante. C'est pour cela que pour la technologie DDR II, on passe à un prefetch de 4, ce qui nous permet de pouvoir travailler avec une fréquence de fonctionnement moindre et de ce fait pouvoir diminuer la tension d'alimentation qui passe de 2,5V à 1,8V

Type	Fréquence interne	Prefetch	Vitesse	Débit
PC-3200	200MHz	2	400MHz	400MHz*8
PC2-3200	100MHz	4	400MHz	400MHz*8
PC2-5300	133MHz	4	533MHz	533MHz*8

- [Expliquer ce qu'est le CAS Latency Time \(exemple chiffré\)](#)
- [Expliquer ce que sont les différents types de rafraîchissement](#)

Le rafraîchissement consiste simplement à accéder à une ligne et automatiquement, toutes les colonnes de cette ligne sont rafraîchies.

Prenons par exemple une mémoire FPM de chez Micron, la MT4LC4M4B1. Cette mémoire de 4Meg x 4, présente une organisation matricielle de 2048 lignes x 2048 colonnes, les 2048 lignes devant être rafraîchies toutes les 32 msec. Pour y arriver, plusieurs solutions sont envisageables :

1) Rafraîchir toutes les lignes en une seule fois et attendre ensuite 32 msec avant de recommencer. C'est ce que l'on appelle le rafraîchissement en rafale (Burst Refresh).

2) Rafraîchir une ligne toutes les 32msec/2048lignes c à d toutes les 15.63µsec. C'est ce que l'on appelle le rafraîchissement en coup par coup (One Shot Refresh).

Du côté du microprocesseur, le rafraîchissement nécessite que ses bus soient disponibles. Cette disponibilité est possible pendant les cycles de décodage des instructions (cycles fetch) mais également en demandant au microprocesseur l'utilisation de ses bus en utilisant les bornes Hold et Hold Acknowledge. C'est que le l'on appellera le rafraîchissement caché (Hidden Refresh) et le rafraîchissement forcé (Forced refresh).

Le rafraîchissement en rafale sera généralement lié au rafraîchissement en rafale tandis que le rafraîchissement en coup par coup sera lié au rafraîchissement caché.

- [Expliquer ce qu'est le temps de pré chargement dans les mémoires dynamiques](#)
- [Expliquer l'évolution technologique, chiffres à l'appui, entre les mémoires dynamiques de type FPM, EDO, SDRAM, DDR, notamment pour les transferts en rafales et les différentes fréquences rencontrées. Pouvoir associer les différents types de boîtiers \(DIP, SIMM, DIMM\) utilisés \(ne pas savoir les dessiner dans le détail\) ainsi que la taille des bus de données et les vitesses de transferts, le fait que ces mémoires soient synchrones ou asynchrones.](#)

Une façon particulière de noter les mémoires FPM était 6-3-3-3 pour un bus cadencé à 66MHz c-à-d que pour un accès à la première donnée, il fallait compter 6 cycles d'horloge tandis que pour les suivantes, uniquement 3 cycles d'horloge processeur.

Ces mémoires disparaissent avec l'arrivée sur le marché des processeurs Pentium 1 au profit des mémoires EDO (Enhanced Data Output). Dans ces mémoires, on peut fournir l'adresse de la colonne suivante à laquelle on désire accéder tandis que la donnée correspondant à la cellule précédente se trouve toujours sur le buffer de sortie de la mémoire. Cela crée donc un chevauchement des accès permettant de gagner du temps sur chaque cycle.

Pour les mémoires FPM, pour une cadence d'horloge de 66Mhz, nous retrouvons une caractéristique de temps de cycle de 6-3-3-3 ou 5-3-3-3. Pour les mémoires EDO, du fait que l'on puisse déposer l'adresse de la cellule suivante alors que la donnée de la cellule précédente est toujours dans le buffer de sortie, nous retrouvons un temps de cycle de 5-2-2-2, soit un gain de quatre cycles sur l'accès des quatre données.

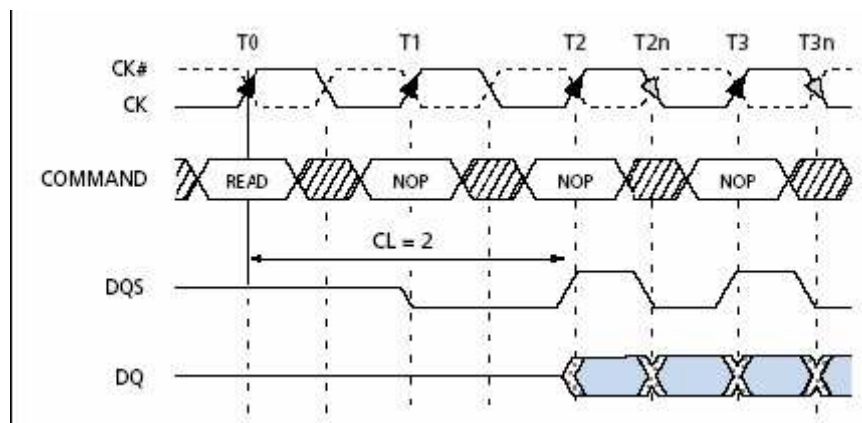
Pour les mémoires synchrones, nous retrouvons comme première mémoire la mémoire SDRAM (RAM dynamique synchrone) c à d que parmi les bornes de la mémoire, nous retrouvons un signal d'horloge sur lequel les différents accès à la mémoire seront synchronisés.

Lorsque l'on caractérise la mémoire, on n'évoque plus le temps d'accès mais la fréquence du bus avec laquelle nous pouvons travailler sur ces mémoires. On parle alors de mémoire SDRAM de type PC100 ou PC133. On évoque également le cycle correspondant à cette horloge en pensant naïvement qu'il

s'agit du temps d'accès. On parle de mémoire à 10nsec, 8nsec...

Reprenons le chronogramme d'une telle mémoire afin d'en calculer le temps d'accès: nous retrouvons le temps tRCD (RAS to CAS Delay time), le temps tRP (RAS Precharge time) et le temps CAS Latency qui joueront un rôle important pour déterminer la vitesse maximale du bus à laquelle le module mémoire peut travailler. On atteint généralement de timings du type 4-1-1-1. Il faut quatre cycles pour accéder à la première donnée, ce qui correspond pour une horloge cadencée à 100Mhz à un temps d'accès de 40nsec. Les données suivantes sont accédées avec une cadence de l'ordre de 1 donnée toutes les 10nsec.

Les mémoires DDR se basent sur le même concept que les mémoires SDRAM excepté que nous envisageons lors des rafales un transfert sur le flanc montant et sur le flanc descendant ce qui permet de doubler la vitesse de transfert, d'où la dénomination donnée à cette technologie: DDR SDRAM (Double Data Rate). Nous parlons alors de DDR 200, DDR266, DDR333 et DDR400.



Nous retrouverons le même paramétrage pour le temps CAS Latency Time suivant le tableau suivant:

ALLOWABLE OPERATING FREQUENCY (MHz)		
SPEED	CL = 2	CL = 3
-5	$83 \leq f \leq 125$	$83 \leq f \leq 200$
-55	$83 \leq f \leq 100$	$83 \leq f \leq 183$
-6	$83 \leq f \leq 100$	$83 \leq f \leq 166$
-65	$83 \leq f \leq 100$	$83 \leq f \leq 150$

On retrouve une autre façon de pouvoir identifier un module mémoire, non pas sur la fréquence d'horloge mais sur le débit de données que cette mémoire peut avoir. Si nous envisageons une mémoire dont la fréquence de fonctionnement serait de 100MHz, nous pouvons, lors des rafales, espérer le débit suivant: $200 * 8 = 1600$ 100MHz * 2 (2 flancs) * 8 (nombre d'octets transférés sur le bus de données). On parle alors de mémoire PC1600

Pour la DDR266, nous obtenons le calcul suivant: $133 * 2 * 8 = 2100$. On parle de PC2100.

Pour la DDR333, nous obtenons le calcul suivant: $166 * 2 * 8 = 2700$. On parle de PC2700.

Pour la DDR400, nous obtenons le calcul suivant: $200 * 2 * 8 = 3200$. On parle de PC3200.

La dernière technologie en matière de mémoire est la DDRII. Pour bien comprendre l'évolution, nous devons intégrer une nouvelle caractéristique dans les mémoires qui est le prefetch. Pour bien comprendre, nous allons reprendre l'architecture d'une mémoire comme composée de trois éléments fondamentaux : la cellule mémoire, le buffer d'entrée/sortie et le bus de données. Dans le cas d'une mémoire de type PC100, l'ensemble de ces trois groupes fonctionne à une cadence de 100 MHz et la cellule mémoire fournit un seul bit.

Pour la mémoire de type DDR I, le fait de pouvoir travailler pour la DDR PC 1600 à une fréquence de 200 MHz en effectuant des transferts sur le flanc montant et sur le flanc descendant (2x100MHz), oblige les constructeurs, du fait des limitations technologiques propres aux mémoires, de travailler avec une double cellule mémoire capable de fournir 2 bits en conservant une cadence de

fonctionnement de 100 MHz. Ce nombre de bits transférés par la cellule mémoire élémentaire s'appelle le prefetch.

Comme nous l'avons vu précédemment, plus la fréquence de fonctionnement augmente, plus la dissipation thermique du composant est importante. C'est pour cela que pour la technologie DDR II, l'on passe à un prefetch de 4, ce qui nous permet de pouvoir travailler avec une fréquence de fonctionnement moindre et de ce fait pouvoir diminuer la tension d'alimentation qui passe de 2,5V à 1,8V

Début février 2005, Samsung annonce qu'il est le premier à avoir produit un prototype de mémoire DDR3. Fonctionnant en 1.5V, cette puce de 512Mb gravée en 80nm est de type DDR3-1066. A titre de comparaison, la DDR fonctionne en 2.5V et la DDR2 fonctionne en 1.8V. Pour que cela soit possible, le prefetch, qui était passé de $2n$ à $4n$ bits lors du passage de la DDR à la DDR-2, passe désormais à $8n$ bits. Nous conservons donc pour cette nouvelle génération de mémoire une fréquence interne de 133 Mhz, tandis qu'avec un prefetch de 8, nous pouvons atteindre une vitesse de $133*8=1064$ Mhz ce qui nous permettra donc d'atteindre un débit de $1064\text{Mhz}*8=8\text{Go/sec}$. Samsung annonce la commercialisation de cette mémoire pour début 2006.

Selon un autre constructeur de mémoire Micron, les prévisions sont les suivantes: pour début 2007, nous pourrions retrouver sur le marché la DDR3-1333 et pour 2008, la DDR3-1600.

a) Les boîtiers DIP (Dual In Line Package).

A l'origine, les boîtiers mémoires sont semblables aux composants électroniques classiques sous forme de boîtiers de type DIP. Tout boîtier de ce type est identifié par le nombre de bornes ainsi que la largeur du boîtier : on parlera par exemple de boîtiers 14 DIP 300 ou 14 DIP 600.

14 DIP 300 signifie que l'on retrouve 14 bornes en deux rangées parallèles de 7 bornes espacées chacune de 1/10 de pouce. Pour la valeur 300, nous aurons une largeur de 3/10 de pouce. Dans les premiers ordinateurs, ces boîtiers étaient directement intégrés sur la carte mère.



b) Les barrettes mémoire SIMM (Single In Line Memory Module) 30 contacts et 72 contacts

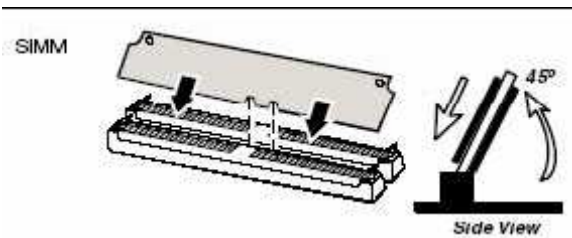


Les barrettes SIMM 30 contacts permettent des transferts sur 8 bits. Dans le cas par exemple des ordinateurs équipés d'un processeur de type 486, processeur 32 bits avec un bus de données de 32 bits, il faut prévoir d'installer les mémoires par groupe de quatre barrettes pour couvrir ainsi les 32 bits du bus de données.



Les barrettes SIMM 72 contacts apparaissent permettant ainsi de n'utiliser qu'une seule barrette avec ce type de processeur mais avec l'apparition des premiers PENTIUM, processeur 32 bits avec un bus de données de 64 bits il faut installer les barrettes par groupe de deux.

Celles-ci peuvent être ajoutées facilement sur la carte mère grâce à des connecteurs.



c) Les barrettes mémoire DIMM (Dual In Line Memory Module)

L'insertion du module est différente de l'insertion d'un module SIMM. Celui-ci doit simplement s'insérer verticalement

Bien que portant le même nom, les barrettes mémoire DIMM évoluent en même temps que les différentes technologies mémoires :

- DIMM 168 broches pour les mémoires de type SDRAM
- DIMM 184 broches pour les mémoires de type DDR I. Outre la différence résultant d'un nombre de bornes différent pour une dimension presque identique, les modules DIMM 184 ne disposent que d'une seule encoche dont la position dépend de la tension d'alimentation du module.
- DIMM 240 broches pour les mémoires de type DDR II

- [Donner le chronogramme de base d'un accès en lecture dans une mémoire dynamique \(non en rafale\) avec les différents signaux utilisés et leur définition.](#)
- [Expliquer ce qu'est l'exécution dynamique dans les microprocesseurs.](#)

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

1. La prédiction de branchement consiste à deviner l'emplacement de la prochaine instruction à exécuter et donc à la charger dans le pipeline. En effet, un saut oblige le processeur à recharger ses pipelines avec des données pertinentes. Pendant ce temps, le programme perd l'accélération apportée par le pipeline. Concernant le Pentium II, Intel annonce une capacité de prédiction record de 90%.

2. L'analyse de flux permet au processeur de modifier l'ordre d'entrée des instructions d'un programme dans le pipeline pour en optimiser le taux d'occupation. Par exemple, sur les Pentium, il s'agit de choisir entre les deux pipelines entiers en fonction d'un éventuel calcul flottant - obtenu par la dérivation du pipeline entier. Le tout, bien sûr, en tenant compte du temps nécessaire à l'exécution des instructions. Le nouvel ordre établi peut être complètement différent de celui imaginé par un piètre compilateur. L'exécution spéculative, enfin, fonctionne de concert avec la prédiction de branchement.

Ainsi, l'architecture scalaire du Pentium II exécute les instructions des différentes portions de code envisageable (après un saut, par exemple).

- [Expliquer de façon technique ce que l'on entend par le partage des interruptions et son évolution.](#)
- [Expliquer ce qu'est le busmastering ou DMA 'first-party'.](#)

Etant donné que ce mode de transfert est lié fortement aux caractéristiques des disques durs, nous allons reprendre dans un tableau l'ensemble des différentes normes avec la vitesse de transfert correspondante et un comparatif avec le mode PIO correspondant.

PIO mode 0	ATA (ATA-1 ou IDE)	3.33 MB/sec	1994
DMA Mode 0 ou DMA Multiword Mode 0	ATA	2.1-4.16 MB/sec	
PIO Mode 1	ATA	5.22 MB/sec	1996
PIO Mode 2	ATA	8.33 MB/sec	
PIO Mode 3	ATA-2 (EIDE ou FASTATA)	11.1 MB/sec	
DMA Mode 1 ou DMA multiword Mode 1	ATA-2	4.2-13.3 MB/sec	

PIO Mode 4	ATA-2	16.6 MB/sec	
DMA Mode 2 ou DMA multiword Mode 2	ATA-2	8.3-16.6 MB/sec	
Ultra DMA Mode 0	ATA/ATAPI 4	16.6 MB/sec	1998
Ultra DMA Mode 1	ATA/ATAPI 4	25.0 MB/sec	
Ultra DMA Mode 2, UDMA 33 ou ATA/33	ATA/ATAPI 4	33.3 MB/sec	1999
Ultra DMA Mode 3	ATA/ATAPI 5	44.4 MB/sec	2000
Ultra DMA Mode 4, UDMA 66 ou ATA/66	ATA/ATAPI 5	66.6 MB/sec	2000
Ultra DMA Mode 5, UDMA 100 ou ATA/100	ATA/ATAPI 6	100 MB/sec	2000
Ultra DMA Mode 6, UDMA 133 Ou ATA/133	ATA/ATAPI 7	133 MB/sec	2001

L'augmentation du débit des différentes normes s'explique par une diminution du temps de cycle nécessaire pour effectuer un transfert. L'utilisation des deux flancs d'horloge pour effectuer un transfert permet également d'augmenter le débit. C'est ce qui explique que l'on ait pu passer à la norme UDMA 33 présentant un débit de 33.3 MB/sec alors que l'horloge présentait un temps de cycle identique à celle utilisée pour la norme DMA Mode 0 (multiword) qui était limitée à un débit de 16.6 MB/sec. Le passage aux normes UDMA 66 et UDMA 100 s'est effectué en augmentant la cadence d'horloge ce qui a d'ailleurs justifié le passage d'un câble standard IDE à 40 conducteurs à un câble IDE à 80 conducteurs. L'UDMA 100 présente un temps de cycle de 40 ns tandis que l'UDMA 33 de 120 nsec. Attention: un cycle de transfert n'est pas nécessairement égal à un cycle d'horloge. Il faut en effet parfois plusieurs cycles d'horloge pour effectuer un transfert complet (le temps nécessaire pour ce transfert est également appelé cycle).