



Architecture des ordinateurs

La Mémoire

Question préliminaire

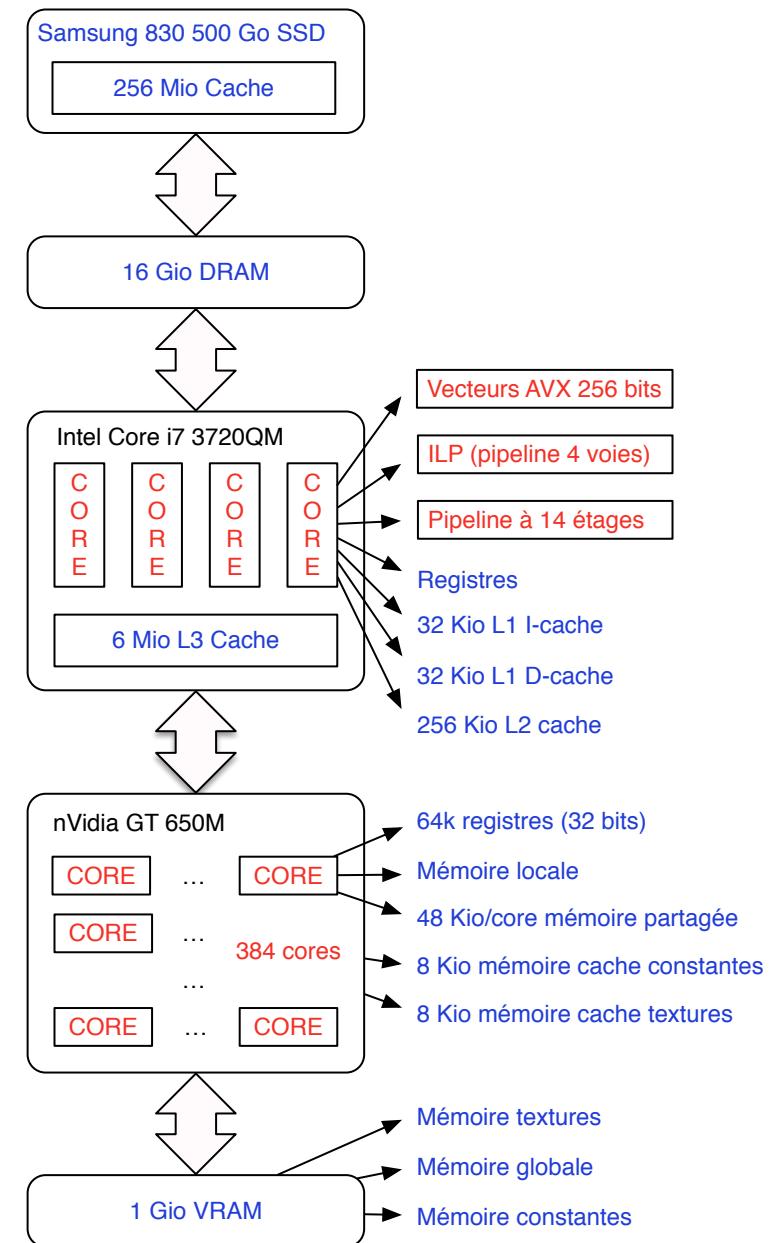
À votre avis, combien de types de mémoire possède un bon ordinateur portable ?

Pas loin de 20 !!!

Analyse d'une architecture moderne



- 2.7 milliards de transistors
- **17 types de mémoire**
- 5 types de parallélisme
- Au moins 4 modèles de programmation + APIs



Importance du système mémoire

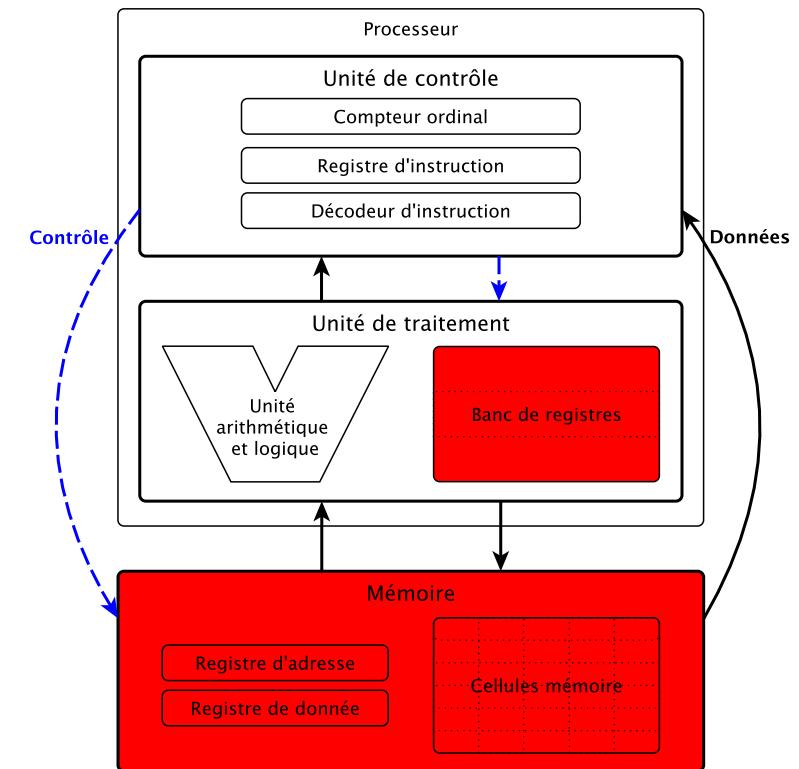
- ▶ Chaque instruction nécessite au moins un accès mémoire
 - ▶ Pour charger l'instruction
- ▶ À cela s'ajoutent les instructions de type `load` et `store`
 - ▶ 1/3 des instructions d'un programme en moyenne
 - ▶ Typiquement deux `load` pour chaque `store`
 - ▶ Exemple pour $A = B + C$:

```
LOAD R0, B      ; Charge B dans le registre R0
LOAD R1, C      ; Charge C dans le registre R1
ADD  R2, R0, R1 ; Place R0+R1 dans R2
STORE R2, A     ; Stocke le contenu de R2 dans A
```

- ▶ Élément clé de la performance d'une application
 - ▶ *Compute bound* si le temps est borné par la vitesse du processeur (cryptage, vidéo, simulation etc.)
 - ▶ *Memory bound* si il est borné par la vitesse du système mémoire (recherche, big data, gestion etc.)

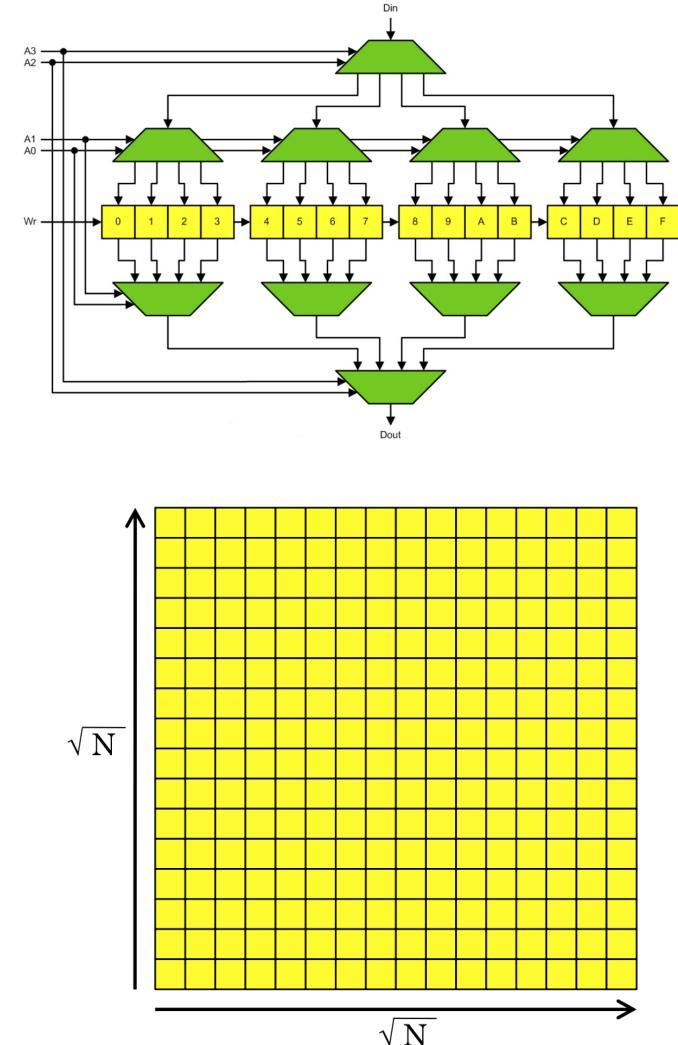
Vision idéale du système mémoire

- ▶ Pour l'instant, on a eu une vision simple du système mémoire
- ▶ Deux espaces :
 - ▶ Le banc de registres – petit nombre de cellules adressables
 - ▶ La mémoire centrale – très grand nombre de cellules adressables
- ▶ Accès à la mémoire centrale plus lent que l'accès aux registres (passage par un bus, par des registres intermédiaires...)



Limites physiques

- ▶ Temps d'accès, mémoire de taille N
- $$T_{accès} = T_{adresse} + T_{fil} + T_{cellule}$$
 - ▶ $T_{adresse} = C_1 \times \log(N)$: temps de traversée des portes logiques pour décoder l'adresse
 - ▶ $T_{fil} = C_2 \times \sqrt{N}$: temps de traversée des fils dans un espace mémoire 2D
 - ▶ $T_{cellule} = C_3$: temps de lecture ou d'écriture dans une cellule
- ▶ Dépend de N : plus une mémoire est grande, plus elle est lente !



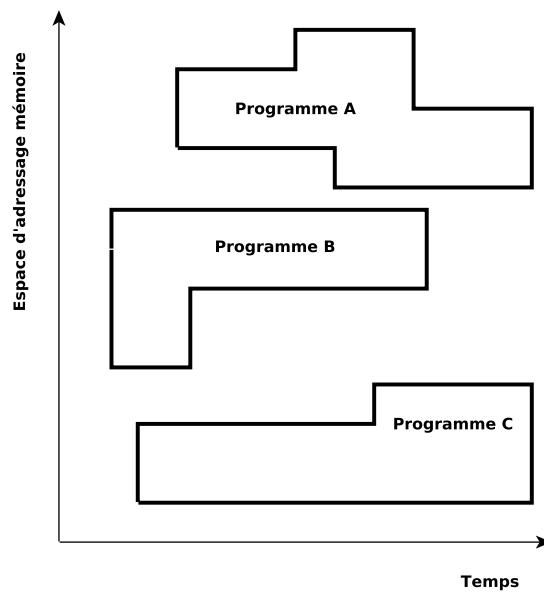
Hiérarchie mémoire

Motivation pour une hiérarchie mémoire

- Ce qu'on voudrait avoir idéalement :
 - ▶ Une seule mémoire de grande capacité, fonctionnant à la même vitesse que le processeur, pour un coût raisonnable
- La réalité :
 - ▶ Plus une mémoire est grande, plus elle est lente
 - ▶ Plus une mémoire est grande, plus son coût par bit est bas
 - ▶ Plus une mémoire est rapide, plus son coût par bit est élevé
- Une solution :
 - ▶ Créer un système mémoire composé de deux mémoires de types différents, l'une petite mais rapide, l'autre grande mais lente, et donnant l'illusion **la plupart du temps** d'une mémoire grande et rapide
 - ▶ Étendre cette idée à une hiérarchie de plusieurs niveaux
 - ▶ En profitant du fait que les accès ne sont pas aléatoires

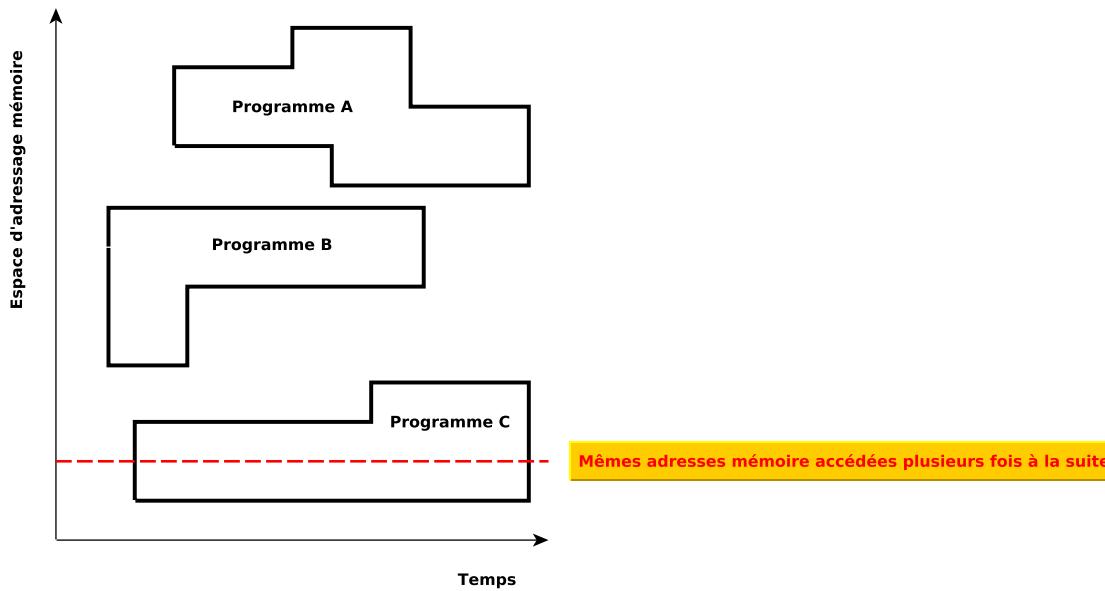
Principe de localité des données

- ▶ Empreinte mémoire de l'exécution de trois programmes :



Principe de localité des données

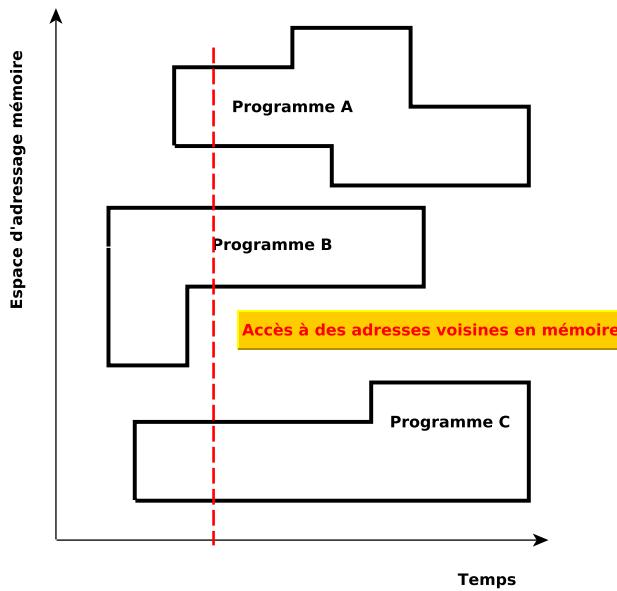
- ▶ Empreinte mémoire de l'exécution de trois programmes :



- ▶ **localité temporelle** : si une information est accédée en mémoire, il y a de fortes chances pour qu'elle le soit une nouvelle fois prochainement

Principe de localité des données

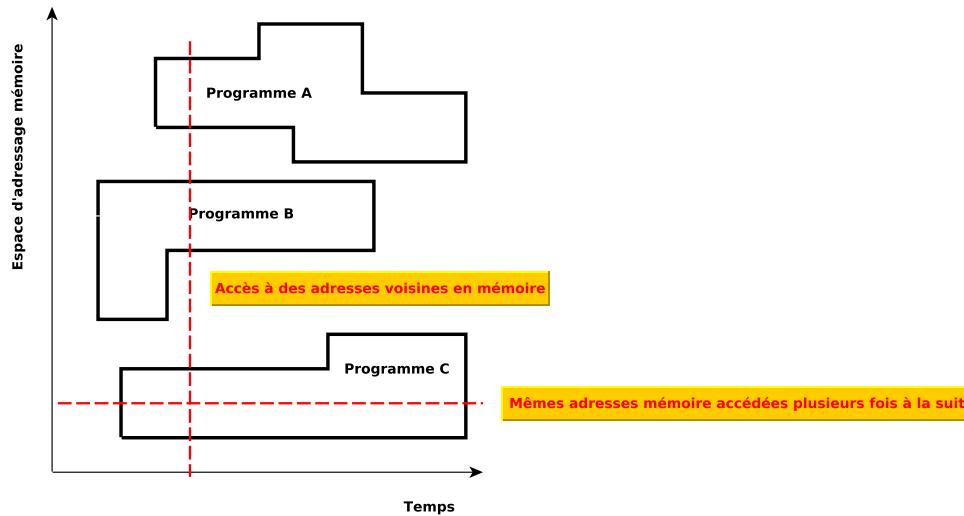
- ▶ Empreinte mémoire de l'exécution de trois programmes :



- ▶ **localité spatiale** : si une information est accédée en mémoire, il y a de fortes chances pour que celles situées dans son voisinage le soient aussi

Principe de localité des données

- ▶ Empreinte mémoire de l'exécution de trois programmes :



- ▶ Toute la hiérarchie mémoire est construite sur ce principe
- ▶ Conséquences du modèle de von Neumann :
 - ▶ Suite d'instructions en mémoire
 - ▶ Rappels de séquences d'instructions (boucles et fonctions)
 - ▶ Suites de données en mémoire (tableaux et structures)

Hiérarchie mémoire

Technologie	SRAM
Taille	x100 octets
Temps d'accès	<0.5 ns
Coût	Processeur

Technologie	SRAM
Taille	x10-x100 Kio
Temps d'accès	1-10 ns
Coût	~5000 € / Go

Technologie	DRAM
Taille	x Gio
Temps d'accès	30-100 ns
Coût	~10 € / Go

Technologie	Disque
Taille	x To
Temps d'accès	8-30 ms
Coût	~0.1 € / Go

Technologie	Bande
Taille	Infinie
Temps d'accès	Secondes
Coût	~0.1 € / Go

Niveaux hauts

Registres

Instructions et données

Cache

Blocs (ou lignes)

Mémoire centrale

Pages

Disque dur

Fichiers

Archives sur bandes magnétiques

Niveaux bas

Plus rapide

Gestion	Compilateur
Taille	1-8 octets

Gestion	Contrôleur
Taille	8-128 octets

Gestion	Système
Taille	2-4 Kio

Gestion	Utilisateur
Taille	x Mio

Plus grand

Illustration : avant ou après ?

Incrémantation des valeurs d'un tableau

[code]

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 100000000

int main() {
    double* tab = calloc(N, sizeof(double));
    clock_t time;

    time = clock();
    for (size_t i = 0; i < 100; i++)
        for (size_t j = 0; j < N; j++)
            tab[j] += 3.14;
    time = clock() - time;

    printf("Temps_de_calcul_:_%g_s\n", ((double)time) / CLOCKS_PER_SEC);
    return 0;
}
```

- ▶ Compilez et exécutez le code, notez le temps de calcul
- ▶ Intervertissez les deux boucles (d'abord sur j puis sur i)
Le temps de calcul est-il le même ?

Illustration : accès en ligne ou colonne ?

Initialisation d'un tableau

[code]

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 10000

int main() {
    double (*tab)[N] = malloc(sizeof(double[N][N]));
    clock_t time;

    time = clock();
    for (size_t i = 0; i < N; i++)
        for (size_t j = 0; j < N; j++)
            tab[j][i] = 0.;
    time = clock() - time;

    printf("Temps d'initialisation : %g s\n", ((double)time) / CLOCKS_PER_SEC);
    free(tab);
    return 0;
}
```

- ▶ Compilez et exédez le code, notez le temps d'initialisation
- ▶ Changez l'instruction d'initialisation par « `tab[i][j] = 0.;` »
Le temps d'initialisation est-il le même ?

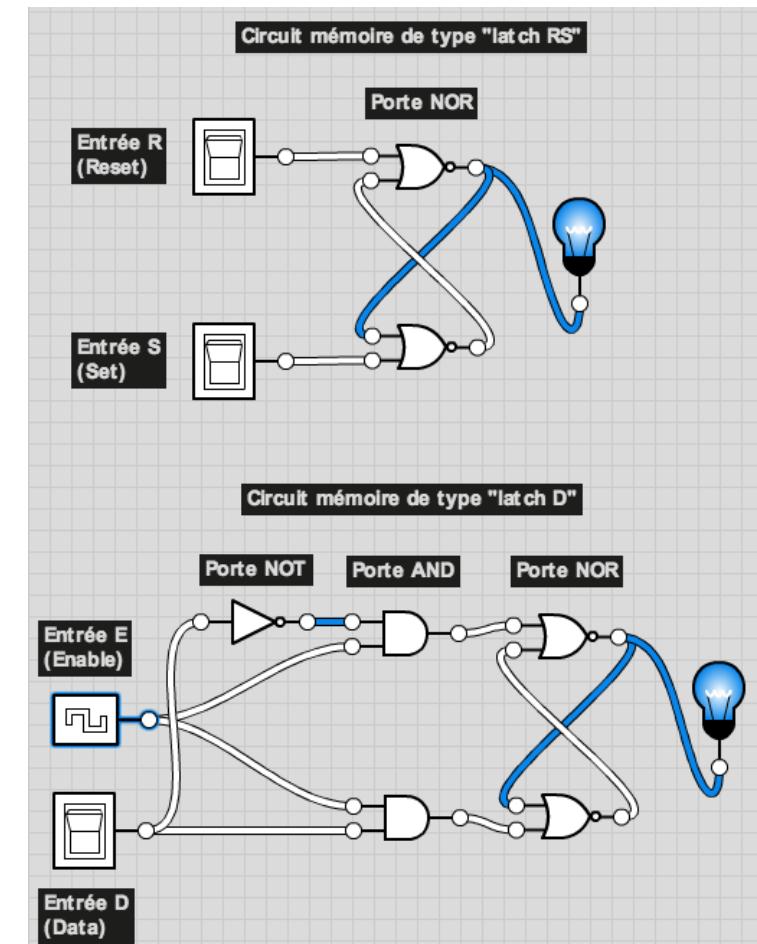
Technologies mémoire

Types de technologies mémoire

- ▶ Différentes technologies mémoire coexistent dans une architecture moderne
 - ▶ Chacune a ses forces et ses faiblesses
- ▶ La mémoire volatile perd l'information sans alimentation électrique
 - ▶ Static Random Access Memory (SRAM)
 - ▶ Dynamic Random Access Memory (DRAM)
- ▶ La mémoire persistante conserve l'information sans alimentation électrique
 - ▶ Read-Only Memory (ROM)
 - ▶ Electrically-Erasable Programmable Read-Only Memory (EEPROM ou mémoire « flash »))
 - ▶ Disques magnétiques
 - ▶ Bandes magnétiques
- ▶ Un système les utilise toutes pour différentes tâches

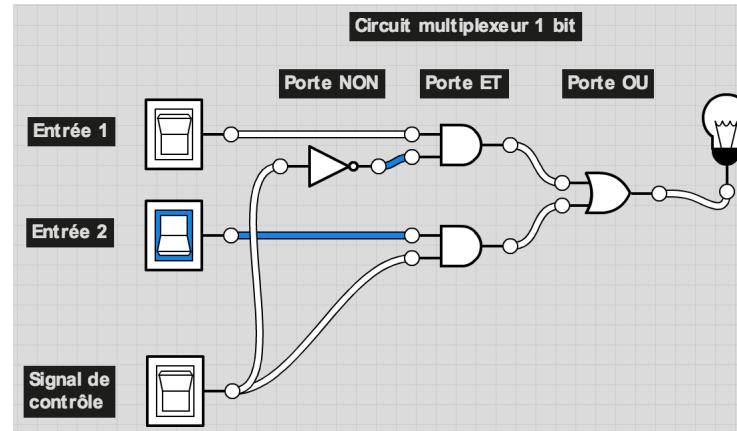
Circuit mémoire de base : le « latch »

- ▶ Circuit à deux états stables pouvant stocker de l'information
- ▶ Sa sortie et son état courant dépendent de son entrée et de son état précédent
- ▶ Latch RS (Reset-Set)
 - ▶ État maintenu si Reset et Set bas
 - ▶ État haut si Set haut et Reset bas
 - ▶ État bas si Set bas et Reset haut
 - ▶ Set et Reset hauts interdit
- ▶ Latch D (Data)
Étend le RS pour éviter l'état interdit : l'état de l'entrée Data est enregistré quand l'entrée Enable est à haut



Circuit « multiplexeur »

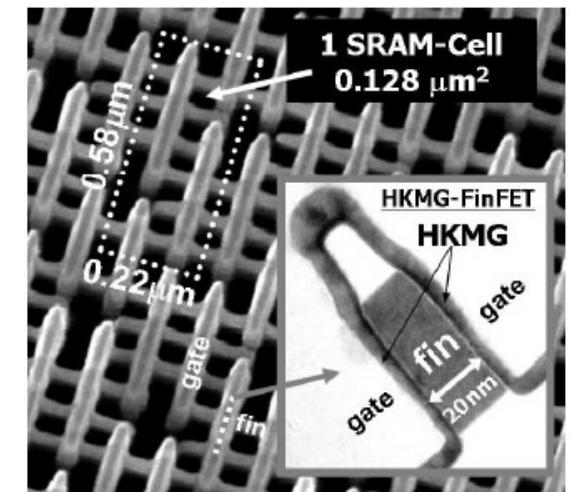
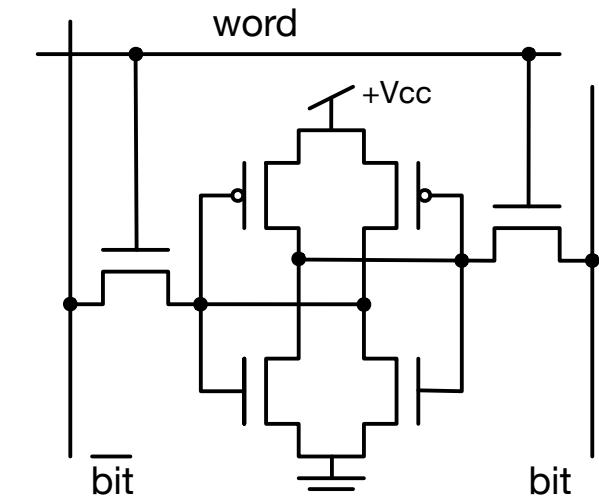
- ▶ Circuit permettant de sélectionner un signal parmi deux en utilisant un signal de contrôle
- ▶ Peut s'étendre : n signaux de contrôle permettent de sélectionner un signal parmi 2^n



- ▶ Utilisé pour « décoder » les adresses mémoire
 - ▶ Signaux de contrôle = adresse mémoire
 - ▶ Signal en sortie = « word line » (horizontal sur les dessins) permet de sélectionner la bonne cellule mémoire

Static RAM (SRAM)

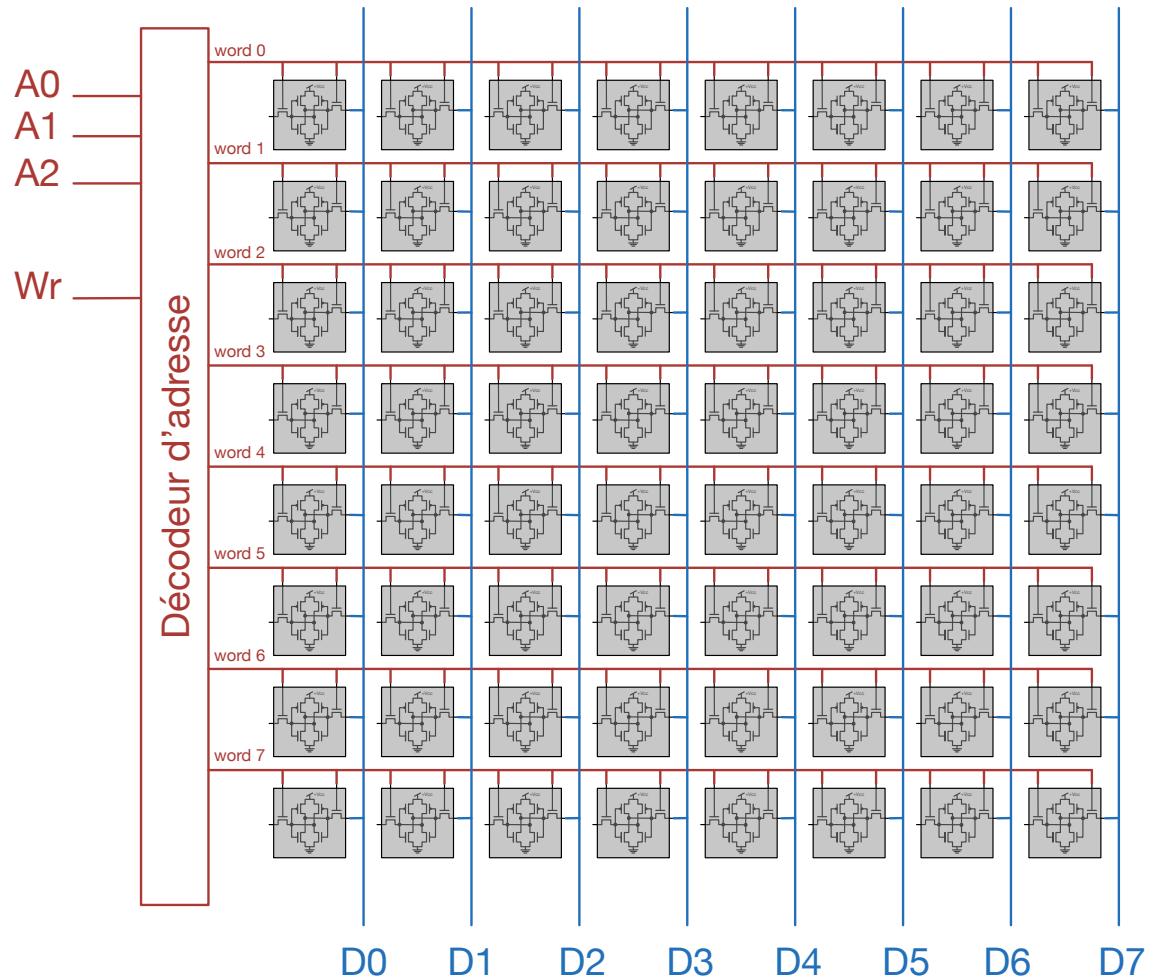
- ▶ RAM : Random Access Memory
 - ▶ Temps d'accès constant quelle que soit l'adresse ou l'information accédée avant
 - ▶ (contrairement aux bandes magnétiques qui sont un support séquentiel par ex.)
- ▶ Static : conserve son état tant qu'elle est alimentée électriquement
- ▶ Cellules mémoire proches du latch
 - ▶ Nécessitent 6 transistors par bit
 - ▶ Faible densité
 - ▶ Très rapide
 - ▶ Très chère
 - ▶ Forte consommation d'énergie
- ▶ Pour registres, buffers, caches etc.



A bird's-eye view of $0.128\mu\text{m}^2$ FinFET SRAM cells
(post silicide formation)

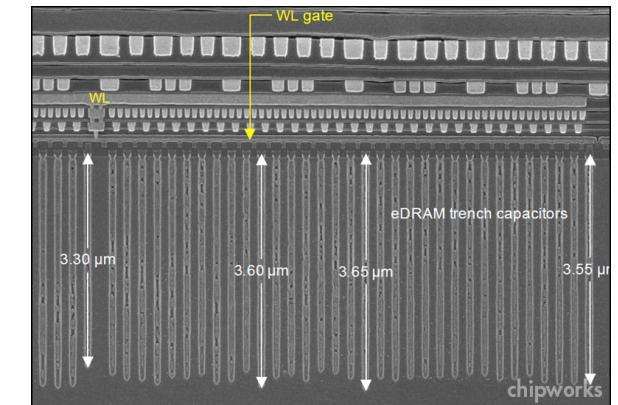
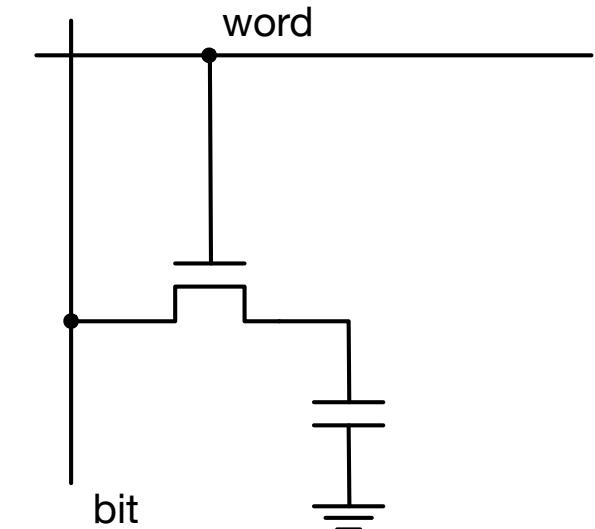
Organisation typique de la SRAM

- ▶ Tableau de cellules
- ▶ Ici 8 mots de 8 bits
- ▶ L'adresse entière est envoyée au décodeur
- ▶ Le décodeur active la ligne correspondante
- ▶ En fonction du signal Wr la ligne est lue ou écrite



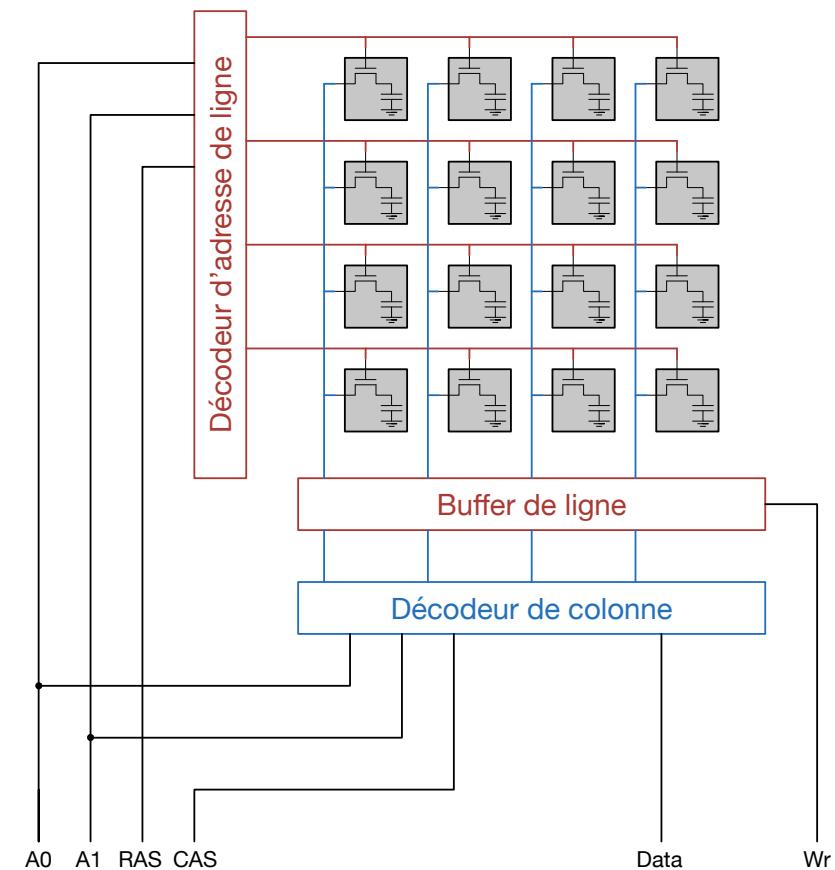
Dynamic RAM (DRAM)

- ▶ Solution plus dense que la SRAM :
 - ▶ Utiliser un simple condensateur !
 - ▶ Et un transistor pour en contrôler l'accès
 - ▶ Deux états : chargé (1) ou déchargé (0)
- ▶ Dynamic : état instable, doit être rafraîchi
 - ▶ Condensateur déchargé en 10 à 100 ms
 - ▶ Doit être rechargé périodiquement
 - ▶ Par lecture et réécriture de l'information
- ▶ Cellules mémoire simples
 - ▶ Haute densité
 - ▶ Coût réduit
 - ▶ TRÈS LENTES
- ▶ Mémoire centrale, vidéo etc.



Organisation typique de la DRAM

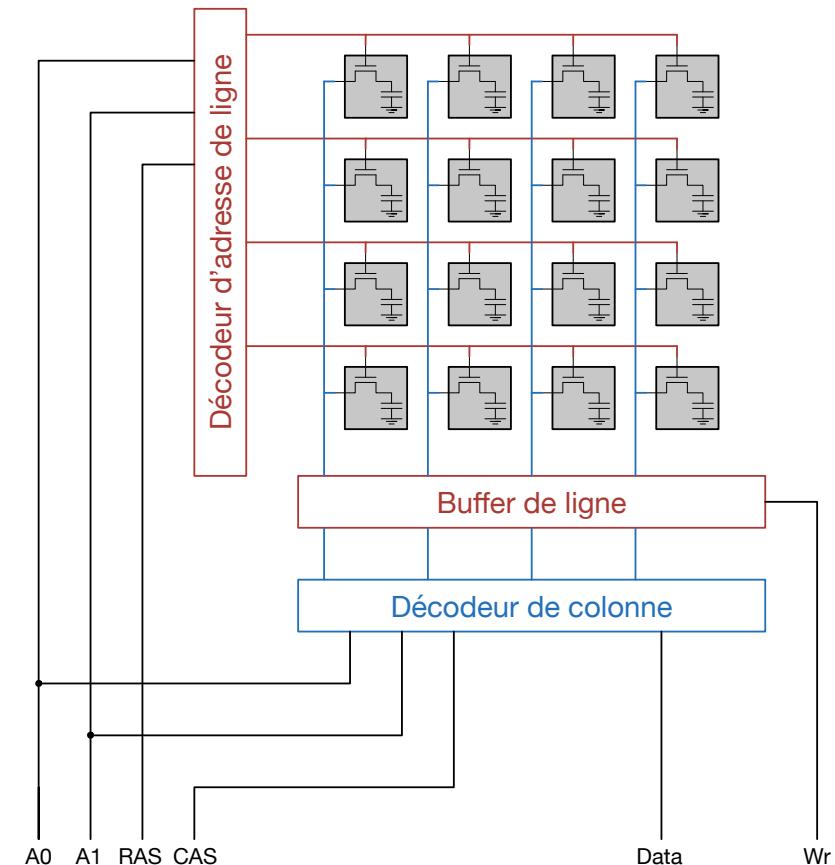
- ▶ Tableau 2D de cellules
- ▶ Ici 4×4 bits de DRAM
- ▶ L'adresse est décomposée, reçue et décodée en deux parties
 - ▶ Réduit le nombre de connecteurs
- ▶ Fonctionnement d'un accès :
 - ▶ Envoi adresse de ligne et RAS (Row Address Strobe)
 - ▶ Envoi adresse de colonne et CAS (Column Address Strobe)
 - ▶ Opération de lecture ou écriture



Exemple d'accès à une DRAM

Accès en deux temps :

- ➊ Envoi d'une adresse de ligne et du signal RAS
 - ▶ Toute une ligne est activée
 - ▶ Elle est mise en buffer de ligne
 - ▶ La ligne est rafraîchie
- ➋ Envoi d'une adresse de colonne et du signal CAS
 - ▶ Le bit recherché est lu dans le buffer de ligne



Exemple d'accès à une DRAM

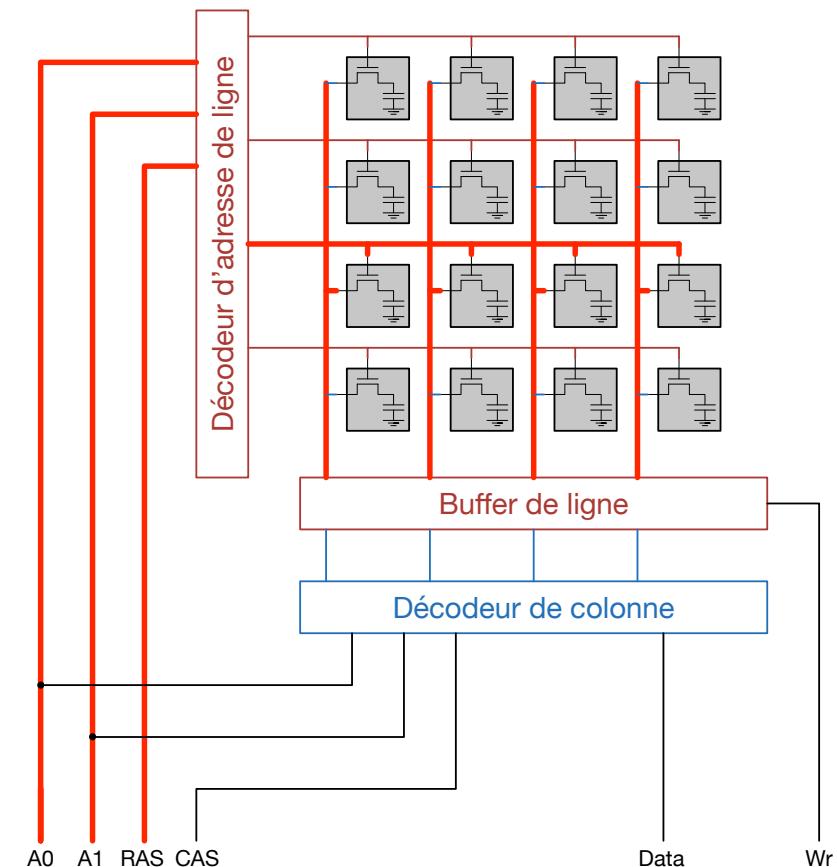
Accès en deux temps :

1 Envoi d'une adresse de ligne et du signal RAS

- ▶ Toute une ligne est activée
- ▶ Elle est mise en buffer de ligne
- ▶ La ligne est rafraîchie

2 Envoi d'une adresse de colonne et du signal CAS

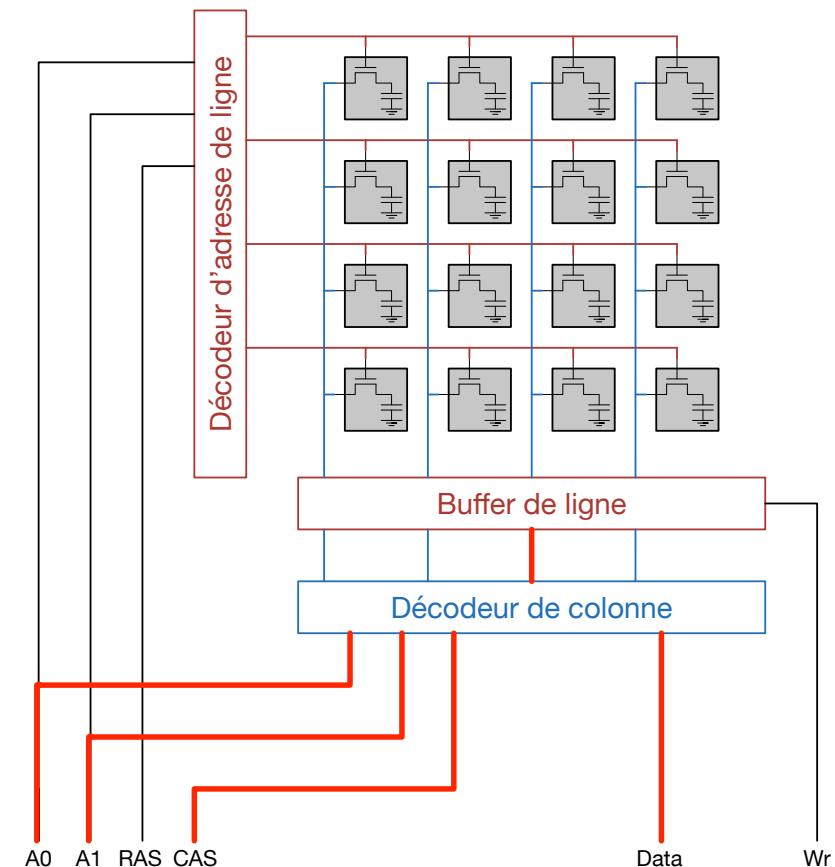
- ▶ Le bit recherché est lu dans le buffer de ligne



Exemple d'accès à une DRAM

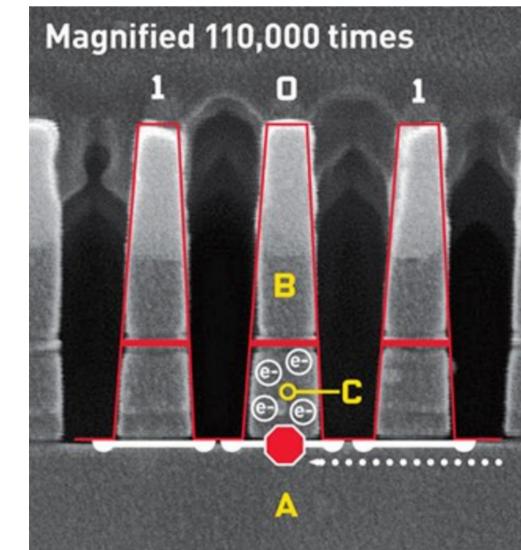
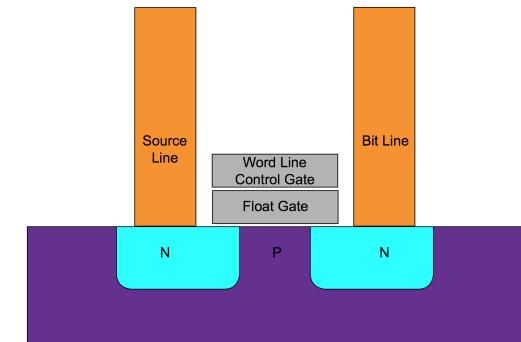
Accès en deux temps :

- ➊ Envoi d'une adresse de ligne et du signal RAS
 - ▶ Toute une ligne est activée
 - ▶ Elle est mise en buffer de ligne
 - ▶ La ligne est rafraîchie
- ➋ Envoi d'une adresse de colonne et du signal CAS
 - ▶ Le bit recherché est lu dans le buffer de ligne



Solid-state drive

- ▶ Mémoire de masse utilisant des circuits intégrés
 - ▶ Mémoire de type NAND flash
 - ▶ Nombre limité de cycles d'écriture
 - ▶ 1 à 3 bits par cellules (SLC, MLC, TLC)
Plus de bits → moins de durée de vie
 - ▶ Un contrôleur gère l'usure des circuits
- ▶ Information persistante même sans alimentation électrique
- ▶ Pas de partie mobile (« solid »)
 - ▶ Haute densité
 - ▶ Coût réduit
 - ▶ Durée de vie réduite
 - ▶ TRÈS TRÈS LENTS
- ▶ Utilisé comme mémoire secondaire



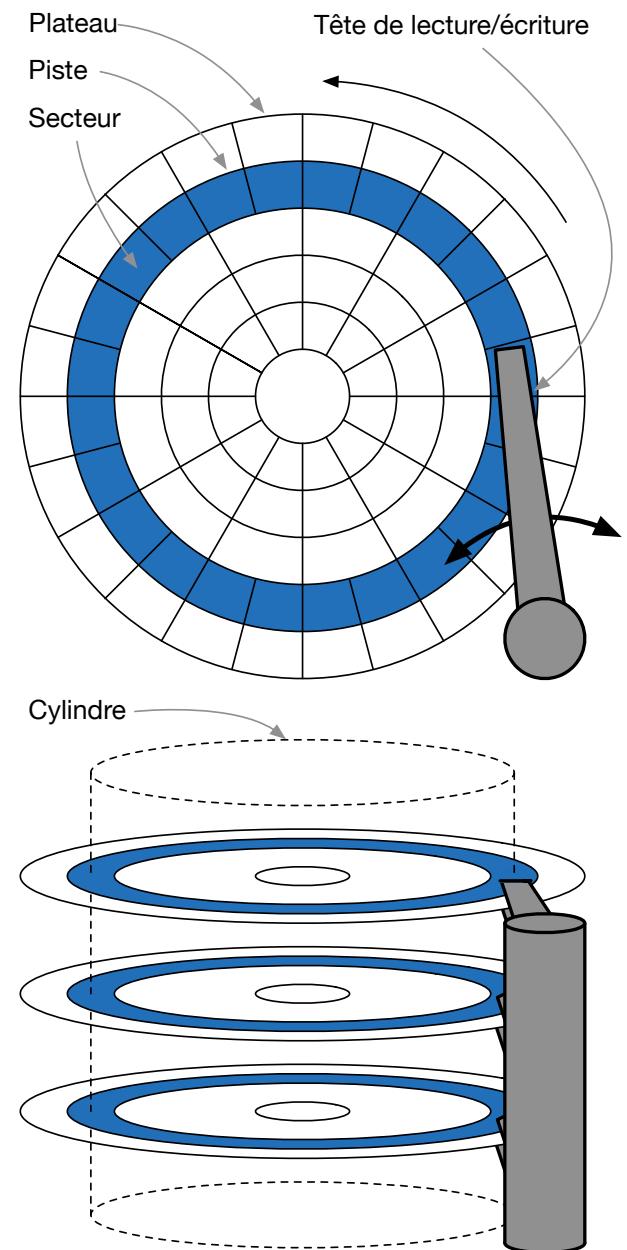
Disque dur magnétique

- ▶ Mémoire de masse utilisant des plateaux rigides recouverts d'une surface magnétique
 - ▶ Plateaux tournant à haute vitesse, jusqu'à 15000 RPM
 - ▶ Têtes capables de lire et d'écrire
- ▶ Information persistante même sans alimentation électrique
- ▶ Comprend des éléments mécaniques
 - ▶ Haute densité
 - ▶ Coût réduit
 - ▶ Durée de vie réduite
 - ▶ TRÈS TRÈS TRÈS LENTS
- ▶ Utilisé comme mémoire secondaire



Organisation typique d'un disque dur

- ▶ Organisation de la surface :
 - ▶ La surface d'un plateau est divisée en anneaux concentriques appelés *pistes*
 - ▶ Chaque piste est divisée en *secteurs*
 - ▶ Les pistes éloignées du centre ont plus de secteurs : pistes de même nombre de secteurs regroupées en *zones*
 - ▶ L'ensemble des $n^{ièmes}$ pistes de chaque plateau forme un *cylindre*
- ▶ Fonctionnement d'un accès :
 - ▶ Positionnement du bras pour que la tête soit sur la bonne piste
 - ▶ Rotation pour atteindre le bon secteur
 - ▶ **Le secteur est l'unité minimale d'information sur un disque dur**



Performance d'un disque dur

- ▶ Deux principaux facteurs de performance
- ▶ Temps de recherche
 - ▶ Temps pris par le bras pour se déplacer vers la bonne piste
 - ▶ Dépend du cylindre accédé avant (encore la localité !)
 - ▶ Au pire, 20 ms, en moyenne 6-9 ms
- ▶ Temps de rotation
 - ▶ Temps pris pour arriver au bon secteur
 - ▶ Au pire une rotation complète
 - ▶ À 7200 RPM, au pire un tour 8 ms, en moyenne un demi tour 4 ms
- ▶ Accéder à un secteur peut prendre entre 10 et 30 ms
- ▶ Taille d'un secteur typique : 4096 octets (physique) mais le contrôleur de disque permet 512 octets (logique)

Point sur les technologies mémoire

- ▶ Trois principales familles aujourd'hui : SRAM, DRAM et disques durs
- ▶ Comparaisons (2014) :

Technologie	SRAM	DRAM	Disque dur
Temps d'accès	1-10 ns	30-100 ns	8-30 ms
Capacité	x100 Kio	x1 Gio	x1 To
Coût	~5000 €/Go	~10 €/Go	~0.1 €/Go
Tendance temps d'accès	Stable depuis 2004	×2 en 7-8 ans	×2 en 14-15 ans
Tendance capacité	×2 en 3 ans	×2 en 2-3 ans	×2 en 3 ans

- ▶ À comparer avec un processeur : 3 GHz → 0.33 ns
- ▶ Si un cycle prenait une seconde :
 - ▶ Accès SRAM : 3-30 secondes
 - ▶ Accès DRAM : 1 min 30 à 5 minutes
 - ▶ Accès disque dur : 1 à 3 ans !!!