Plan

- **Représentation des nombres**
- **Circuits logiques**
- **Unité Arithmétique et Logique**
- **Notions de temps et de mémorisation**
- **Contrôle et jonction des composants**
- **Evolution des ordinateurs – Historique**
- **Un microprocesseur simple**
- **Programmation d'un microprocesseur**
- **Système complet**
  - **Système matériel**
  - **Système d'exploitation**
- **Les microprocesseurs actuels**
- **Exploitation de la performance des microprocesseurs**

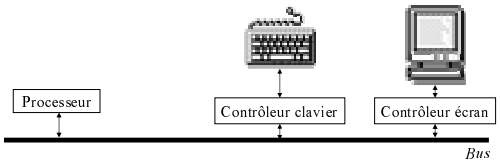
Système Matériel et Système d'Exploitation

- Système matériel = processeur + périphériques.
- Système d'exploitation = logiciel permettant:
  - d'offrir une abstraction du système matériel à l'utilisateur
  - de gérer l'ensemble des ressources du système matériel
- Principales ressources du système matériel:
  - Processeur:
    - Quel processus doit s'exécuter ?
    - Pendant combien de temps ?
  - Entrées/Sorties (disque, réseau, clavier, écran…):
    - Comment le processeur communique avec un périphérique ?
  - Mémoire:
    - Où placer données et programmes pour chaque utilisateur ?
    - Comment répartir la mémoire entre les utilisateurs ?

Système Multi-Utilisateur

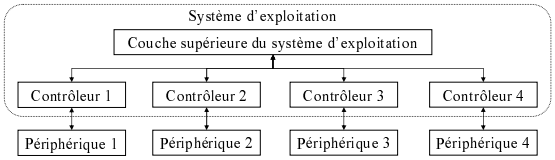
- En apparence, plusieurs processus sont exécutés simultanément sur le même processeur.
- En fait, le processeur est utilisé pendant des fractions de temps (quantum) par chaque processus.
- Un processus devenu inactif sera remplacé par un processus en attente.
- La file d'attente des processus est ordonnée par le système d'exploitation:
  - Date et durée du dernier quantum
  - Propriétaire du processus
  - …
- Le passage d'un processus à l'autre est un **changement de contexte**.
- Pour pouvoir revenir au processus, il faut sauvegarder l'état du processeur:
  - Registres généraux
  - PC, pointeur de pile, PSW (*Program Status Word*: mode noyau/utilisateur)…
- Le contexte est sauvegardé sur la pile du système d'exploitation.

Entrées/Sorties (*Input/Ouput*)



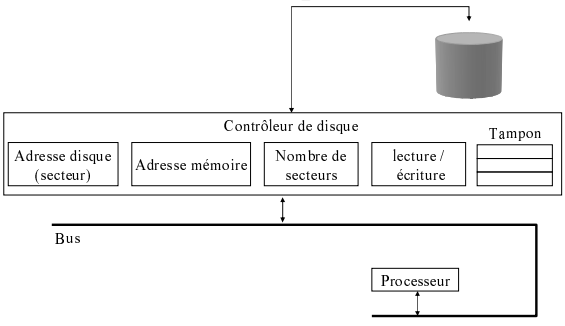
- Le contrôleur commande le périphérique et en masque, en partie, le fonctionnement interne.
- Les principales commandes sont: **lecture** et **écriture**. Les transferts se font par octet ou par blocs d'octets (*burst*).
- Les commandes sont passés par le biais de **registres spécifiques d'entrée/sortie**, et les données par le biais de **tampons**.
- Registres et tampons peuvent être des zones de la mémoire principale:
  - pas d'instruction spécifique,
  - accès périphérique ≈ accès mémoire.

Drivers



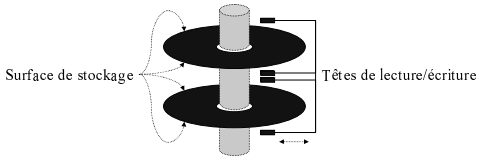
- L'ajout de nouveaux périphériques à un système matériel doit induire des modifications minimales du système d'exploitation.
- Autant que possible, les spécificités matérielles liées à la commande d'un périphérique sont encapsulées dans un ***driver***, un composant logiciel ajouté au système d'exploitation.
- Schématiquement, le système d'exploitation veut essentiellement lire et écrire des octets aux périphériques.
- Le système d'exploitation fournit une interface standardisée pour connecter les *drivers* et envoyer des commandes génériques.

Exemple



- Registres d'entrées/sorties pour le contrôleur de disque.

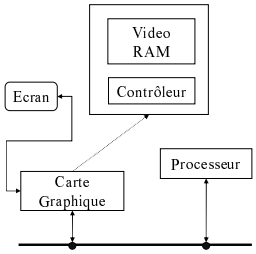
Exemple



- Dans un disque, l'information est divisée en cylindres, pistes, secteurs, grâce à plusieurs plateaux et têtes de lecture.
- Le contrôleur peut fournir une vue abstraite du disque: *N* secteurs contigus.
- La commande *lire k secteurs à partir du secteur i et envoyer à l'adresse A* est interprétée par le contrôleur:
  - *i* est converti en cyclindre, piste (tête), secteur
  - la tête de lecture est déplacée vers la piste
  - la tête de lecture attend le passage du secteur
  - l'information est renvoyée sous forme d'une suite d'octets

Exemple

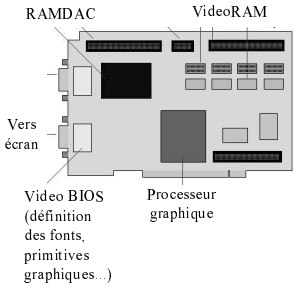
- Périphérique de visualisation.
- La carte graphique contient une zone mémoire (VRAM) accédée par le processeur par le biais du contrôleur.
- A un pixel de l'image est associé de 1 à 24 bits (*bitmap*)
  - 1 bit: N/B
  - 24 bits: 3x8 bits pour RGB (*Red Green Blue*)
  - 8/16 bits: index d'une palette de couleurs (table de valeurs RGB)
- Exemple: affichage de graphiques
  - sans accélérateur graphique:
    - le processeur calcule l'image *bitmap* et l'envoie au contrôleur (occupation du bus).
  - avec accélérateur (la plupart des cartes PC aujourd'hui):
    - le système d'exploitation envoie une commande abstraite (dessiner un rectangle)
    - API (e.g., *DirectX*) puis traduction dans *driver* de la carte
    - la carte graphique convertit la commande en bitmap et la stocke dans la VRAM.
    - le *RAMDAC* (*Random Access Memory Digital Analog Converter*) transforme l'image en mémoire en image à l'écran.



Exemple

- Rôle croissant de la partie graphique:
  - Initialement: tout est dans le CPU
  - Puis: 2D dans processeur graphique, 3D dans CPU
  - Puis: rendu 3D (création image à partir des polygones) dans processeur graphique, géométrie 3D (manipulation des polygones) dans CPU
  - Récemment: rendu+géometrie dans processeur graphique
- Puissance processeurs graphiques proche puissance processeurs généralistes
- Risque de basculement des systèmes autour du processeur graphique

⇒ Augmenter les capacités graphiques des processeurs généralistes (MMX, SSE, 3DNow, Altivec...).



Utilisation du Périphérique

- Test continuel du périphérique (*polling*):
  1. appel d'une routine spécifique par l'utilisateur,
  2. les registres d'entrée/sortie sont lus (ou modifiés),
  3. le processeur vérifie régulièrement si les données sont arrivées dans le tampon (ou ont été consommées),
  4. les données sont recueillies (ou envoyées).

  ⇒ Le processeur est fréquemment sollicité.
- Grande différence de performance entre périphériques et processeur:
  - Exemple: ADSL ≈ 64ko/s → 1 octet / $1{,}53 \times 10^{-5}$s; sur processeur à 1GHz, ≈ 15300 cycles entre deux octets.

Programmation d'une Entrée/Sortie

- Test continuel:
  - KBDR: code de la touche frappée
  - KBSR: registre d'entrée (zone mémoire)
    - KBSR[15] = 1 si une touche a été frappée mais KBDR non encore lu
    - KBSR[15] = 0 lorsque KBDR est lu

```
DEBUT   LDI   R1, @KBSR   ; Lecture du registre
        BRzp  DEBUT       ; d'état.
        LDI   R0, @KBDR   ; Lecture du code de la touche.
        BR    SUITE
@KBSR   .FILL xF400
@KBDR   .FILL xF401
```

  - CRTDR: code ASCII du caractère à afficher
  - CRTSR: registre de sortie (zone mémoire)
    - CRTSR[15] = 1 si caractère à afficher mais non encore utilisé
    - CRTSR[15] = 0 lorsque CRTDR est lu

```
DEBUT   LDI   R1, @CRTSR   ; Lecture du registre
        BRzp  DEBUT        ; d'état.
        STI   R0, @CRTDR   ; Envoi du code ASCII
        BR    SUITE
@CRTSR  .FILL xF400
@CRTDR  .FILL xF401
```

Utilisation du Périphérique

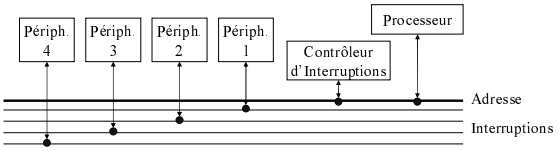
- Interruptions
  1. appel d'une routine spécifique,
  2. les registres d'entrée/sortie sont modifiés,
  3. le processeur peut exécuter d'autres tâches,
  4. lorsque les données arrivent (lecture), le contrôleur avertit le processeur par le bus que les données sont prêtes,
  5. le processeur interrompt éventuellement la tâche en cours et lit les données.

Programmation d'une Entrée/Sortie

- Interruptions.
- Test de la présence d'une interruption à la fin de chaque instruction.
- Le processeur doit autoriser les interruptions (bit d'autorisation).
- Envoi de l'adresse de la routine de service par le périphérique.
  - Si KBSR passe à 1, interruption du processeur, sauvegarde du contexte (PC, bits de condition), envoi index de routine de service, activation routine de service, puis restauration du contexte (RTI).

```
DEBUT           LDI     R0, @KBDR       ; Lecture du code de la touche.
                RTI
@KBDR  .FILL    .FILL   xF401
```

Gestion de Plusieurs Périphériques



- Plusieurs interruptions simultanées possibles.
- Un périphérique signale une requête d'interruption sur une ligne dédiée du bus.
- Le contrôleur gère les priorités entre interruptions et envoie une requête d'interruption au processeur en signalant le périphérique retenu (numéro d'interruption); e.g., par le bus d'adresse.

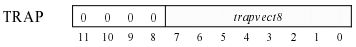
Exemple

- *Plug And Play* sur un PC:
  - Initialement, chaque périphérique (carte) avait un niveau d'interruption (0 à 7) et une adresse fixe pour ses tampons (exemple: niveau 1 pour le clavier et adresses 0x60 à 0x64).

  ⇒ Conflits possibles entre périphériques.

  ⇒ On a donné à l'utilisateur la possibilité de changer le numéro d'interruption et l'adresse des tampons.
  - *Plug And Play*:
    - les périphériques ont des numéros d'interruption et des adresses de registre programmables,
    - au démarrage, le système recueille la liste des périphériques et les interruptions souhaitées (fixes pour les anciens matériels),
    - puis assigne les interruptions aux périphériques.

Simplification et Protection de l'Accès aux Périphériques



- Accès aux routines du système similaire à appel de procédure (passage de procédure et retour de valeur): TRAP/RET.
- Empêcher l'utilisateur d'accéder directement aux registres d'entrée/sortie (protection, erreurs de manipulation des périphériques).
- Table de correspondance indexée par **vecteur d'interruption**.

Numéro d'inter-ruption	Adresse routine d'inter-ruption
0	0x3000
1	0x3100
2	0x4F01
3	0x2A0B
...	...

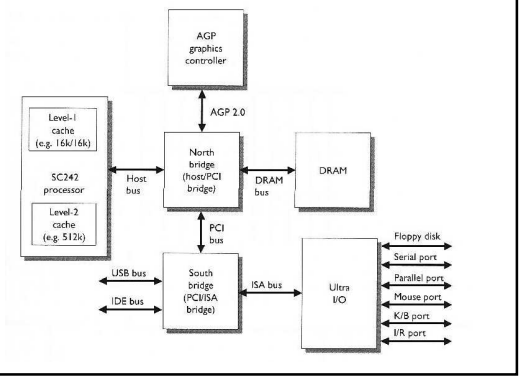
Communication Directe Entre Périphériques

- Accès direct du périphérique à la mémoire (*DMA: Direct Memory Access*):
  1. le processeur indique au **contrôleur DMA** l'adresse en mémoire, l'adresse sur le périphérique, le nombre d'octets, le type d'accès (lecture/écriture) et le périphérique utilisé pour le transfert,
  2. le contrôleur DMA enclenche la communication avec la périphérique par le biais du bus, comme s'il s'agissait du processeur,
  3. une fois le transfert terminé, le contrôleur DMA déclenche une interruption pour le processeur.

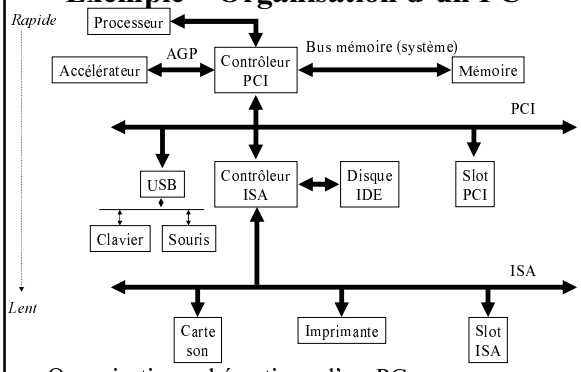
  ⇒ le processeur n'est pas sollicité pendant le transfert,

  ⇒ en cas d'accès au bus pendant le transfert, le processeur peut être ralenti ou bloqué.
- Exemples d'utilisation:
  - accès disque dur,
  - visionner un film sur un PC à partir d'un DVD.

Exemple – Organisation d'un PC



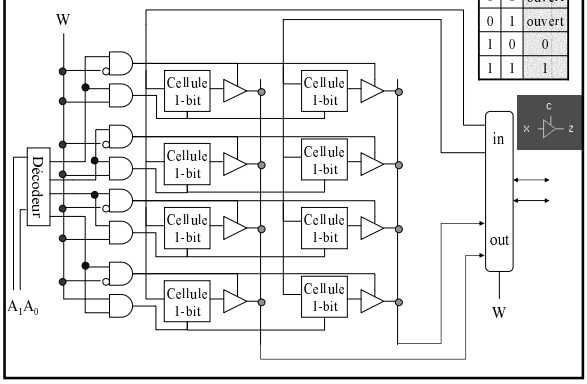
Exemple – Organisation d'un PC



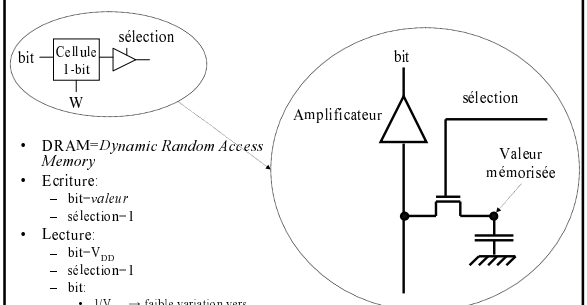
- Organisation schématique d'un PC

Mémoire Physique

C	X	Z
0	0	ouvert
0	1	ouvert
1	0	0
1	1	1



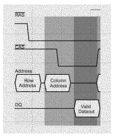
Cellule 1-bit DRAM

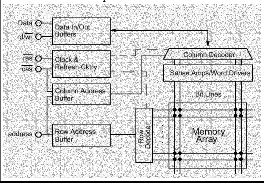


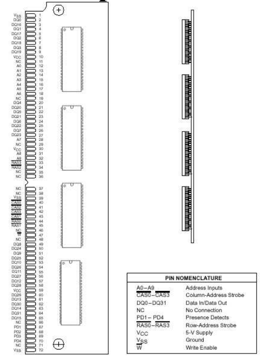
- DRAM=*Dynamic Random Access Memory*
- Ecriture:
  - bit=*valeur*
  - sélection=1
- Lecture:
  - bit=V_{DD}
  - sélection=1
  - bit:
    - $1/V_{DD}$ → faible variation vers V_{DD}
    - $0/V_{SS}$ → faible variation vers V_{SS}
  - écriture bit

Composant DRAM simple

- Réduction du nombre de pins → multiplexage de l'adresse.
- Perte au niveau des capacités → rafraîchissement périodique (≈ qq ms).
- Diverses organisations possibles:
  - Exemple: fournir 8 bits par cycle:
    - 8 composants 1-bit, 4 composants 2-bits…



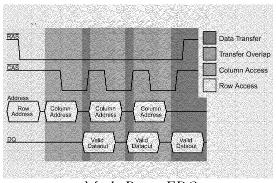




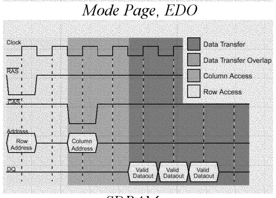
SIMM (*Single In-line Memory Module*)
4Mo: 4 chips 1Mbit x 8; 32 bits à la fois

SDRAM

- Mode page:
  - Maintenir l'adresse de la ligne (*row*)
  - Changer uniquement l'adresse de colonne (*column*)
  - ⇒ Accès rapide aux bits de la même ligne (octets consécutifs).
  - Ajout d'un latch en sortie: maintien de la valeur de sortie → changement de l'adresse de colonne plus rapide (pendant lecture); EDO (*Extended Data Out*).
- SDRAM:
  - Les signaux CAS et RAS sont synchronisés avec l'horloge CPU → pas de perte liée à la synchronisation.
  - Registre indiquant le nombre de bits à lire: très efficace sur les adresses consécutives.



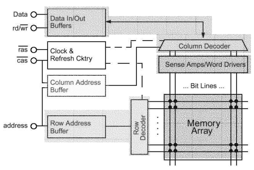
Mode Page, EDO



SDRAM

Tendances

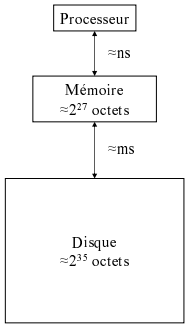
- Débit amélioré par SDRAM, DDR *(Double Data Rate)*…
- Latence élevée; écart croissant avec temps de cycle processeur:
  - RAS/CAS,
  - Longueur des lignes d'adresse et de données,
  - Amplification.
- Quelques chiffres:
  - Exemple: 64Mbits,
  - bits groupés par paquets de 16.



Type de DRAM	# lignes	# col.	# bits transférés	Taille buffer de ligne	# bancs internes	Fréquence	Tps préchgt.	Tps accès ligne	Tps accès col.	Tps transfert
Standard /EDO	4096	1024	16	16Kb	1	-	40ns	15ns	30ns	15ns
SDRAM	4096	256	16	4Kb	4	100MHz	20ns	30ns	30ns	10ns

Mémoire Virtuelle

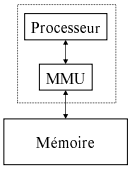
- Taille des mots aujourd'hui: 32 ou 64 bits.
- Espace d'adressage potentiel: 4 gigaoctets ou 16 exaoctets (2^{60}).
- Taille mémoire aujourd'hui: ≈128Mo sur un PC, ≈1Go sur une station.
- Taille mémoire physique << Taille mémoire adressable.
- Besoins en mémoire de l'ensemble des processus d'un PC ou d'une station >> taille mémoire physique.
- **Mémoire virtuelle**:
  - Donner à l'utilisateur l'illusion qu'il dispose d'une mémoire de $2^{\text{taille du mot en bits}}$ octets.
  - La mémoire physique contient les données les plus récemment et fréquemment accédées,
  - Les autres données sont stockées sur le disque,
  - Notion de hiérarchie mémoire,
  - En général un processus ne consomme qu'une fraction de l'espace d'adressage disponible → capacité du disque suffisante.

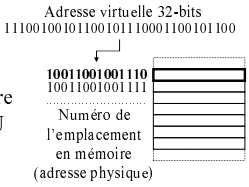


Hiérarchie Mémoire

MMU

- Le processeur utilise des adresses *virtuelles*.
- Une donnée située à l'adresse *virtuelle* A_v est placée en mémoire *physique* à l'adresse A_p
  ⇒ il faut convertir A_v en A_p avant d'émettre la requête,
  – le composant qui assure cettre traduction s'appelle la MMU (*Memory Management Unit*)





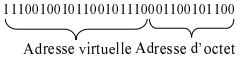
Pages

- Pour chaque processus et pour chaque adresse virtuelle utilisée, il faut conserver la traduction virtuelle/physique
  ⇒ il est plus efficace de regrouper les adresses par bloc et de conserver une traduction d'adresse par *bloc* plutôt que par *octet*.

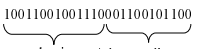
⇒ On découpe la mémoire virtuelle et la mémoire physique en **pages**: blocs d'octets (≈ de 512o à 64Ko selon les systèmes)

- La latence d'accès à un disque est élevée
  ⇒ il est plus efficace de ramener les données par bloc d'octets plutôt que par octet
- Adresse d'un octet = (adresse d'une page, adresse de l'octet dans la page)
  – l'adresse de l'octet dans la page est la même pour l'adresse virtuelle et pour l'adresse physique.

Mot de 32-bits, pages de 4Ko



Adresse virtuelle de la page | Adresse d'octet dans la page



Adresse physique de la page | Adresse d'octet dans la page

– la traduction de la partie basse de l'adresse n'est pas nécessaire
⇒ le nombre de traductions à conserver est divisé par $2^{\text{taille de page en octet}}$.

Traduction d'Adresse

- Les traductions sont elles-mêmes découpées en pages et conservées en mémoire dans une table (*Page Tables*).
- Comment trouver une traduction ?
  - l'emplacement du début de la table est connu,
  - on ajoute l'adresse virtuelle de page,
  - on récupère l'adresse physique à l'emplacement calculé.

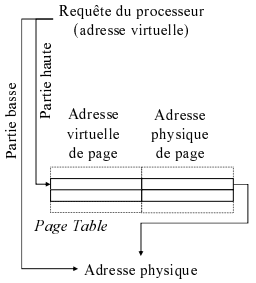


Table de Pages

- Mot de 32 bits, pages de 4Ko → 2^{20} entrées dans la table ≈ 4 Mo (≈ 4 octets/entrée).
- Mot de 64 bits, pages de 4Ko → 2^{52} entrées dans la table ≈ 4 Petaoctets
  - Espace utilisé très inférieur mais quand même impossible de conserver toute la table en mémoire,
  - ⇒ Ne conserver en mémoire que les entrées utilisées
  - ⇒ Table à 2 (ou +) niveaux
    - 1er niveau: page de traduction disponible ? & emplacement
    - 2ème niveau: pages de traduction

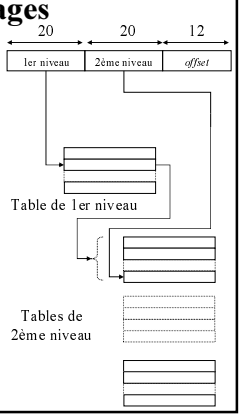
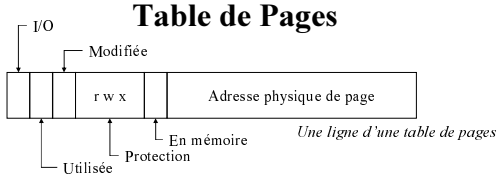


Table de Pages

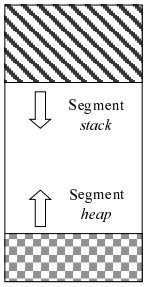


Une ligne d'une table de pages

- Traduction physique/virtuelle,
- Mais aussi:
  - protection entre processus (lecture, écriture, exécution)
    - un processus ne peut accéder (sciemment ou par erreur) aux pages en mémoire physique d'un autre processus (sauf en cas de partage explicite)
  - informations pour la politique de remplacement des pages
    - le système d'exploitation conserve en mémoire les pages les plus utilisées
    - si une page à enlever a été modifiée, il faut d'abord l'écrire sur le disque
  - information de disponibilité (en mémoire)
    - si la page requise n'est pas en mémoire, il y a une **faute de page**
  - information pour les entrées/sorties.
- Taille d'une entrée de la table des pages ≈ 32 bits.

Segmentation

- Une approche à la fois complémentaire et différente.
- Pour chaque processus, l'espace d'adressage est divisé en plusieurs zones de stockage, parfois de taille variable:
  - la pile des données locales (*stack*)
  - la pile des données allouées dynamiquement (*heap*)
  - le programme
  - ...
- On associe un **segment** à chacune de ces zones.
- Deux implémentations possibles:
  - *k* bits parmi les *n* bits d'adresse indiquent le numéro du segment,
  - *k* bits supplémentaires indiquent le numéro du segment; un emplacement mémoire est spécifié par le couple (numéro de segment, adresse).
    - on reproduit le principe de la mémoire virtuelle en donnant l'illusion à chaque zone de stockage qu'elle dispose de l'ensemble de l'espace d'adressage
- On peut combiner mémoire virtuelle (pages) et segmentation (*Alpha, Pentium,...*) ou utiliser la segmentation seule (*8086*).



Espace d'adressage virtuel de l'Alpha