Feuille d'exercices – Architecture des ordinateurs

Feuille débranchée (sans ordinateur)

NSI – Première – Séquence 4 : Architecture des ordinateurs

Cette feuille reprend, sans ordinateur, les notions vues dans le cours de cette séquence. Certains exercices réutilisent ce que vous savez de l'écriture binaire et des puissances de 2.

Partie	**Objectif**	**Durée indicative**
A – Constituants et mémoires	Vocabulaire, rôles, hiérarchie	20 min
B – Le modèle de von Neumann	Schéma, cycle d'exécution	20 min
C – Quelques ordres de grandeur	Bus, fréquence, cœurs	25 min

1 Constituants et mémoires

Exercice 1: Vrai ou faux ?

Pour chacune des affirmations suivantes, dire si elle est **vraie** ou **fausse**, et justifier en une phrase.

1. La mémoire vive fait partie du logiciel de l'ordinateur.
2. Le contenu de la mémoire vive est conservé lorsque l'ordinateur est éteint.
3. Un SSD est plus rapide que la mémoire vive.
4. La mémoire cache est située dans le processeur.
5. Un programme Python séquentiel s'exécute 4 fois plus vite sur un processeur à 4 cœurs que sur un processeur à 1 cœur de même fréquence.
6. Dans une machine de von Neumann, le programme et les données sont rangés dans la même mémoire.

Correction de l'exercice 1

1. **Faux** : la mémoire vive est un composant physique, elle fait partie du **matériel**. Ce sont les programmes qu'elle contient qui relèvent du logiciel.
2. **Faux** : la mémoire vive est **volatile**, son contenu est perdu à la mise hors tension.
3. **Faux** : c'est l'inverse. Un accès à la RAM prend de l'ordre de 100 ns, un accès à un SSD de l'ordre de 100 µs, soit environ mille fois plus.
4. **Vrai** : c'est ce qui la rend si rapide, au prix d'une faible capacité.
5. **Faux** : un programme séquentiel n'utilise qu'un seul cœur ; les autres cœurs ne l'accélèrent pas.
6. **Vrai** : c'est le principe du **programme enregistré**, au cœur du modèle de von Neumann.

Exercice 2: Qui fait quoi ?

Associer chaque composant (colonne de gauche) à son rôle (colonne de droite).

1. Unité arithmétique et logique	a. Contient l'adresse de la prochaine instruction
2. Unité de commande	b. Relie les composants et transporte les informations
3. Compteur ordinal	c. Effectue les calculs et les comparaisons
4. Mémoire vive	d. Conserve durablement les fichiers
5. Bus	e. Lit, décode et fait exécuter les instructions
6. SSD	f. Contient les programmes en cours et leurs données

Correction de l'exercice 2

1 – c ; 2 – e ; 3 – a ; 4 – f ; 5 – b ; 6 – d.

Exercice 3: La hiérarchie des mémoires

On considère les mémoires suivantes : mémoire vive, registres, disque dur, mémoire cache, SSD.

1. Les ranger de la plus rapide à la plus lente.
2. Les ranger de la plus petite capacité à la plus grande. Que remarque-t-on ?
3. Lesquelles sont volatiles ?
4. Un logiciel installé est rangé sur le SSD ; pour être exécuté, il doit d'abord être copié en mémoire vive. Expliquer pourquoi l'ouverture d'un logiciel prend un certain temps, alors qu'une fois ouvert il réagit beaucoup plus vite.

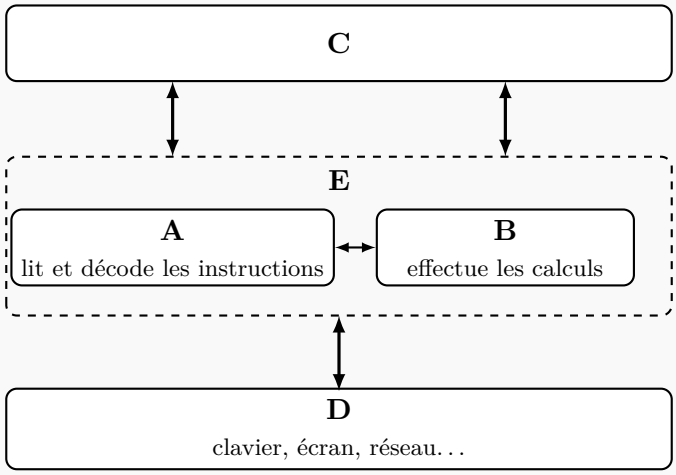
Correction de l'exercice 3

1. Registres, mémoire cache, mémoire vive, SSD, disque dur.
2. Exactement le même ordre : registres, cache, mémoire vive, SSD, disque dur. Plus une mémoire est rapide, plus sa capacité est faible (car elle est plus chère à fabriquer).
3. Les registres, la mémoire cache et la mémoire vive.
4. À l'ouverture, le programme et ses données doivent être lus sur le SSD, environ mille fois plus lent que la mémoire vive : c'est cette copie qui prend du temps. Une fois le programme en mémoire vive, le processeur y accède beaucoup plus rapidement, d'où un logiciel réactif.

2 Le modèle de von Neumann

Exercice 4: Compléter le schéma

Le schéma ci-dessous représente l'architecture de von Neumann. Donner le nom des éléments A, B, C, D et E, puis indiquer dans quel(s) élément(s) se trouvent les programmes en cours d'exécution.



Correction de l'exercice 4

A : unité de commande ; B : unité arithmétique et logique (UAL) ; C : mémoire ; D : entrées-sorties ; E : processeur (unité de commande + UAL).
Les programmes en cours d'exécution se trouvent dans la **mémoire** (C), **au même endroit**

que les données : c'est le principe du programme enregistré. (L'instruction en cours d'exécution est en plus recopiée dans le registre d'instruction, dans l'unité de commande.)

Exercice 5: Le compteur ordinal

Au début d'un cycle d'exécution, le compteur ordinal (CO) contient la valeur 12.

1. À quelle adresse mémoire l'unité de commande va-t-elle lire l'instruction ? Où cette instruction est-elle placée ?
2. L'instruction lue est une addition. Quelle est la valeur du CO à la fin du cycle ?
3. L'instruction lue est maintenant un saut vers l'adresse 5. Quelle est la valeur du CO à la fin du cycle ?
4. Un programme occupe les adresses 0 à 9 et les données les adresses 10 à 20. Le programmeur a oublié de terminer son programme par une instruction d'arrêt. Que va faire le processeur après avoir exécuté l'instruction d'adresse 9 ? Quel principe du modèle de von Neumann cette situation illustre-t-elle ?

Correction de l'exercice 5

1. À l'adresse **12**. L'instruction lue est placée dans le **registre d'instruction** (RI).
2. Une addition n'est pas un saut : le CO a simplement été augmenté de 1 lors de la lecture, il vaut **13**.
3. Le CO a d'abord été augmenté à 13 lors de la lecture, puis l'exécution du saut l'a remplacé par **5**. Le prochain cycle lira donc l'instruction d'adresse 5.
4. Le CO passe à 10 : le processeur va lire le contenu de la case 10 **comme s'il s'agissait d'une instruction**, et tenter de l'exécuter, ce qui produit un comportement imprévisible. Cela illustre le fait qu'instructions et données sont stockées dans la même mémoire et codées de la même façon (en binaire) : rien ne les distingue, sinon l'usage qu'en fait le processeur.

3 Quelques ordres de grandeur

Exercice 6: Taille du bus d'adresses

Chaque fil du bus d'adresses transporte un bit. On suppose que chaque adresse désigne une case d'**un octet**.

1. Un ancien processeur possède un bus d'adresses de 16 bits. Combien d'adresses différentes peut-il écrire ? Quelle quantité de mémoire peut-il utiliser, en octets puis en Kio (1 Kio = 1024 octets) ?
2. Même question pour un bus d'adresses de 32 bits, en Gio (1 Gio = 2^{30} octets).
3. Combien de bits faut-il au minimum dans le bus d'adresses pour pouvoir utiliser 16 Gio de mémoire vive ?

Correction de l'exercice 6

1. Avec 16 bits, on peut écrire $2^{16} = 65\,536$ adresses, donc utiliser 65 536 octets, soit $65\,536/1024 =$ **64 Kio**.
2. Avec 32 bits : 2^{32} octets $= 2^2 \times 2^{30}$ octets = **4 Gio**.
3. 16 Gio $= 2^4 \times 2^{30} = 2^{34}$ octets : il faut au minimum **34 bits**. Un bus d'adresses de 32 bits ne suffit donc plus : c'est l'une des raisons du passage aux processeurs 64 bits.

Exercice 7: Le processeur qui attend

Un processeur fonctionne à la fréquence de 3,2 GHz. On admet qu'il effectue une opération simple par cycle d'horloge.

1. Calculer la durée d'un cycle d'horloge, en nanosecondes (1 ns $= 10^{-9}$ s).
2. Un accès à la mémoire vive dure environ 80 ns. Combien de cycles le processeur pourrait-il effectuer pendant ce temps ?
3. Un accès à la mémoire cache dure environ 1 ns. Même question.
4. En déduire l'intérêt de la mémoire cache.

Correction de l'exercice 7

1. Durée d'un cycle : $\dfrac{1}{3{,}2 \times 10^9} = 3{,}125 \times 10^{-10}$ s, soit **0,3125 ns** (environ 0,3 ns).
2. $80/0{,}3125 =$ **256** cycles : pendant un seul accès à la mémoire vive, le processeur aurait pu effectuer 256 opérations.
3. $1/0{,}3125 = 3{,}2$, soit environ **3** cycles seulement.
4. Si la donnée se trouve dans le cache, le processeur attend environ 3 cycles au lieu de 256 : le cache évite au processeur de passer l'essentiel de son temps à attendre la mémoire vive (c'est le « goulot d'étranglement de von Neumann »).

Exercice 8: Combien de cœurs ?

Sur un processeur à un seul cœur, un logiciel de montage vidéo met 12 minutes à produire une vidéo.

1. On suppose que ce travail peut être réparti parfaitement entre plusieurs cœurs. Combien de temps faudrait-il avec 4 cœurs ? Avec 12 cœurs ?
2. En réalité, seuls 75 % du travail peuvent être répartis entre les cœurs ; les 25 % restants doivent être exécutés par un seul cœur. Calculer la durée avec 4 cœurs.
3. Toujours dans la situation de la question 2, montrer que, même avec un très grand nombre de cœurs, la durée ne pourra jamais descendre sous une certaine valeur, que l'on précisera.

Correction de l'exercice 8

1. Avec 4 cœurs : $12/4 =$ **3** minutes. Avec 12 cœurs : $12/12 =$ **1** minute.
2. La partie non répartissable dure $0{,}25 \times 12 = 3$ minutes. La partie répartissable dure $0{,}75 \times 12 = 9$ minutes sur un cœur, donc $9/4 = 2{,}25$ minutes sur 4 cœurs. Durée totale : $3 + 2{,}25 =$ **5,25 minutes**, soit 5 min 15 s (et non 3 minutes).
3. Quel que soit le nombre de cœurs, les 3 minutes de la partie non répartissable restent incompressibles ; seule la partie répartissable diminue (et tend vers 0). La durée ne descendra donc jamais sous **3 minutes**. Multiplier les cœurs n'a d'intérêt que si le programme est conçu pour en profiter.