Technologie du matériel

Architecture des ordinateurs

Guillaume Blin

IGM-LabInfo UMR 8049,
Bureau 4B066
Université de Marne La Vallée
gblin@univ-mlv.fr
http://igm.univ-mlv.fr/~gblin

Références

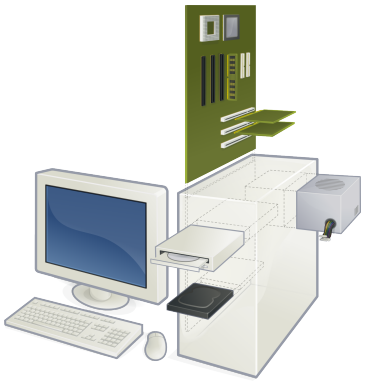
- Cours de Frédéric Goualard de l'Université de Nantes
  http ://supports.goualard.free.fr/

Plan

Plan

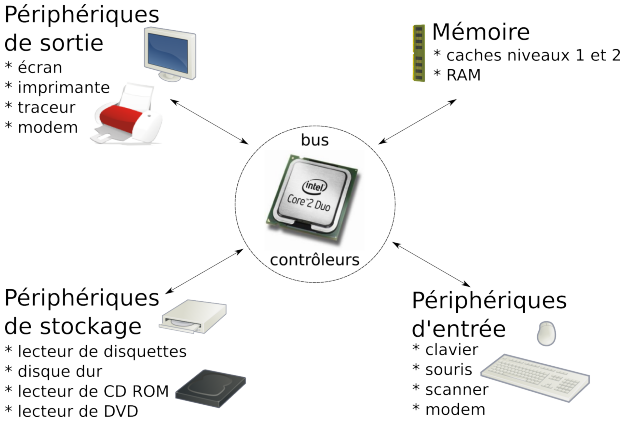
Le PC et ses périphériques

- Intéractions entre le CPU et les autres composants du PC ?
- Intéractions entre le CPU et les périphériques ?
- Technologies utilisées par les périphériques ?



Retour sur l'architecture de Von Neumann

- Vision schématique de l'ordinateur :

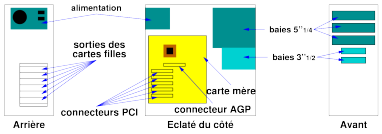


- Périphérique d'entrée : de l'extérieur vers le processeur
- Périphérique de sortie : du processeur vers l'extérieur

Architecture du PC

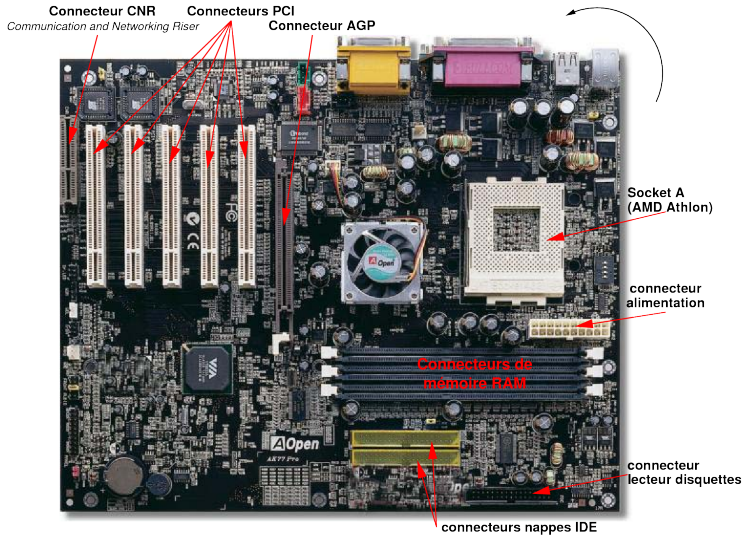
- PC = architecture modulaire « en Lego » :
  - Carte mère : CPU + mémoire + connecteurs
  - Cartes filles : cartes additionnelles spécialisées :
    - Carte modem
    - Carte son
    - Carte video
    - ...
- L'utilisateur peut facilement rajouter ou changer des fonctionnalités grâce aux cartes filles

Le boitier-tour ATX

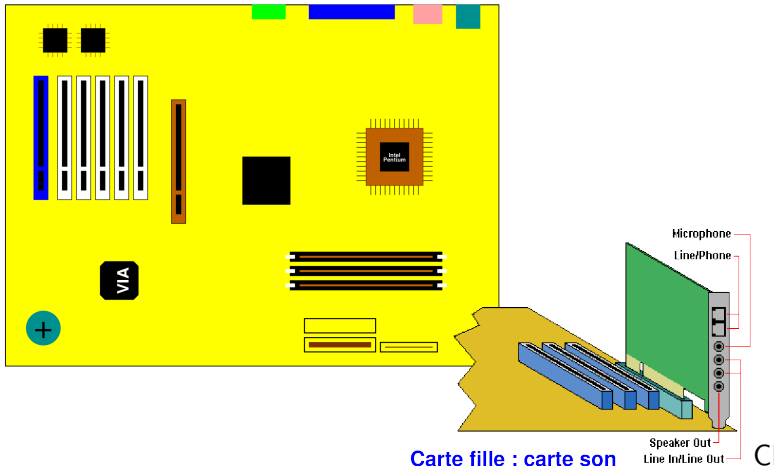


- Baies 5"1/4 : lecteur de CD-ROM, lecteur de disquette 5"1/4
- Baies 3"1/2 : lecteur de disquette 3"1/2, disque dur, lecteur zip

La carte mère ATX



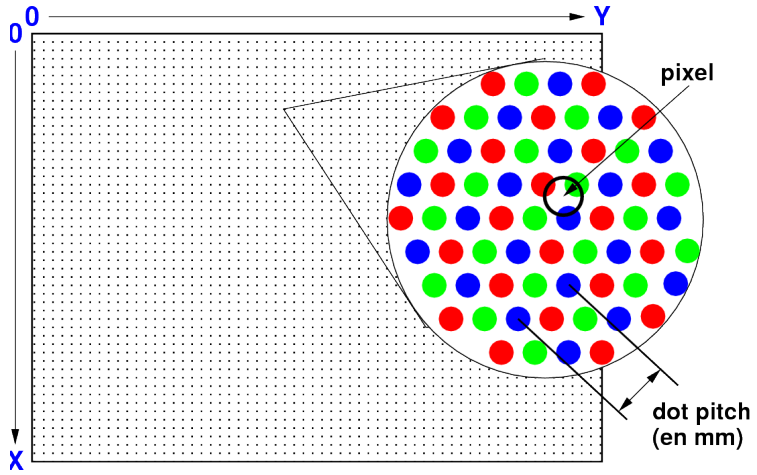
La carte mère ATX



Carte fille : carte son CNR supports audio, modem USB and Local Area Networking interfaces

Plan

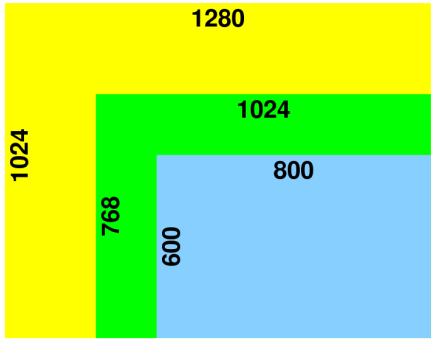
L'écran



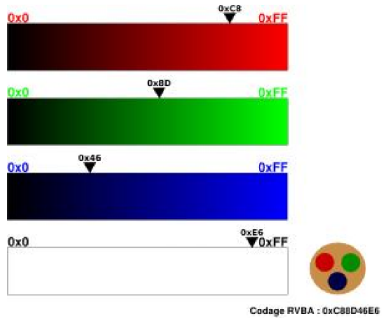
La résolution

- Résolution : nombre de pixels affichés
- Profondeur de couleur : nombre de bits par pixel (8 bits = 256 couleurs/pixel, ...)
- Codage des couleurs en RVB

Résolutions classiques :



Le codage RVBA



Le codage RVBA :

- n bits codant le niveau de rouge, vert, bleu
- m bits codant le niveau de transparence (*alpha channel*)
- Codage classique : $8 \times 8 \times 8 \times 8 = 32$ bits par pixel

Standards d'affichage

- Résolution : nombre de colonnes $\times$ nombre de lignes
- Profondeur : nombre de bits par pixel

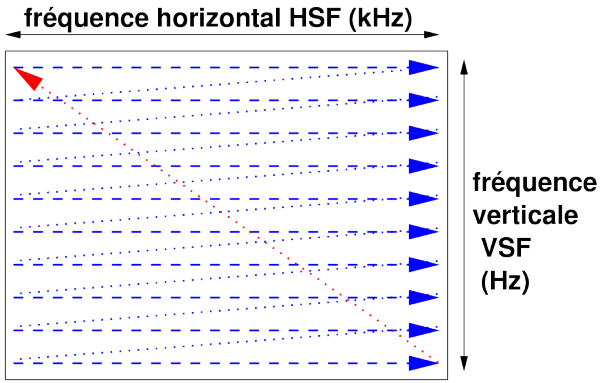
Création	Nom	Résolution & profondeur
1981	Hercules (texte seul)	$80 \times 25 \times 1$
1981	CGA (Color Graphics Array)	$320 \times 200 \times 2$
1987	VGA (Video Graphics Array)	$640 \times 480 \times 4$
1990	XGA (eXtended Graphics Array)	$1024 \times 768 \times 24$
	UXGA (Ultra XGA)	$1600 \times 1200 \times 24$

Types d'écrans

Deux technologies :

- Écrans CRT (*Cathod Ray Tube*)
- Écrans LCD (*Liquid Crystal Display*)

Fréquence de rafraîchissement (CRT)



$$\text{VSF} = \frac{\text{HSF}}{\text{nb. lignes}} \times 0.95$$

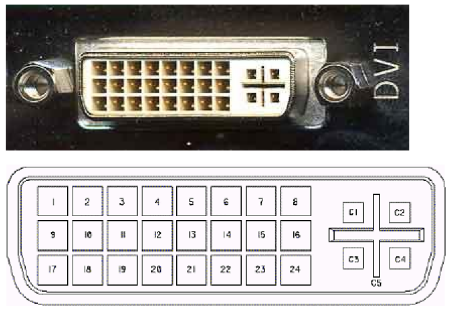
VSF minimale recommandée : 75 Hz

Le connecteur VGA 15 broches

1. Signal ROUGE
2. Signal VERT
3. Signal BLEU
6. Terre ROUGE
7. Terre VERT
8. Terre BLEU
10. Terre synchro
13. Synchro horizontale
14. Synchro verticale

- Signal analogique (voltage codant le niveau de R/V/B)
- la connectique est restée les normes ont évolués (SVGA, ...)
- supplenté par DVI

Le connecteur DVI (Digital Visual Interface)

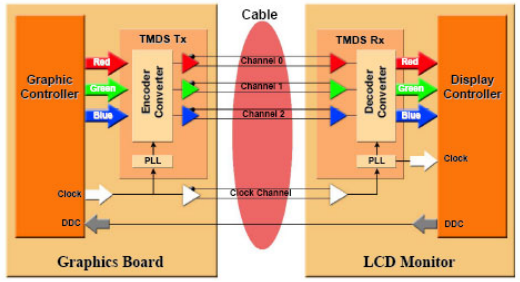


1. Data 2–
2. Data 2+
4. Data 4–
5. Data 4+
9. Data 1–
10. Data 1+
12. Data 3–
13. Data 3+
17. Data 0–
18. Data 0+
20. Data 5–
21. Data 5+
23. Clock–
24. Clock+

Analogique
C1. Signal ROUGE
C2. Signal VERT
C3. Signal BLEU
C4. Synchro Horizontale
C5. Terre

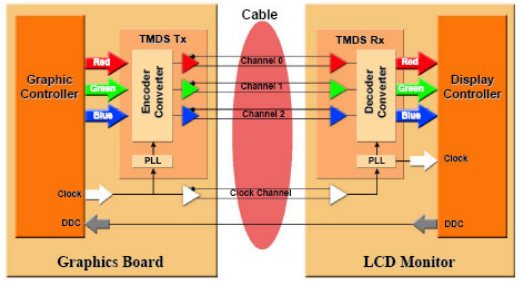
- type de connexion numérique reliant une carte graphique à un écran

Le connecteur DVI



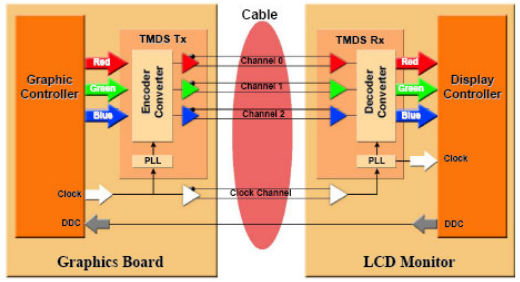
- La norme DVI 1.0 divise la transmission des données en trois segments : l'émetteur (la carte graphique), le câble d'acheminement du signal et le récepteur (l'écran).

Le connecteur DVI



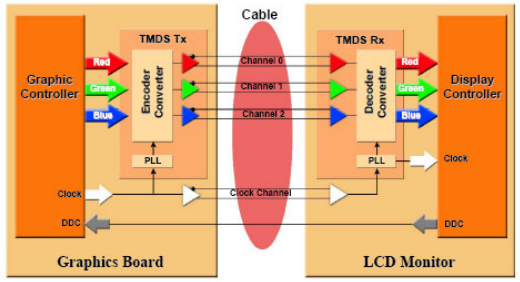
- La couleur de chaque pixel = trois composants (RVB)

Le connecteur DVI



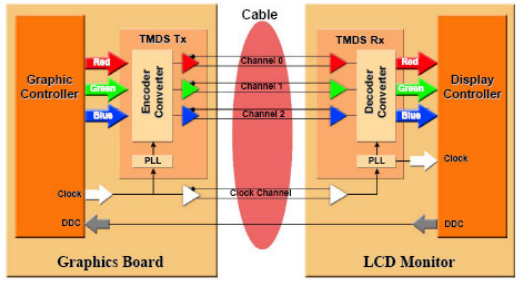
- Chaque couleur est déterminée par huit bits de données, ce qui permet d'avoir 256 teintes pour chacune des trois couleurs. En combinant les 256 teintes par couleurs, on peut donc afficher 16,7 millions de couleurs différentes.

Le connecteur DVI



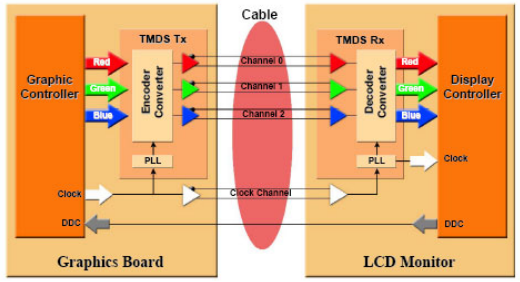
- La puce graphique génère les informations de couleur de chaque pixel sous la forme d'un flux de données de 24 bits (parallèle), soit 8 bits par couleur.

Le connecteur DVI



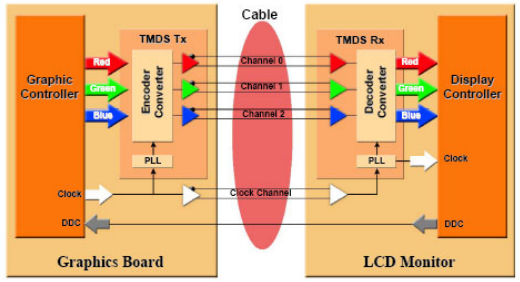
- A partir de ce flux de données, le RAMDAC crée les signaux VGA analogiques.

Le connecteur DVI



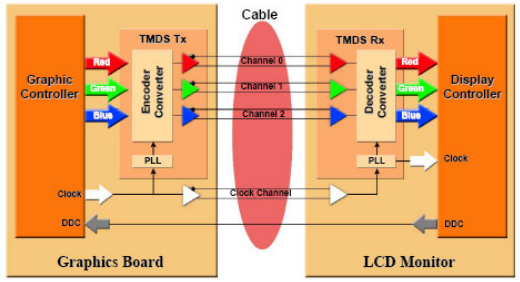
- Via l'interface DVI, les données parallèles sont envoyées à un émetteur TMDS, qui code les données en un signal série.

Le connecteur DVI



- Quand le signal atteint le récepteur (l'écran), le signal série est alors converti de nouveau en un flux de données parallèles.

Le connecteur DVI

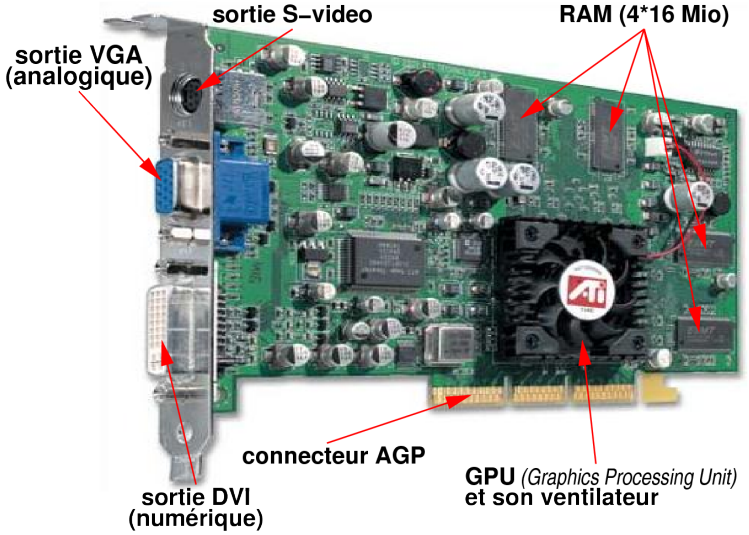


- La conversion en un flux de données en séries est nécessaire car les connexions séries sont moins sensibles aux interférences que les connexions parallèles, en particulier sur de grandes distances.

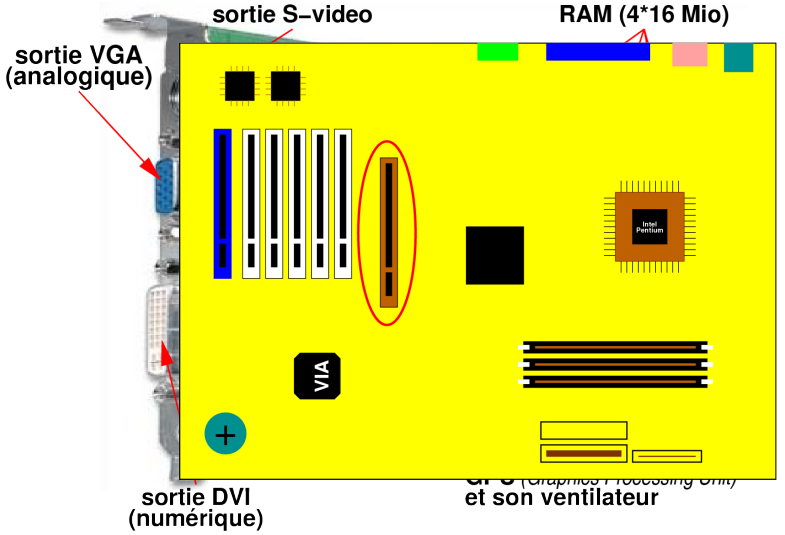
La carte video

- Carte video = carte fille déterminant :
  - La résolution et la profondeur de l'écran
  - La couleur de chaque pixel affiché
- Carte connectée sur le bus local grâce au connecteur AGP
- Mémoire video sur la carte : stocke la couleur de chaque pixel affiché
- Modification de l'écran :
  1. Le CPU annonce à la carte une modification de l'image
  2. La carte video recalcule le contenu de l'écran
  3. La nouvelle image est stockée dans la mémoire video
  4. Le contenu de la mémoire video est envoyé à l'écran

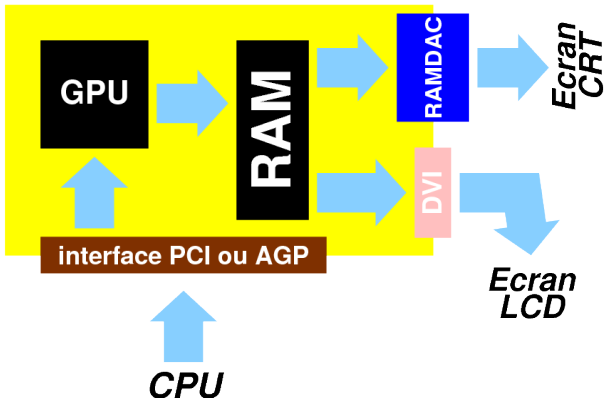
La carte video



La carte video



La carte video

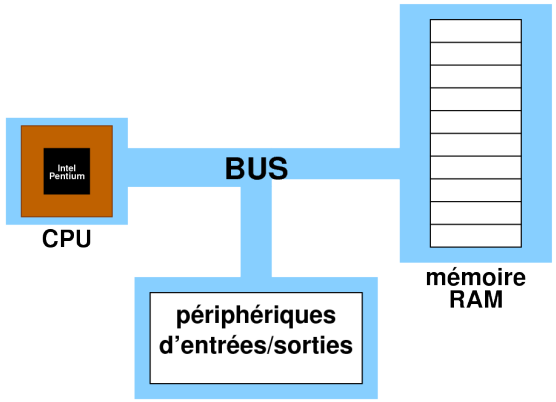


- RAMDAC : convertisseur numérique→analogique

Plan

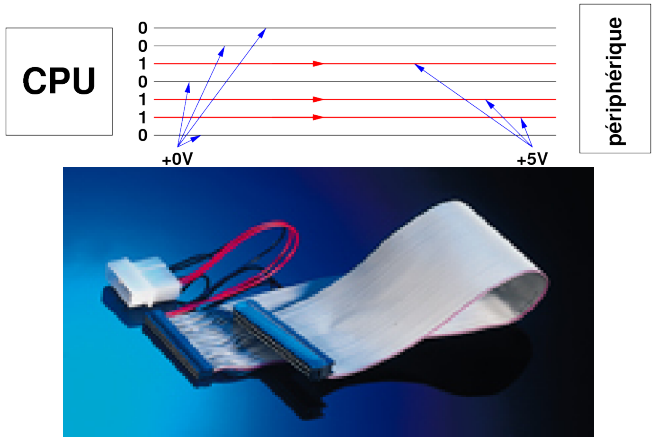
Le CPU et le « monde extérieur »

- Le processeur doit communiquer avec :
  - la mémoire RAM (*Random Access Memory*)
  - les périphériques d'entrées/sorties
- Moyens de ces communications : les *bus*



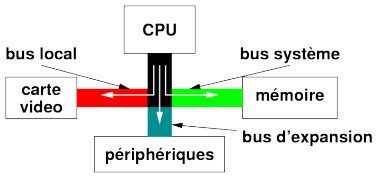
Les bus

- Définition d'un *bus* :
  - Collection de fils (ou de connections sur un circuit imprimé) transmettant de l'information



Les bus

- Bus système : Bus entre le CPU, la mémoire et le chipset *NorthBridge*
- Bus interne : Bus entre le CPU et les différents composants sur la carte mère
- Bus local : Bus très rapide relié directement au CPU (e.g. pour la video)
- Bus d'expansion : Bus reliant les périphériques externes à la carte-mère
- Les fils de chaque bus sont regroupés en 3 ensembles :
  - Le bus de données
  - Le bus d'adresses
  - Le bus de contrôle



Le bus de données

- Transporte les informations
- *Largeur du bus :* nombre de fils
  - (= nombre de bits transportés en parallèle)
- Pentium : bus de données de 64 bits
  - (transporte 8, 16, 32, ou 64 bits à la fois)

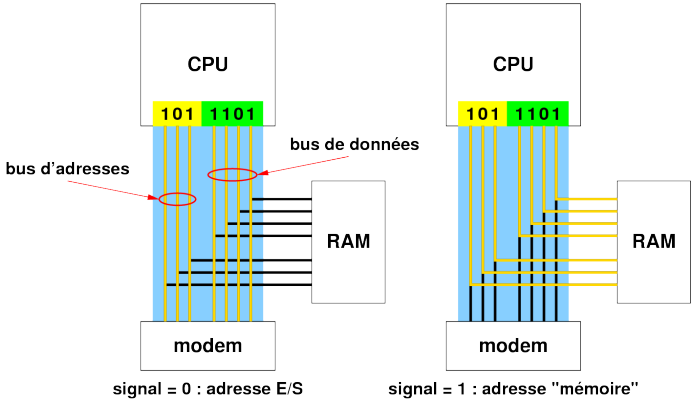
Mais : Pentium = machine 32 bits (*taille des registres*) !

Le bus d'adresses

- Adresse associée à tout élément susceptible de recevoir/donner de l'information :
  - Case mémoire dans la RAM
  - Case dans mémoire video
  - Périphérique d'entrées/sorties
  - ...
- Le CPU *voit* les périphériques comme de la mémoire
- Valeurs sur le bus d'adresses :
  - *Envoi d'information.* Adresse où déposer l'information
  - *Requête d'information.* Adresse où prendre l'information

Le bus d'adresses

- PC = 1 seul bus d'adresses *mais* 2 espaces d'adresses :
  - adresses « mémoires »
  - adresses d'entrées/sorties
- Un *signal* indique quel type d'adresse transite sur le bus

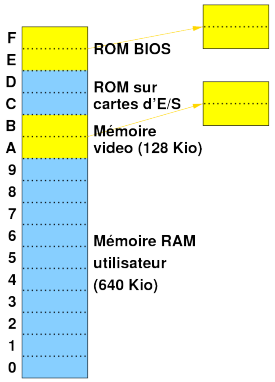


Le bus d'adresses

- Taille du bus (nombre de fils) : détermine le nombre d'adresses disponibles
  - i8086 : taille = 20 $\Rightarrow 2^{20} = 1$ Mio adressable
  - Pentium : taille = 32 $\Rightarrow 2^{32} = 4$ Gio adressables
  - Pentium Pro/II : taille = 36 $\Rightarrow 2^{36} = 64$ Gio adressables
- Mais : espace d'adresses d'entrées/sorties restreint à $2^{16} = 64$ Kio pour tous
- Problème : comment adresser les 32 Mio de mémoire d'une carte graphique ?
- espace d'adresses « mémoire » sert aussi à certains périphériques

Exemple : le 8086

- Utilisation de l'espace d'adresses « mémoires » du 8086 :
  - Bus de 20 fils $\Rightarrow$ 1 Mio adressable
  - Architecture segmentée : 16 segments de 64 Kio



- Attention ! Ne pas confondre :
  - Utilisation d'une adresse de l'espace d'adressage « mémoire »
  - Utilisation des cases de la mémoire RAM
- La mémoire video est *vue* comme une partie de la RAM dans les segments `A` et `B` mais l'adresse `A000 :0000` correspond *physiquement* à la mémoire sur la carte video

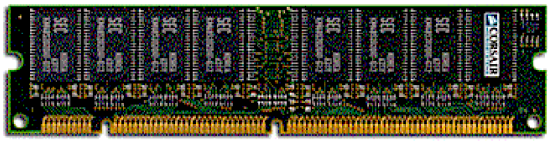
Le bus de contrôle

Transporte des signaux de contrôle des bus d'adresses et de données :

- Sens de l'échange sur le bus de données (lecture ou écriture ?)
- Volume de données échangées sur le bus de données : 8, 16, 32, 64 bits ?
- Type d'adresse transitant sur le bus d'adresses : E/S ou mémoire ?

La mémoire RAM

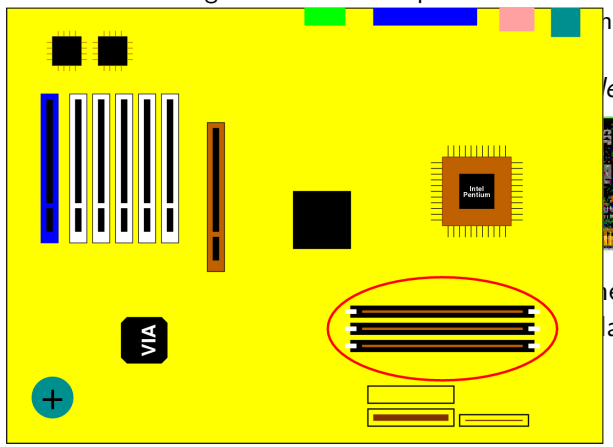
- RAM (*Random Access Memory*) :
  - Adressage direct et non séquentiel
  - Mémoire volatile : contenu disparaît à l'extinction du PC
  - Vision conceptuelle : tableau d'octets
- Barette SIMM (*Single In-line Memory Module*) :



- *Temps d'accès mémoire* : temps mis entre une demande d'opération (lecture/écriture) par le CPU et la fin de son exécution.

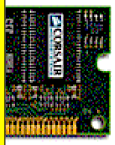
La mémoire RAM

- RAM (*Random Access Memory*) :
  - Adressage direct et non séquentiel



nction du PC

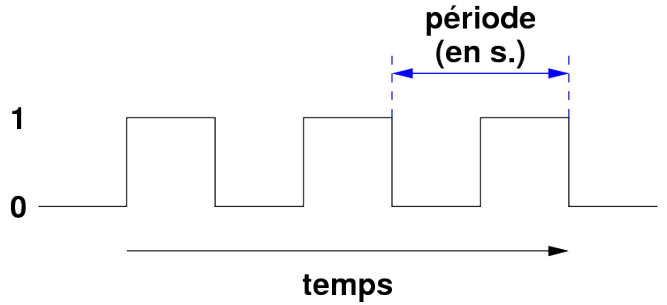
e) :



e demande
a fin de son

L'horloge

- Élément électronique créant un signal oscillant rapidement et régulièrement entre 2 valeurs

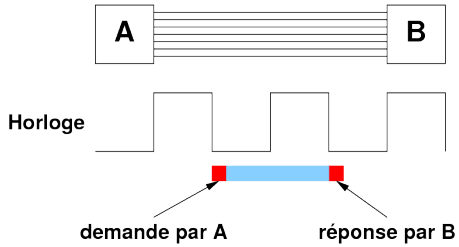


$$\text{fréquence} = \frac{1}{\text{période}}\ \text{Hz}$$

- Activité du CPU synchronisée avec le passage de 1 à 0 de l'*horloge interne*

Protocoles de transfert sur bus

- Bus *synchrone*

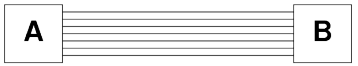


– Protocole simple/rapide
– Problème de synchro sur bus long
– Même fréquence pour A et B

Toutes les transactions synchronisées sur une horloge du bus

Protocoles de transfert sur bus

- Bus *synchrone*
- Bus *asynchrone*



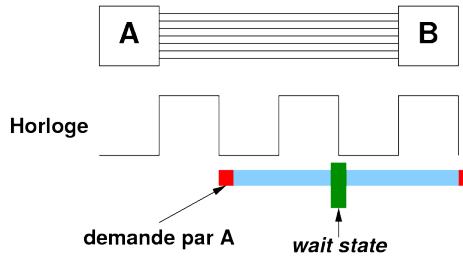
– A et B à fréquences différentes
– Pas de pb. synchro sur bus long
– Complexe => plus lent

Protocole d'échange :

– Demande par A
– Lecture par B de l'adresse et ACK
– A lit ACK et attend
– B place la donnée sur le bus et le signale
– A lit la donnée et ACK
– B lit ACK et libère le bus
– A libère le bus

Protocoles de transfert sur bus

- Bus *synchrone*
- Bus *asynchrone*
- Bus *semi-synchrone*

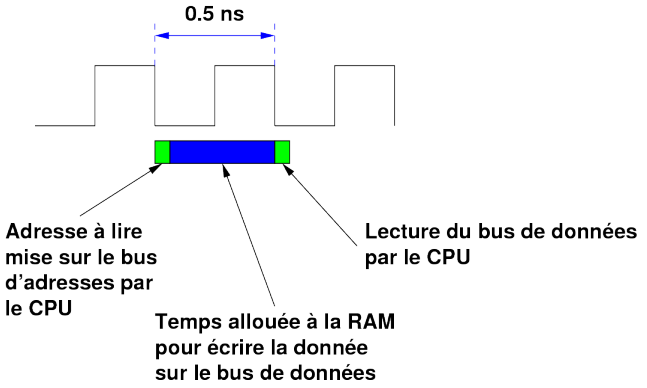


– Protocole simple/rapide
– Problème de synchro sur bus long
– A et B à fréquences différentes

Toutes les transactions synchronisées sur une horloge du bus
Une transaction peut prendre plus d'un cycle

Communications CPU/RAM

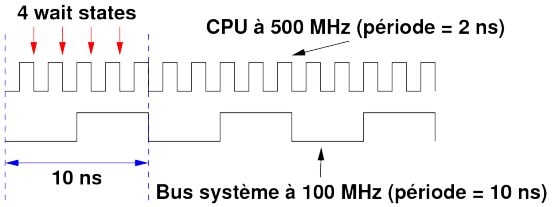
- Pentium à 2 GHz = 0.5 ns entre chaque opération
- Opération de lecture :



- Problème : temps d'accès de la RAM de l'ordre de 50 ns
  - Le CPU va trop vite pour la RAM
  - Connexion directe CPU/RAM impossible

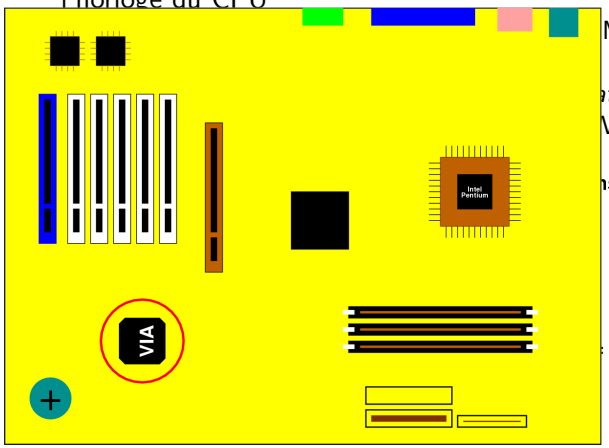
Communications CPU/RAM

- Échange de données à une cadence différente de celle de l'horloge du CPU
  - le bus système a sa propre horloge (66–133 MHz)
  - Protocole d'échange semi-synchrone
  - Insertion d'états d'attente du CPU (*wait states*)
  - Interface CPU/RAM : contrôleur système (*NorthBridge*)



Communications CPU/RAM

- Échange de données à une cadence différente de celle de l'horloge du CPU



Amélioration des performances

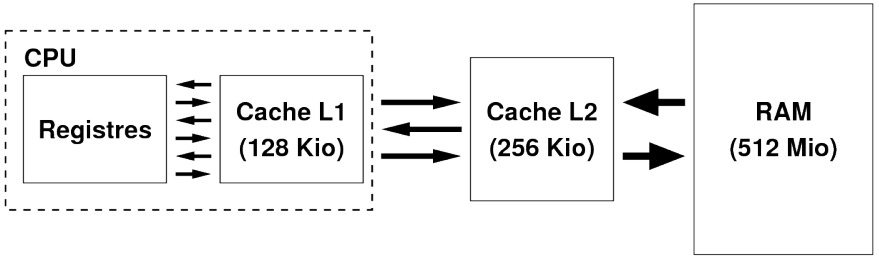
- Introduction des *wait states* : coûteux en temps
- Élimination des *wait states* ?
  - Utilisation de mémoire plus rapide (cher)
  - Utilisation des propriétés des programmes

```
for i :=1 to 10 do
 T[i] := 0;
```

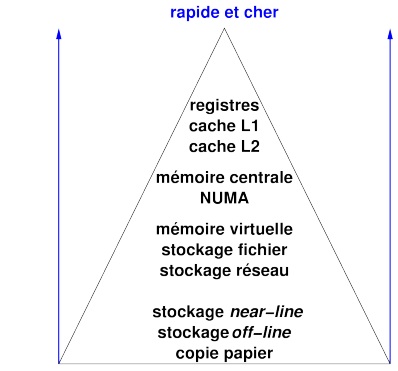
- Localité temporelle des références :
  - `i` est accédé répétitivement
- Localité spatiale des références :
  - Toutes les cases consécutives en mémoire représentant `T` sont accédées

La mémoire cache

- Idée : utiliser de la mémoire très rapide en petite quantité pour stocker les cases de la RAM récemment accédées ou susceptibles de l'être bientôt
  - Mémoire cache de niveau 1 (cache L1) sur le CPU
  - Mémoire cache de niveau 2 (cache L2) à côté de la RAM



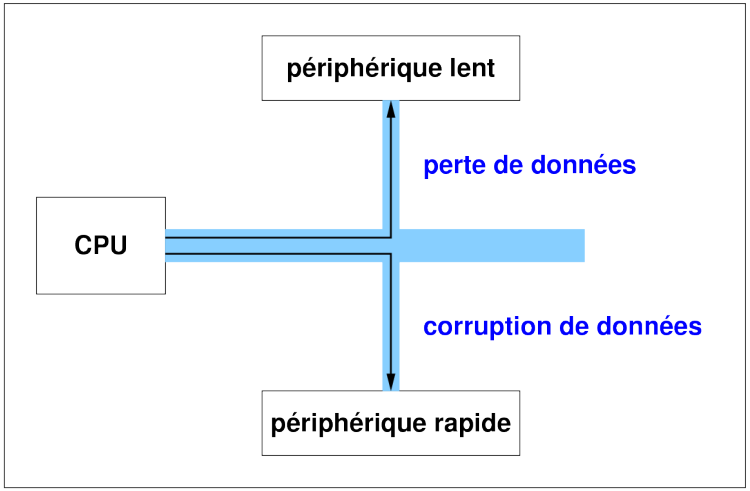
Hiérarchie des mémoires



NUMA : *Non–Uniform Memory Access* (e.g. mémoire video)
near–line : jukebox automatisé
off–line : montage manuel

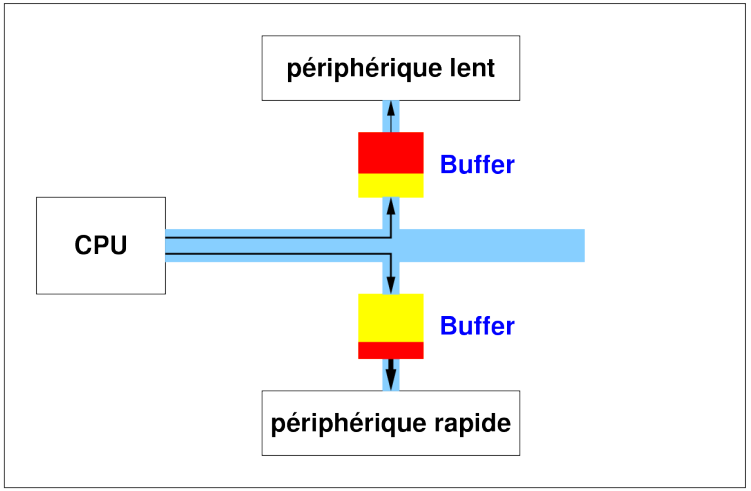
Buffering

- Communications entre le CPU et un périphérique lent/rapide ?



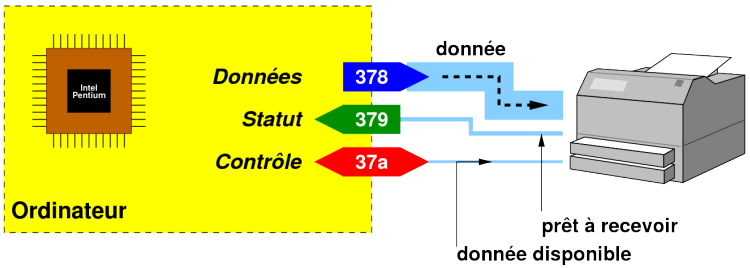
Buffering

- Communications entre le CPU et un périphérique lent/rapide ?



Handshaking

- Alternative/complément au *buffering* pour un périphérique lent
- Le périphérique utilise un ou plusieurs ports d'E/S.
- Exemple (l'imprimante) :
  1. Écriture de la donnée sur le port 378
  2. Lecture du port 379 pour savoir si l'imprimante est prête
  3. Envoi d'un signal sur le port 37a pour indiquer qu'une donnée est disponible



Bus d'expansions

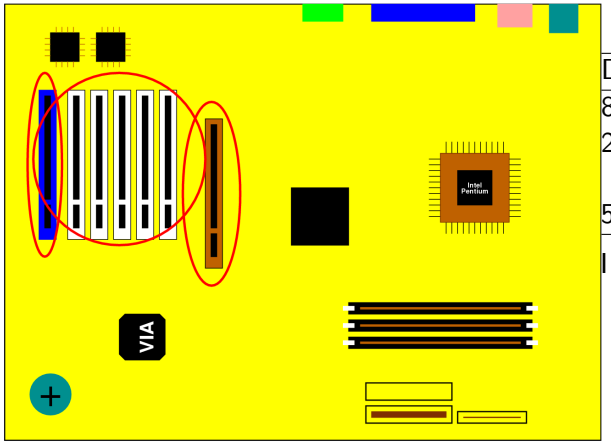
- Relient les périphériques externes entre eux et à la carte mère
- Plusieurs standards :

Nom	Usage	Débit en pointe
ISA	cartes sons, modems	8.3 Mio/s
PCI	cartes graphiques, cartes sons, modems, ...	266 Mio/s
AGP	cartes graphiques	528 Mio/s

- Le CPU doit négocier pour utiliser le bus PCI avec les autres périphériques
- Aucune négotiation pour utiliser le bus AGP

Bus d'expansions

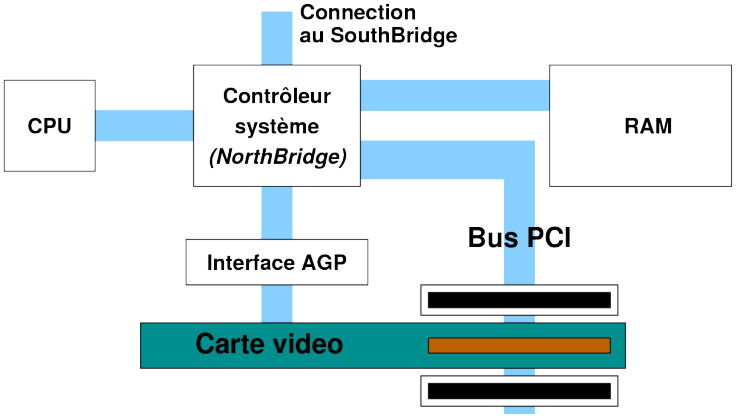
- Relient les périphériques externes entre eux et à la carte mère



Débit en pointe
8.3 Mio/s
266 Mio/s
528 Mio/s

l avec les autres

Organisation



Chipset NorthBridge :

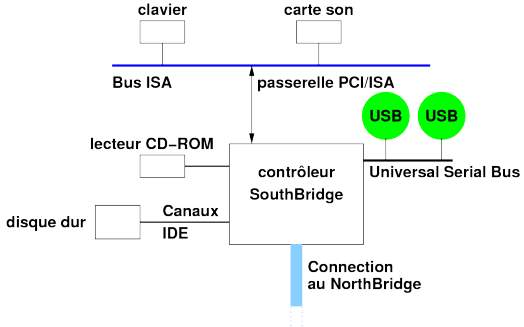
- Contrôle de l'accès à la mémoire
- Contrôle de l'accès au CPU

Autres bus

- Connection des disques durs et lecteurs de CD-ROM ?
- Gestion du bus ISA ?
- Gestion du bus USB (pour connection de petit matériel) ?

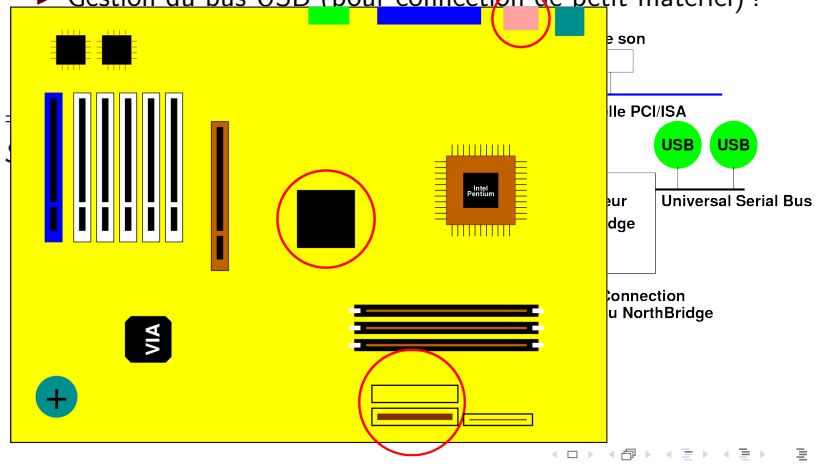
⇒ Contrôleur *SouthBridge*

- Contrôle de tous les bus secondaires

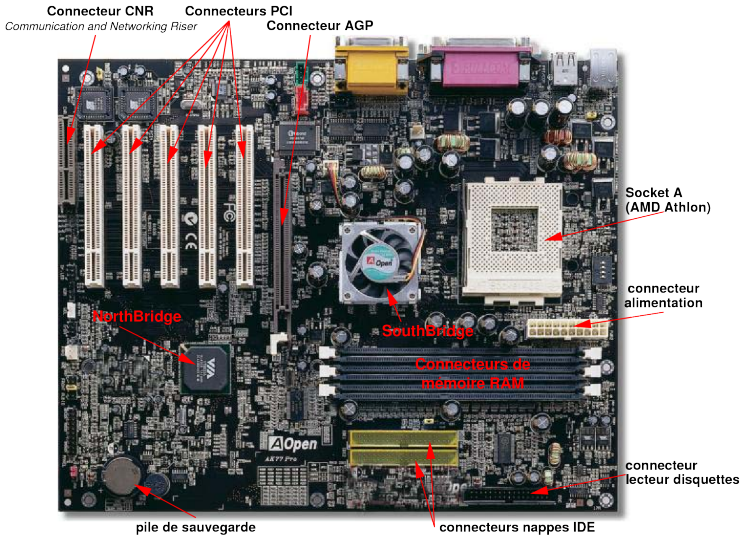


Autres bus

- Connection des disques durs et lecteurs de CD-ROM ?
- Gestion du bus ISA ?
- Gestion du bus USB (pour connection de petit matériel) ?



Retour sur la carte-mère ATX



Accès aux bus

- Bus partagé par plusieurs systèmes
- Qui contrôle qui peut « parler » sur le bus ?
- Deux possibilités :
  - Un maître et des esclaves
    - le CPU décide de tout.
    - Inconvénient : doit participer à toutes les transactions
  - Plusieurs maîtres
    - Arbitrage pour décider du maître temporaire
    - Problèmes à considérer : temps, complexité, *équité*

Plan

Communication avec les périphériques

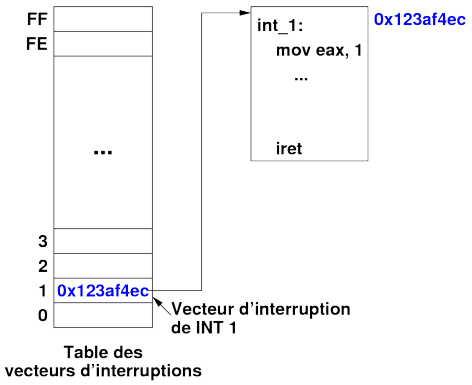
- Communications entre le CPU et le clavier, la souris, . . . ?
  - Comment savoir qu'une information est disponible ?
- Deux méthodes :
  - *Polling* (scrutation permanente)
    - Le CPU regarde régulièrement l'état du périphérique d'E/S
    - $\Rightarrow$ coûteux
  - *Interruption*
    - L'E/S interrompt le CPU pour lui signaler un évènement

Les interruptions

- Trois types :
  - *Exceptions*
    - Indication de débordement de pile, division par zéro, . . .
  - *Interruptions matérielles*
    - Demande par un périphérique de l'attention du processeur
  - *Interruptions logicielles*
    - Services offerts par le BIOS ou le système d'exploitation
- Sur le PC : 256 interruptions possibles (`INT 0x0` à `INT 0xFF`)

Les interruptions

- Association d'une routine de traitement à chaque interruption
- Indirection : vecteur d'interruption *i* pour `INT` *i*
- Possibilité de changer le traitement d'une interruption par modification du vecteur d'interruption



Les interruptions matérielles

- Générées par des périphériques
- Signalées au processeur par un signal sur un fil spéciale : IRQ (*Interrupt Request Line*)
- PC : 16 IRQ (Exemple : IRQ 1 = clavier, IRQ 13 = FPU)
- Arrivée d'une interruption matérielle :
  - *Interruption masquable* :
    - le processeur décide ou non de la prendre en compte
    - Masquage des interruptions : indicateur `IF` dans le mot d'état
  - NMI (*Non Maskable Interrupt*) : Obligation de s'arrêter et de traiter l'interruption

Liste des IRQs

Numéro	Usage typique	Numéro	Usage typique
IRQ 0	Horloge interne	IRQ 8	Horloge
IRQ 1	Clavier	IRQ 9	*Aucun*
IRQ 2	Réservé	IRQ 10	*Aucun*
IRQ 3	COM 2	IRQ 11	*Aucun*
IRQ 4	COM 1	IRQ 12	Souris PS/2
IRQ 5	Carte son	IRQ 13	FPU
IRQ 6	Lecteur disquettes	IRQ 14	IDE 1
IRQ 7	LPT 1	IRQ 15	IDE 2

Les interruptions logicielles : aperçu

- Interruptions générées explicitement par un programme
- Appel d'une routine : `int xx` avec `xx` l'index de la routine dans la table des vecteurs d'interruptions
- Accès à des services de manipulation des périphériques
- Pour chaque service : sous-services déterminés par les registres

Instructions associées

- Instructions assembleurs :
  - `int i` : empile `eflags` avant d'empiler `eip` et de brancher à `[IDT+4×i]` (IDT = adresse de la table des vecteurs)
  - `iret` : dépile `eip` et `eflags`
- Exemple d'utilisation (sous Linux) :

```
 segment .data
chaine : db "Hello world",0xa
len equ $ - chaine
  segment .text
 ...
 mov ebx, 1 ; écriture sur stdout = 1
 mov ecx, chaine ; ecx <- ptr. sur début de chaine
 mov edx, len ; edx <- nb. d'octets à écrire
 mov eax, 4 ; service 4 = sys_write
 int 0x80
; au retour, eax contient le nombre d'octets effectivement écrits
 mov eax, 1 ; service 1 = sys_exit
 int 0x80
```


Instructions associées

- Instructions assembleurs :
  - `int i` : empile `eflags` avant d'empiler `eip` et de brancher à `[IDT+4×i]` (IDT = adresse de la table des vecteurs)
  - `iret` : dépile `eip` et `eflags`
- Exemple d'utilisation (sous Linux) :

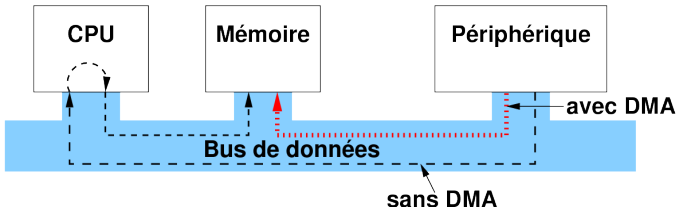
Sous Linux:
Seule l'interruption 0x80 est accessible au programmeur

```
 segment .data
chaine : db "Hello world",0xa
len equ $ - chaine
  segment .text
 ...
 mov ebx, 1 ; écriture sur stdout = 1
 mov ecx, chaine ; ecx <- ptr. sur début de chaine
 mov edx, len ; edx <- nb. d'octets à écrire
 mov eax, 4 ; service 4 = sys_write
 int 0x80
; au retour, eax contient le nombre d'octets effectivement écrits
 mov eax, 1 ; service 1 = sys_exit
 int 0x80
```

Quelques interruptions logicielles

- Int 0x10 : services de la carte video
  - Exemple : passage en mode graphique 320 × 200 × 16
```
mov al, 0xd ; passage en 320x200x16
mov ah, 0x0 ; sélection du service "choix du mode"
int 0x10 ; appel de l'interruption 0x10
```
  - Int 0x16 : services du clavier
  - Int 0x21 : services du systèmes d'exploitation MS-DOS
  - Int 0x33 : services de la souris
  - Int 0x80 : services de Linux

Le mode DMA (Direct Memory Access)

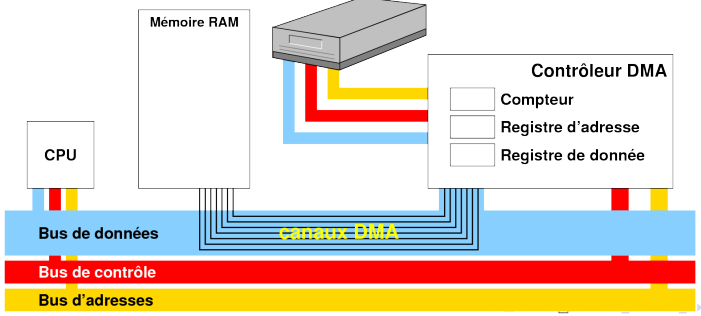


- Communication directe entre le périphérique et la mémoire sans utiliser le CPU
- Le CPU reste libre pour faire autre chose en parallèle
- Mais : compétition pour l'utilisation du bus de données

Utilisation du mode DMA

Lecture/écriture sur un disque :

1. CPU positionne le disque à l'endroit voulu pour la lec./écr.
2. CPU met dans le *registre d'adresse* l'adresse RAM où écrire/lire
3. CPU initialise le *compteur* avec le nombre d'octets à lire/écrire
4. CPU lance la demande : données transférées entre le disque et la mémoire à travers le *registre de donnée* sans intervention du CPU
5. En fin de transfert : envoi d'une interruption au CPU

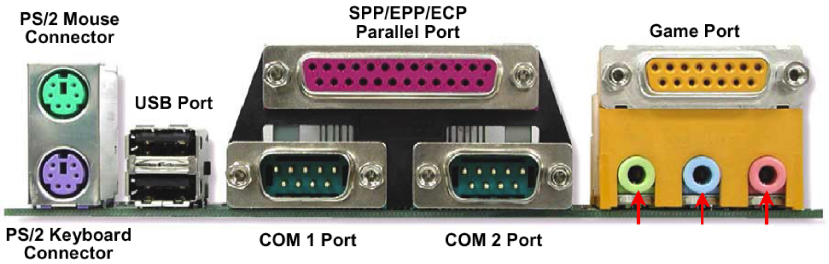


IRQ vs. Plug and Play

- *IRQ* :
  - Seulement 16 IRQ
    (moins, en retirant les IRQs réservées)
  - Affectation statique d'une IRQ à un périphérique
  - Utilisation d'une IRQ pour plusieurs périphériques
    - ⇒ ne peuvent être utilisés en même temps
- *Plug and Play* :
  - Gestion par le système de la table d'allocation des adresses de ports d'E/S, des canaux DMA et des IRQs
  - Affectation dynamique d'un canal/adresse/IRQ à chaque périphérique PnP en fonction des besoins

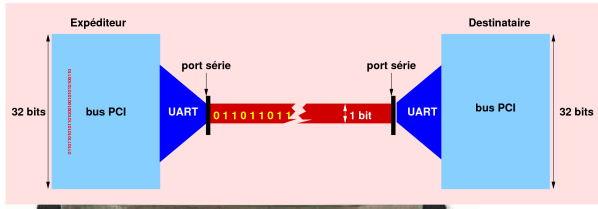
Le bus USB

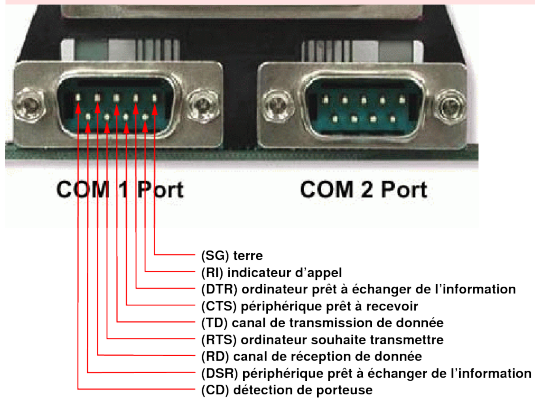
- USB (*Universal Serial Bus*)
  - Utilisé pour connecter de nombreux périphériques (souris, disque externe, graveur, imprimante ...)
  - Débit maximum : 1.5/12/480 Mbps (USB 2.0)
  - Périphériques chaînables et *hot-swappables*
  - Alimentation : 500 mA (à partager) sous 5 V
  - Échanges sur le bus contrôlés par un hôte (pas *peer-to-peer*)
  - Protocole d'échanges prédéfini par le standard



Le port série RS-232

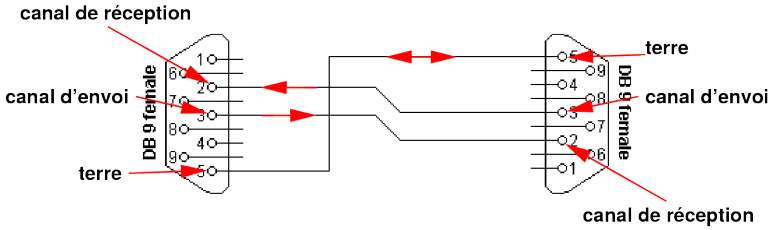
- Communication asynchrone (pas d'horloge)
- $0 \equiv +3 \rightarrow +5$ V et $1 \equiv -25 \rightarrow -3$ V
  - Grande amplitude $\Rightarrow$ grandes distances (de l'ordre de la dizaine de m)
- Informations envoyées en série sur une seule ligne
- UART (*Universal Asynchronous Receiver/Transmitter*) remet les données en parallèle





Le port série RS-232

- Contrôle de flot :
  - *Logiciel* (caractères Xon/Xoff—ASCII 17/19) : receveur envoie Xoff pour arrêter la transmission et Xon pour la redémarrer
  - *Matériel* (handshaking par RTS/CTS) : envoyeur met RTS à 1 ; receveur met CTS à 1 pour accepter l'envoi.
- Exemple (communication *null modem* à 3 fils) :

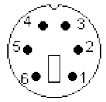


Le clavier

- Connection *bi-directionnelle* par un connecteur DIN ou PS/2
- L'ordinateur a priorité sur la direction de l'échange
- Protocole d'échange série synchrone (horloge déterminée par clavier)
- Clavier peut envoyer une donnée si *horloge*=1 et ligne *donnée*=1
- Si *horloge*=0 et *donnée*=1 : clavier *bufferise* les frappes
- Si *horloge*=0 et *donnée*=0 : clavier s'apprête à recevoir de l'info de l'ordinateur

Connecteur DIN

1. Horloge clavier
2. Donnée
4. Terre
5. +5V

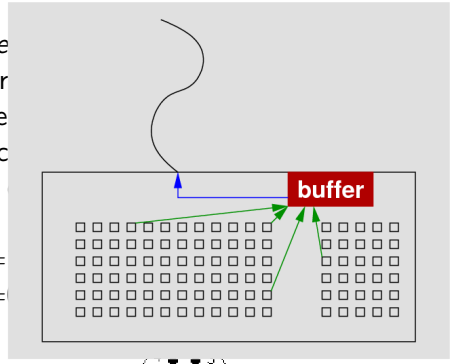


Connecteur PS/2

1. Horloge clavier
2. Terre
3. Donnée
5. +5V

Le clavier

- Connection *bi-directionne*
- L'ordinateur a priorité sur
- Protocole d'échange série (horloge déterminée par c
- Clavier peut envoyer une *donnée*=1
- Si *horloge*=0 et *donnée*=
- Si *horloge*=0 et *donnée*= l'info de l'ordinateur



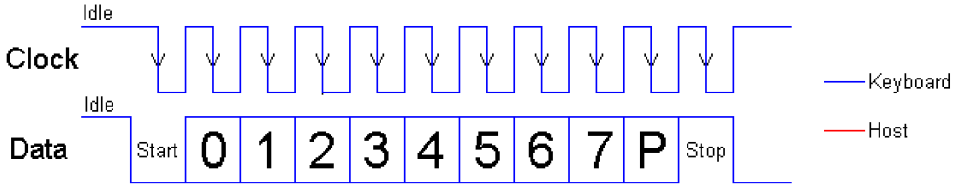
Connecteur DIN

1. Horloge clavier
2. Donnée
4. Terre
5. +5V

Connecteur PS/2

1. Horloge clavier
2. Terre
3. Donnée
5. +5V

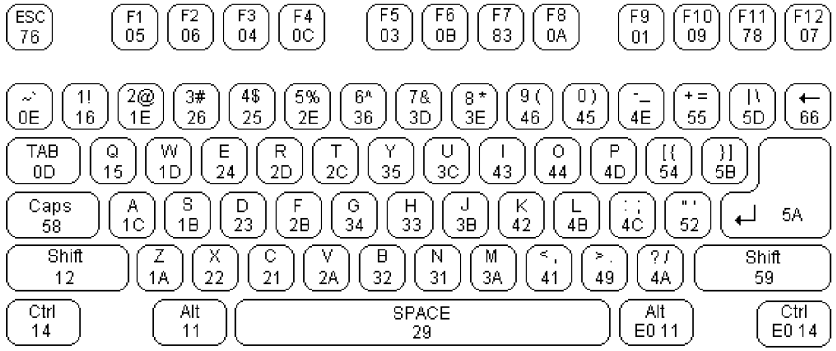
Clavier ⇒ ordinateur



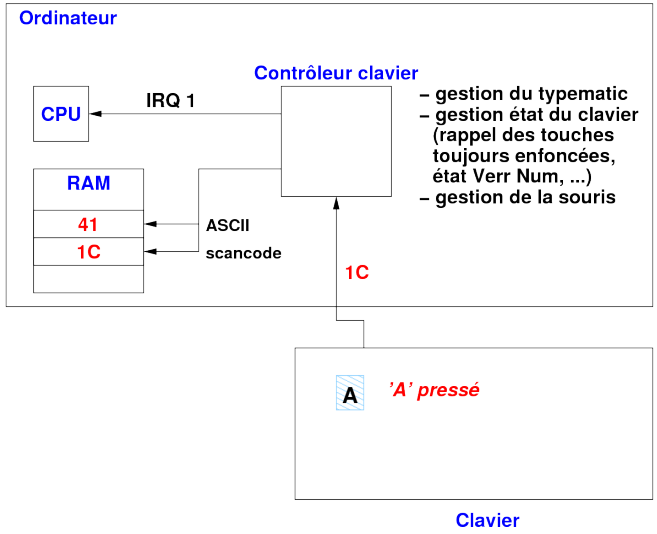
- Start bit : valeur 0
- Stop bit : valeur 1
- Bit P : bit de parité (0 si nombre impair de 1 dans bits 0–7)
- Lecture des bits sur front descendant
- LSB envoyé en premier
- Fréquence de l'horloge : 20–30 kHz

Le scancode

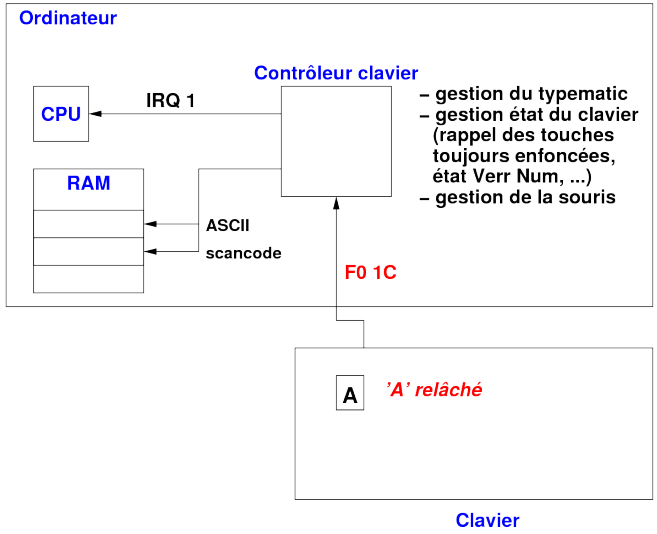
- Données envoyées par le clavier ?
- Le *scancode*



Le contrôleur de clavier



Le contrôleur de clavier

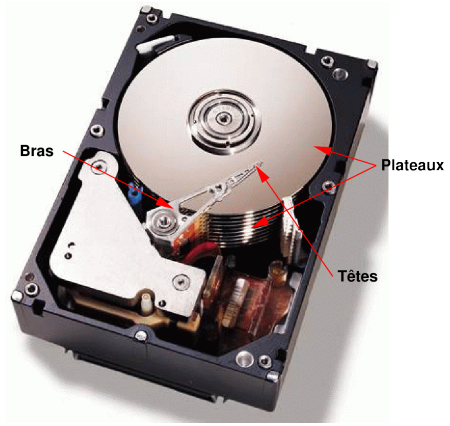


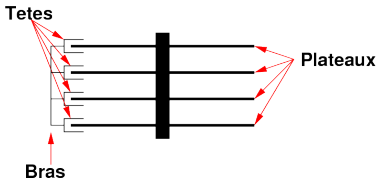
La souris

- Même protocole d'échange que le clavier
- Informations transmises :
  - État des boutons (pressés ou non)
  - Valeur absolue du déplacement en X et en Y depuis le dernier envoi
  - Direction du mouvement en X et Y

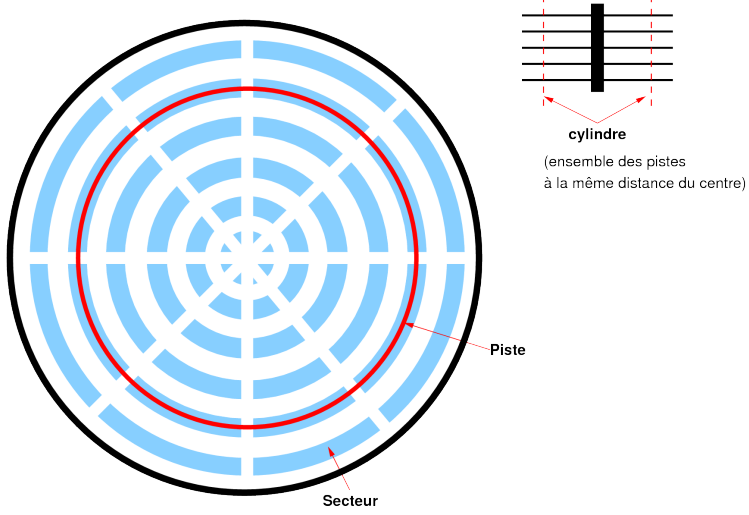
Le disque dur

- Rotation rapides des plateaux (7500–15000 tours/mn)



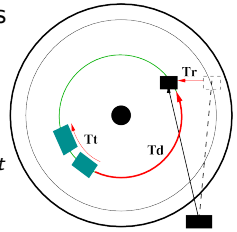


Les plateaux



Performances du disque

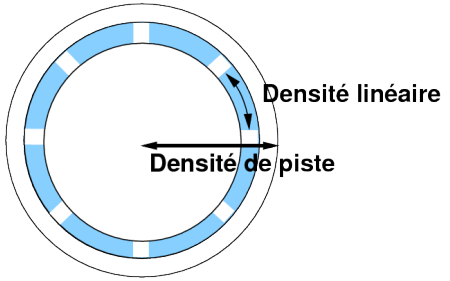
- *Temps de recherche* : temps requis pour déplacer le bras au-dessus de la piste choisie $T_r = s + m \times n$
  avec s le temps de démarrage, m une constante dépendant du disque et n le nombre pistes traversées
- *Délai rotationnel* : temps nécessaire pour arriver au-dessus du secteur choisi (en moyenne, la moitié du temps nécessaire à une révolution) $T_d = \frac{1}{2\omega}$
  avec ω le nombre de rotations par secondes
- *Temps de transfert* : $T_t = b/(\omega \times N)$
  avec b le nombre d'octets à transférer,
  N le nombre d'octets par piste
- *Temps d'accès moyen* : $T_a = T_r + T_d + T_t$



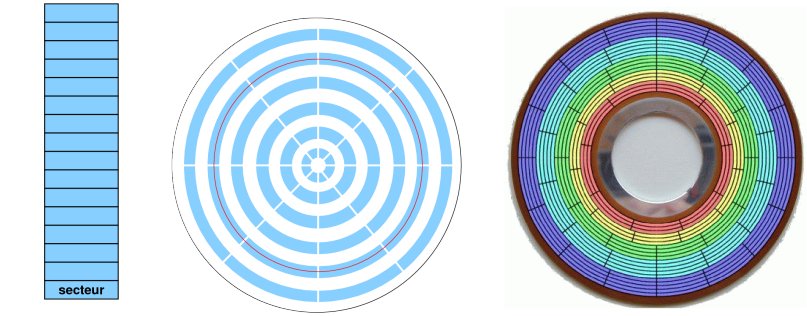
Densités des disques

- *Densité de piste* D_p : nombre de pistes par pouce de rayon (1 pouce = 2.54 cm)
- *Densité linéaire* D_l : nombre de bits par pouce de secteur
- *Densité surfacique* D_s :

$$D_s = D_p \times D_l \text{ bits/pouce}^2$$



Vision logique/physique



- Nombre de bits par secteur constant + taille des secteurs croissante de l'intérieur vers l'extérieur ⇒ densité linéaire décroît.
- *Zoning* : augmentation du nombre de secteurs/piste vers l'extérieur pour conserver une densité linéaire maximale

Formatages

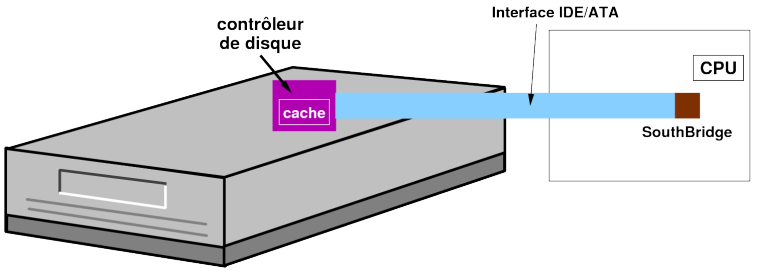
- *Formatage bas niveau* :
  - Création des pistes et des secteurs sur un plateau vierge
  - Ajout des informations de positionnement des secteurs/pistes
  - Enregistrement des secteurs défectueux
  - Indépendant du S.E.
- *Partionnement* : découpage du disque en volumes physiques (e.g. `C :`, `D :` sous Windows)
- *Formatage haut niveau* : écriture de la structure de fichiers (dépendant du S.E.)

Interface disque/PC

- Interface entre le disque et l'extérieur : *contrôleur*
- Accès physique à une donnée très long (déplacement de la tête + lecture)
  - ⇒ présence d'un cache dans le disque
- *Lecture en avant* : lors d'une requête de lecture, mise dans le cache des secteurs se trouvant autour
- *Cache des écritures* : factorisation des requêtes d'écriture pour éviter de bouger sans cesse la tête

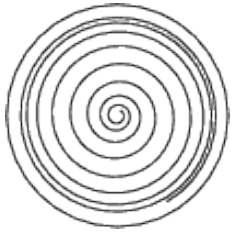
L'interface IDE/ATA

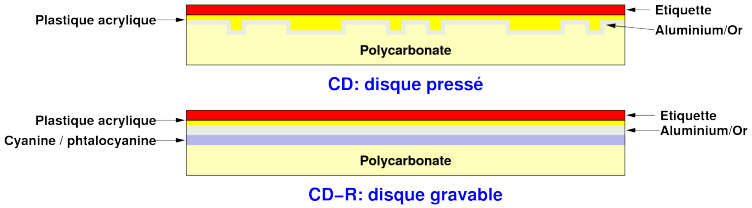
- Deux canaux
  - ⇒ connection pour 2 éléments : un maître et un esclave
- Bus de données de 16 bits
- ATAPI (*AT Attachment Packet Interface*) : connection de CDROM, lecteur de bande, ...



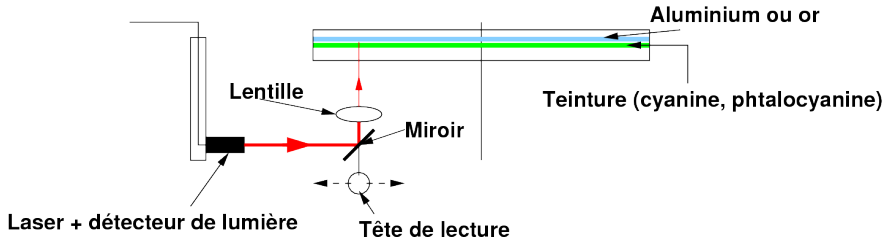
Le CD-ROM

- CD-ROM : conservation numérique des données (0 et 1)
- Une seule piste en spirale (préformée) de l'intérieur vers l'extérieur
- Lecture/écriture optique par laser
- CDs classiques : pressés
- CD-R : gravés
  - Teinture sensible à la chaleur
  - ⇒ chauffé ⟶ absorbant (0)
  - ⇒ non chauffé ⟶ transparent (1)



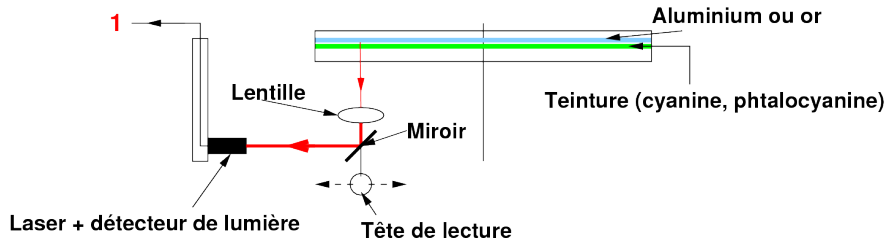


Lecture d'un CD-R



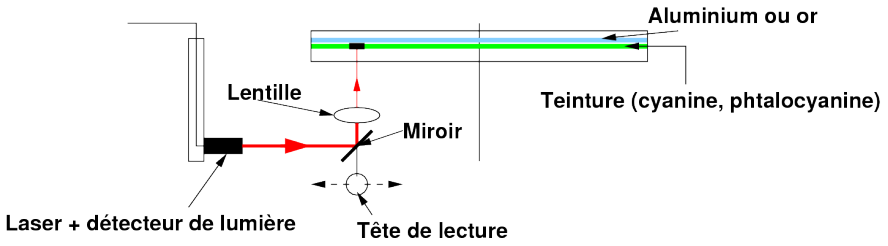
Teinture intacte : réflexion du faisceau Laser

Lecture d'un CD-R



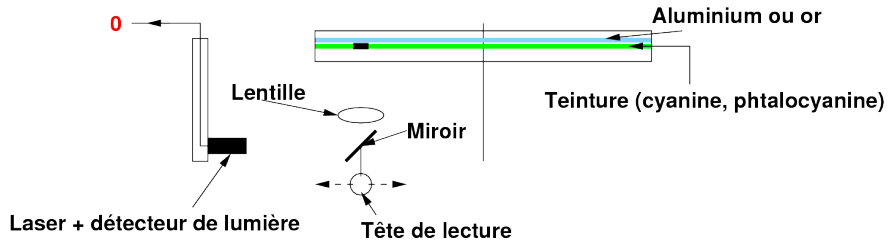
Teinture intacte : réflexion du faisceau Laser

Lecture d'un CD-R



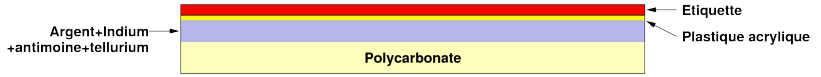
Teinture chauffée : absorption du faisceau Laser

Lecture d'un CD-R



Teinture chauffée : absorption du faisceau Laser

Le CD-RW

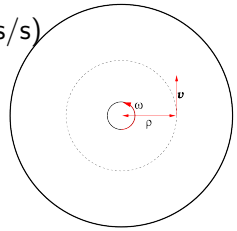


CD–RW: disque réinscriptible

- Le complexe *Ag-In-Sb-Te* a deux états :
  1. état polycristallin réfléchissant
  2. état amorphe absorbant
- Écriture : chauffage à 500–700 °*C* d'un point
  $\rightsquigarrow$ état amorphe en refroidissant (=création d'un 0)
- Effacement : chauffage à 200 °*C* de toute la spirale
  $\rightsquigarrow$ état polycristallin en refroidissant
- Réflectance de *Ag-In-Sb-Te* moins bonne que l'aluminium
  $\Rightarrow$ erreurs de lecture sur certains lecteurs de CD-R

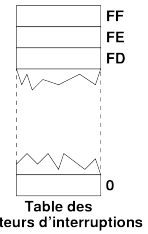
Caractéristiques du lecteur de CD

- Vitesse de base de lecture/écriture : $1\times = 153\,600$ o/s (75 blocs de 2048 octets/s)
- Graveur $12 \times 8 \times 32\times$:
  - CD-R à 12 fois la vitesse de base
  - CD-RW à 8 fois la vitesse base
  - Lecture à 32 fois la vitesse de base
- Rappel : $v = \omega\rho$, avec v la vitesse linéaire, ω la vitesse angulaire et ρ le rayon
- CAV (*Constant Angular Velocity*) : vitesse de rotation constante (e.g. disque dur)
  - $\Rightarrow$ Vitesse linéaire décroît de l'extérieur du disque vers l'intérieur
  - $\Rightarrow$ Volume d'informations lues par secondes non constant
- CLV (*Constant Linear Velocity*) : vitesse de rotation du disque dépend de l'endroit où se trouve la tête (plus rapide près du centre)
  - $\Rightarrow$ Assure un débit constant d'information



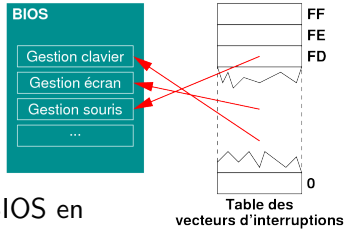
Le BIOS (Basic Input/Output System)

- Ensemble de routines en ROM pour contrôler :
  - le *boot* du système
  - le clavier
  - l'écran
  - les disques
- Au démarrage : chargement du BIOS en mémoire et modification de la table des vecteurs d'interruptions pour l'accès aux services offerts
- *Flash BIOS* : BIOS sur EEPROM pour mise à jour
- BIOS utilisé sous MS-DOS et Windows, pas sous Linux



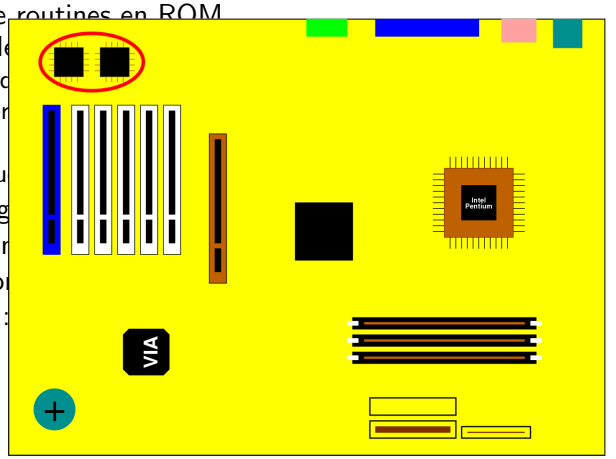
Le BIOS (Basic Input/Output System)

- Ensemble de routines en ROM pour contrôler :
  - le *boot* du système
  - le clavier
  - l'écran
  - les disques
- Au démarrage : chargement du BIOS en mémoire et modification de la table des vecteurs d'interruptions pour l'accès aux services offerts
- *Flash BIOS* : BIOS sur EEPROM pour mise à jour
- BIOS utilisé sous MS-DOS et Windows, pas sous Linux



Le BIOS (Basic Input/Output System)

- Ensemble de routines en ROM pour contrôle
  - le *boot* d
  - le clavier
  - l'écran
  - les disqu
- Au démarrag
  mémoire et r
  d'interruptio
- *Flash BIOS* :
- BIOS utilisé

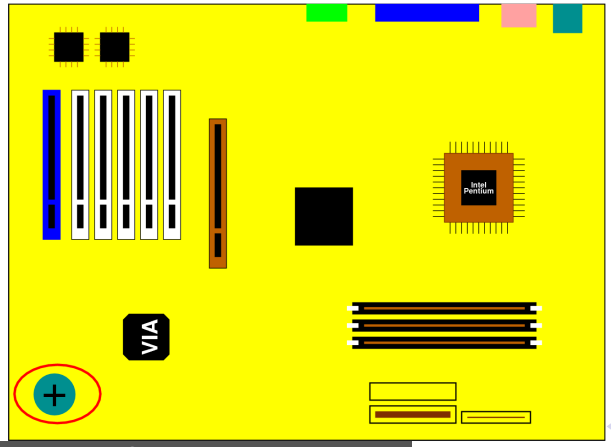


Le boot

- Démarrage de l'ordinateur :
  - POST (*Power on Self Test*)
  - Saut automatique à la mise sous tension à l'adresse 0xFFFF0 en mémoire (cases occupées par le BIOS)
  - Exécution du BIOS de la carte video (adresse fixe : 0xC000)
  - Recherche des autres périphériques à des adresses pré-déterminées
  - Recherche d'un disque contenant un *système d'exploitation* à charger :
    - Sur disque dur : lecture du MBR *Master Boot Record*, cylindre 0, tête 0, secteur 1
    - Le BIOS passe à la main au code du MBR
      ⇒ Chargement du système d'exploitation

Le CMOS

- *Complementary Metal Oxide Semiconductor*
- Circuit alimenté par une pile bouton
- Stocke la configuration du PC + l'heure + la date
- Durée de la pile alimentant le CMOS : ≈ 10 ans



Technologie du matériel

Architecture des ordinateurs

Guillaume Blin

IGM-LabInfo UMR 8049,
Bureau 4B066
Université de Marne La Vallée
gblin@univ-mlv.fr
http://igm.univ-mlv.fr/~gblin