Architecture des ordinateurs

NSI – Première

Constituants d'une machine, mémoires, modèle de von Neumann

Table des matières

1 Les constituants d'un ordinateur **2**
1.1 Matériel et logiciel . . . 2
1.2 La carte mère et ses composants . . . 2
1.3 Les périphériques . . . 3

2 Les mémoires **5**
2.1 Mémoires volatiles et non volatiles . . . 5
2.2 La hiérarchie des mémoires . . . 5

3 Le modèle d'architecture de von Neumann **6**
3.1 Un peu d'histoire . . . 6
3.2 Les quatre constituants . . . 6
3.3 Le processeur : registres et horloge . . . 7
3.4 Le cycle d'exécution d'une instruction . . . 8
3.5 Le langage machine . . . 9

4 Au-delà du modèle de von Neumann **9**
4.1 Le goulot d'étranglement de von Neumann . . . 9
4.2 Architectures monoprocesseur et multiprocesseur . . . 10
4.3 De quoi est faite l'UAL ? . . . 10

Introduction

Vu de l'extérieur, un ordinateur est une machine de **traitement de l'information** : il est capable d'acquérir de l'information, de la stocker, de la transformer en effectuant des calculs, puis de la restituer sous une autre forme. « L'information » est prise ici au sens large : du texte, des nombres, des sons, des images… tous représentés, comme nous l'avons vu dans la séquence sur la représentation des entiers, par des suites de 0 et de 1.

Vu de l'intérieur, un ordinateur est un ensemble de composants électroniques qui interagissent pour obtenir le résultat voulu par l'utilisateur. Depuis le début de l'année, vous écrivez des programmes Python sans vous demander ce qui se passe réellement dans la machine lorsque vous appuyez sur « Exécuter ». C'est précisément l'objet de cette séquence :

— quels sont les **constituants** d'un ordinateur, et quel est le rôle de chacun ?
— où sont rangés les programmes et les données, et pourquoi existe-t-il plusieurs sortes de **mémoires** ?
— comment le processeur **exécute**-t-il un programme, instruction après instruction ?
— pourquoi presque tous les ordinateurs, du supercalculateur au smartphone, reposent-ils sur un même modèle, imaginé en 1945 ?

Remarque : Activité d'introduction

L'activité d'introduction de cette séquence part d'objets que vous connaissez : la fiche technique d'un ordinateur portable, et une cuisine de restaurant. Les notions que vous y aurez

rencontrées sans les nommer (processeur, mémoire vive, stockage, hiérarchie des mémoires, rôle de chaque constituant) sont formalisées dans ce cours.

1 Les constituants d'un ordinateur

1.1 Matériel et logiciel

Définition 1.1: Matériel, logiciel

— Le **matériel** (en anglais *hardware*) désigne l'ensemble des éléments physiques d'un ordinateur, que l'on peut voir et toucher : processeur, mémoire, carte mère, écran, clavier, etc.
— Le **logiciel** (en anglais *software*) désigne l'ensemble des programmes, c'est-à-dire des suites d'instructions, qui indiquent au matériel quels traitements effectuer.

Un ordinateur de bureau, un ordinateur portable, une tablette ou un smartphone reposent tous sur les mêmes principes : seule change la façon dont les composants sont regroupés (dans un grand boîtier, dans un boîtier compact, ou même sur une seule puce dans le cas des smartphones).

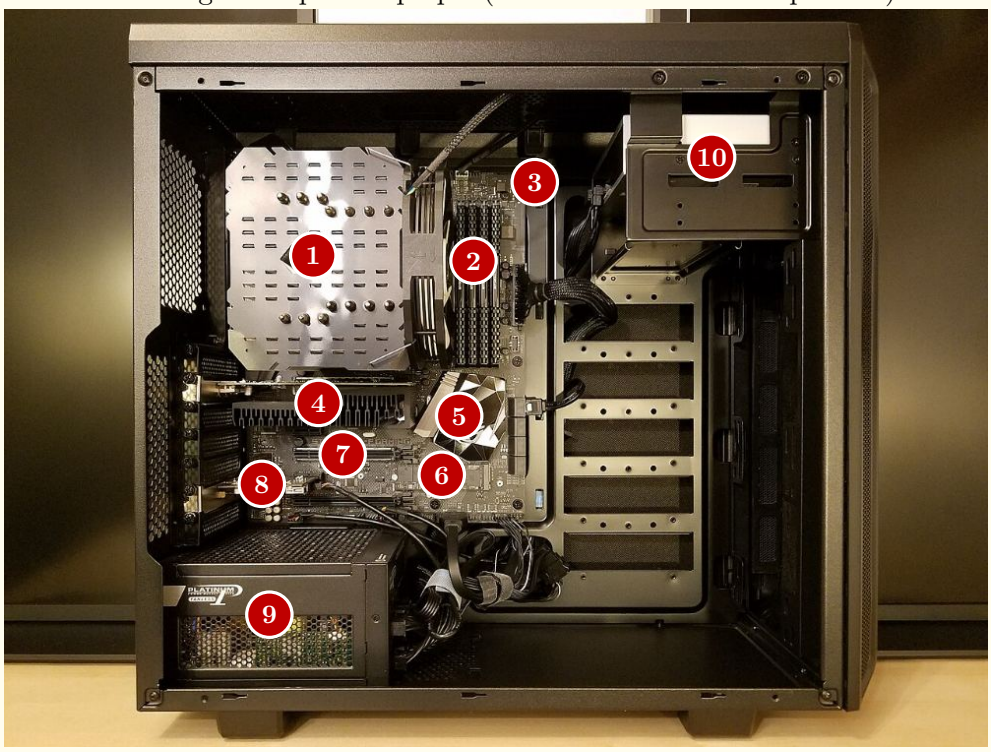
1.2 La carte mère et ses composants

Définition 1.2: Carte mère

La **carte mère** est le circuit imprimé principal de l'ordinateur : c'est une plaque sur laquelle sont fixés ou branchés les composants, et qui contient les pistes conductrices permettant à ces composants de communiquer entre eux.

Exemple 1.3: L'intérieur d'une unité centrale

Voici l'intérieur d'un ordinateur de bureau de 2018, panneau latéral retiré. Il s'agit d'une configuration haut de gamme pour l'époque (environ 1 600 € de composants) :



1. Processeur, caché sous son ventirad (radiateur + ventilateur) : 8 cœurs à 3 GHz ;

2. Mémoire vive : 4 barrettes de 8 Go, soit 32 Go ;
3. Carte mère, vissée au fond du boîtier ;
4. Carte graphique, branchée sur un connecteur PCI Express ;
5. Chipset, sous son dissipateur ;
6. SSD au format M.2 (512 Go), directement fixé sur la carte mère ;
7. Connecteur PCI Express libre, pour ajouter une carte d'extension ;
8. Carte Wi-Fi et Bluetooth ;
9. Alimentation électrique (400 W) ;
10. Lecteur-graveur de disques Blu-ray.

Photo : Bretwa, *ATX computer case – left – 2018-05-18*, Wikimedia Commons, licence CC BY-SA 4.0 (numéros ajoutés ; l'image annotée est diffusée sous la même licence).

Sur la carte mère (ou branchés sur elle), on trouve en particulier :

— le **processeur** (ou **CPU**, de l'anglais *Central Processing Unit*) : c'est lui qui exécute les instructions des programmes ;
— la **mémoire vive** (ou **RAM**, de l'anglais *Random Access Memory*) : elle contient les programmes en cours d'exécution et les données sur lesquelles ils travaillent ;
— le **chipset** : un ensemble de circuits qui coordonne les échanges de données entre le processeur, la mémoire et les périphériques ;
— les **connecteurs d'extension** (PCI Express) qui permettent d'ajouter des cartes, par exemple une carte graphique ;
— les **ports** permettant de brancher les périphériques : USB, HDMI, Ethernet (réseau filaire), prise audio, etc. ;
— le **micrologiciel** de démarrage (BIOS ou UEFI), stocké dans une mémoire non volatile de la carte mère, qui est le tout premier programme exécuté à la mise sous tension.

Définition 1.4: Bus

Un **bus** est un ensemble de liaisons (fils, pistes) permettant de transmettre des informations entre plusieurs composants. On distingue en particulier :
— le **bus de données**, qui transporte les valeurs échangées ;
— le **bus d'adresses**, qui indique à quel emplacement de la mémoire on veut lire ou écrire ;
— le **bus de contrôle**, qui transporte les signaux de commande (lecture ou écriture, horloge, etc.).

Exemple 1.5: Taille du bus d'adresses

Un bus d'adresses de n fils permet d'écrire 2^n adresses différentes (chaque fil transporte un bit). Un processeur dont le bus d'adresses fait 32 bits peut donc désigner $2^{32} = 4\,294\,967\,296$ cases mémoire. Lorsque chaque case contient un octet, cela représente 4 Gio : c'est pourquoi les anciens systèmes « 32 bits » ne pouvaient pas utiliser plus de 4 Gio de mémoire vive.

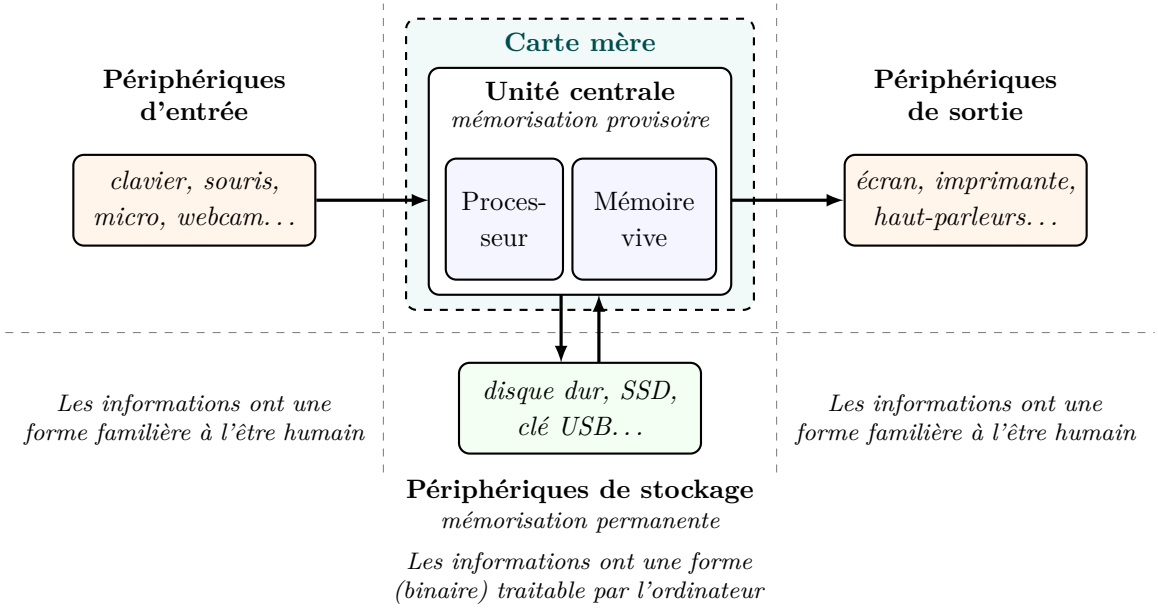
1.3 Les périphériques

Définition 1.6: Périphérique

Un **périphérique** est un dispositif connecté à l'ordinateur qui lui permet d'échanger des informations avec l'extérieur. On distingue :
— les **périphériques d'entrée**, qui transforment une action ou un objet du monde réel en une suite de bits : clavier, souris, micro, webcam, scanner, capteurs...
— les **périphériques de sortie**, qui transforment une suite de bits en une informa-

tion perceptible par un humain ou en une action : écran, imprimante, haut-parleurs, moteurs...
— les **périphériques de stockage**, qui conservent les données durablement et dans lesquels l'ordinateur peut lire et écrire : disque dur, SSD, clé USB, carte SD...

Le schéma suivant résume le fonctionnement d'un ordinateur vu de l'extérieur : les informations entrent par les périphériques d'entrée, sont traitées par l'unité centrale, puis ressortent par les périphériques de sortie ; les périphériques de stockage les conservent durablement.



Remarque : Des frontières parfois floues

Certains périphériques jouent plusieurs rôles à la fois : un écran tactile est à la fois un périphérique d'entrée (on le touche) et de sortie (il affiche) ; une carte réseau envoie et reçoit des données. Le rôle des capteurs et des actionneurs sera approfondi dans la séquence sur les interfaces homme-machine.

Exercice 1.7: Classer des périphériques

Classer chacun des éléments suivants en périphérique d'entrée, de sortie, de stockage (certains peuvent appartenir à plusieurs catégories) : manette de jeu, vidéoprojecteur, SSD externe, casque avec micro, imprimante 3D, lecteur d'empreinte digitale.

Correction

— **Entrée** : manette de jeu, lecteur d'empreinte digitale, et le micro du casque.
— **Sortie** : vidéoprojecteur, imprimante 3D, et les écouteurs du casque.
— **Stockage** : SSD externe.

Le casque avec micro est donc à la fois un périphérique d'entrée et de sortie. (Une manette avec vibrations est aussi, en partie, un périphérique de sortie.)

2 Les mémoires

2.1 Mémoires volatiles et non volatiles

Définition 2.1: Mémoire volatile, non volatile

— Une mémoire est **volatile** si son contenu est perdu lorsque l'appareil est mis hors tension.
— Elle est **non volatile** (ou **permanente**) si son contenu est conservé sans alimentation électrique.

Définition 2.2: Les principaux types de mémoire

— La **mémoire vive** (RAM) est une mémoire **volatile**, rapide, dans laquelle on peut lire et écrire à n'importe quel emplacement en un temps comparable (d'où le nom d'accès « aléatoire », *random access*). Elle contient les programmes en cours d'exécution et leurs données.
— La **mémoire morte** (ROM, de l'anglais *Read Only Memory*) est une mémoire **non volatile**, que l'on ne peut pas (ou difficilement) modifier. Elle contient par exemple le micrologiciel de démarrage. Aujourd'hui, ce rôle est presque toujours tenu par de la mémoire *flash*, non volatile mais réinscriptible, ce qui permet de mettre à jour le BIOS ou l'UEFI.
— La **mémoire de masse** (disque dur, SSD, clé USB…) est une mémoire **non volatile** de grande capacité, beaucoup plus lente que la mémoire vive, qui conserve durablement les fichiers et les programmes installés.

Remarque : Pourquoi faut-il « enregistrer » un fichier ?

Lorsque vous modifiez un document, les modifications sont faites dans la mémoire vive. Si l'ordinateur s'éteint brutalement avant que vous n'ayez enregistré, ces modifications sont perdues : la RAM est volatile. « Enregistrer », c'est recopier les données de la mémoire vive vers la mémoire de masse.

2.2 La hiérarchie des mémoires

Une mémoire idéale serait à la fois très rapide, de très grande capacité, bon marché et non volatile. Une telle mémoire n'existe pas : plus une mémoire est rapide, plus elle est chère et donc de petite capacité. Les ordinateurs combinent donc plusieurs niveaux de mémoire.

Propriété 2.3: Hiérarchie des mémoires

Les mémoires d'un ordinateur s'organisent en niveaux, du plus rapide (et plus petit) au plus lent (et plus grand). Les ordres de grandeur suivants sont ceux d'un ordinateur personnel actuel :

Niveau	Où ?	Capacité typique	Temps d'accès typique
Registres	dans le processeur	quelques centaines d'octets	moins de 1 ns
Mémoire cache	dans le processeur	de quelques Ko à quelques dizaines de Mo	de 1 à 10 ns environ
Mémoire vive (RAM)	sur la carte mère	de 8 à 32 Go	environ 100 ns
SSD	mémoire de masse	de 256 Go à quelques To	de l'ordre de 100 µs
Disque dur	mémoire de masse	de 1 à 20 To environ	de l'ordre de 10 ms

Les registres, le cache et la RAM sont volatiles ; le SSD et le disque dur sont non volatiles.

Définition 2.4: Mémoire cache

La **mémoire cache** est une petite mémoire très rapide, intégrée au processeur, qui conserve une copie des données et des instructions utilisées le plus récemment ou le plus fréquemment. Lorsque le processeur a besoin d'une donnée, il la cherche d'abord dans le cache : s'il l'y trouve, il évite un accès, beaucoup plus lent, à la mémoire vive.

Exemple 2.5: Un ordre de grandeur parlant

Si l'on imagine qu'un accès à un registre dure **1 seconde**, alors, à la même échelle, un accès à la mémoire vive dure de l'ordre de quelques minutes, un accès à un SSD de l'ordre de quelques jours, et un accès à un disque dur de l'ordre d'une année entière !

3 Le modèle d'architecture de von Neumann

3.1 Un peu d'histoire

Les premiers calculateurs électroniques, comme l'ENIAC (1945, États-Unis), étaient programmés en modifiant physiquement des câblages et des interrupteurs : changer de programme pouvait prendre plusieurs jours. En 1945, le mathématicien **John von Neumann**, qui participe à la conception du calculateur suivant, l'EDVAC, rédige un rapport décrivant une machine dans laquelle **le programme est enregistré dans la mémoire**, au même titre que les données. Changer de programme revient alors simplement à changer le contenu de la mémoire.

Ce modèle, appelé **architecture de von Neumann** (ou **architecture séquentielle**), est encore aujourd'hui à la base de la quasi-totalité des ordinateurs.

3.2 Les quatre constituants

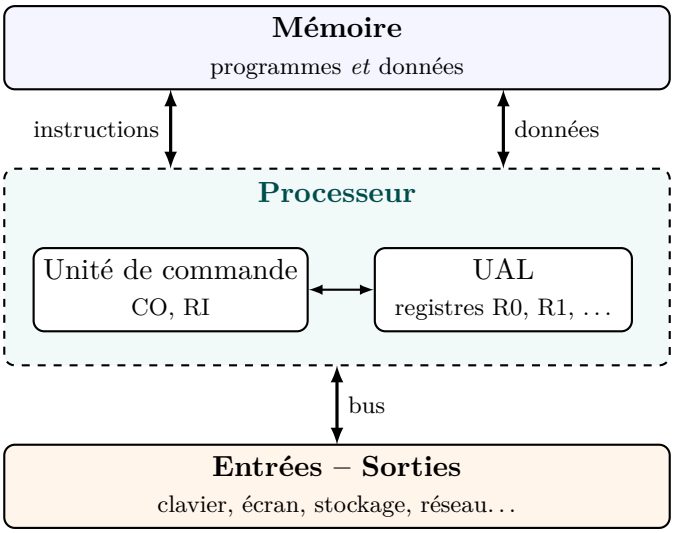
Définition 3.1: Architecture de von Neumann

Une machine de von Neumann est constituée de quatre parties :

- l'**unité arithmétique et logique** (**UAL**, en anglais *ALU*), qui effectue les calculs : opérations arithmétiques (addition, soustraction…), opérations logiques (et, ou, non) et comparaisons ;
- l'**unité de commande** (ou unité de contrôle), qui lit les instructions en mémoire, les décode, et commande leur exécution par les autres unités ;

— la **mémoire**, qui contient à la fois les **programmes** et les **données** ;
— les dispositifs d'**entrées-sorties**, qui permettent de communiquer avec l'extérieur.

Ces unités communiquent entre elles par des **bus**. L'UAL et l'unité de commande forment ensemble le **processeur**.



Propriété 3.2: Le principe du programme enregistré

Dans une machine de von Neumann, les instructions et les données sont stockées **dans la même mémoire**, et codées de la même façon : en binaire. Rien, dans une case mémoire, n'indique si elle contient une instruction ou une donnée : c'est la manière dont le processeur l'utilise qui en décide.

3.3 Le processeur : registres et horloge

Définition 3.3: Registre

Un **registre** est un petit emplacement de mémoire situé **à l'intérieur** du processeur, extrêmement rapide, qui contient une seule valeur (par exemple sur 64 bits). Parmi les registres, on distingue :
— le **compteur ordinal** (**CO**, en anglais *program counter*) : il contient l'**adresse** de la prochaine instruction à exécuter ;
— le **registre d'instruction** (**RI**) : il contient l'instruction en cours d'exécution ;
— les **registres généraux** (notés ici R0, R1, R2, ...) : ils contiennent les valeurs sur lesquelles l'UAL effectue ses calculs.

Définition 3.4: Horloge, fréquence

Le fonctionnement du processeur est rythmé par une **horloge**, qui émet un signal régulier. La **fréquence** de l'horloge est le nombre de cycles par seconde ; elle s'exprime en hertz (Hz). Les processeurs actuels fonctionnent à quelques gigahertz (1 GHz $= 10^9$ Hz), soit plusieurs milliards de cycles par seconde.

3.4 Le cycle d'exécution d'une instruction

Méthode 3.5: Le cycle « chercher – décoder – exécuter »

Le processeur exécute un programme en répétant sans cesse le cycle suivant :

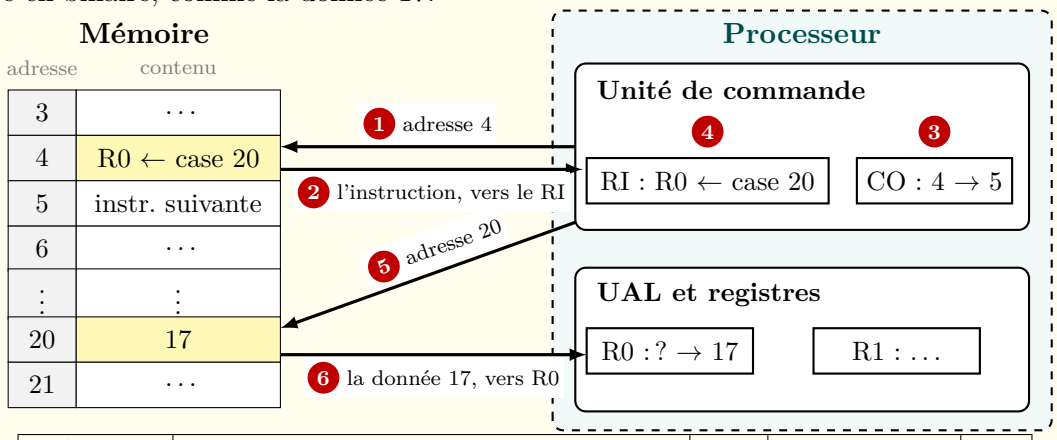
1. **Chercher** (*fetch*) : l'unité de commande lit en mémoire l'instruction située à l'adresse contenue dans le CO, et la place dans le RI ; le CO est ensuite augmenté de 1, pour désigner l'instruction suivante.
2. **Décoder** (*decode*) : l'unité de commande analyse l'instruction contenue dans le RI pour déterminer l'opération à effectuer et ses opérandes.
3. **Exécuter** (*execute*) : l'opération est réalisée (calcul par l'UAL, lecture ou écriture en mémoire, ou modification du CO dans le cas d'un saut).

Puis le cycle recommence, jusqu'à rencontrer une instruction d'arrêt.

Exemple 3.6: Un cycle détaillé, étape par étape

On suit le processeur pendant **un seul cycle**. La mémoire contient un programme rangé à partir de l'adresse 4, et une donnée, le nombre 17, rangée à l'adresse 20. Au début du cycle, le compteur ordinal contient **CO = 4**.

L'instruction rangée à l'adresse 4 demande de *copier dans le registre R0 le contenu de la case mémoire 20*. On l'écrit ici, de façon lisible, « R0 ← case 20 » ; en mémoire, elle est bien sûr codée en binaire, comme la donnée 17.



Étape	Ce qui se passe	CO	RI	R0
	Début du cycle	4	...	?
Chercher	1 L'unité de commande envoie l'adresse contenue dans le CO, 4, sur le bus d'adresses.	4	...	?
	2 La mémoire renvoie le contenu de la case 4 sur le bus de données ; il est rangé dans le RI.	4	R0 ← case 20	?
	3 Le CO augmente de 1.	5	R0 ← case 20	?
Décoder	4 L'unité de commande analyse le RI : c'est une instruction de transfert, il faut lire la case 20 et écrire le résultat dans R0.	5	R0 ← case 20	?
Exécuter	5 L'unité de commande envoie l'adresse 20 sur le bus d'adresses.	5	R0 ← case 20	?
	6 La mémoire renvoie 17 sur le bus de données ; la valeur est rangée dans R0.	5	R0 ← case 20	17

À la fin du cycle, le CO vaut 5 : le cycle suivant ira donc chercher l'instruction rangée à l'adresse 5. On remarque que la mémoire a été sollicitée **deux fois** au cours de ce seul cycle :

une fois pour lire l'**instruction**, une fois pour lire la **donnée**. Les deux passent par les mêmes bus, puisque programme et données sont rangés dans la même mémoire.
Si l'instruction avait été un calcul, par exemple « R2 ← R0 + R1 », l'étape « Exécuter » n'aurait pas utilisé la mémoire : l'unité de commande aurait demandé à l'UAL d'additionner les contenus de R0 et R1, et le résultat aurait été rangé dans R2.

Remarque : Pourquoi « séquentielle » ?

Les instructions sont exécutées **les unes après les autres**, dans l'ordre où elles sont rangées en mémoire, grâce au compteur ordinal qui avance de 1 à chaque cycle. Seules certaines instructions particulières, les **sauts**, permettent de rompre cet ordre.

3.5 Le langage machine

Définition 3.7: Langage machine, jeu d'instructions

Le **langage machine** est le seul langage que le processeur sait exécuter directement : chaque instruction y est une suite de bits, rangée en mémoire comme n'importe quelle donnée. L'ensemble des instructions qu'un processeur sait exécuter s'appelle son **jeu d'instructions** ; il dépend de la famille de processeurs (les processeurs x86 des ordinateurs de bureau et les processeurs ARM des smartphones, par exemple, ont des jeux d'instructions différents).

Les instructions d'un jeu d'instructions sont très élémentaires. On y trouve essentiellement :

— des instructions de **transfert**, qui copient une valeur de la mémoire vers un registre, ou d'un registre vers la mémoire ;
— des instructions de **calcul**, exécutées par l'UAL sur des registres (addition, soustraction, comparaison, opérations logiques...) ;
— des instructions de **saut**, qui modifient le compteur ordinal pour rompre l'ordre séquentiel : ce sont elles qui permettent de réaliser des instructions conditionnelles et des boucles.

Remarque : Et Python dans tout ça ?

Python est un langage de **haut niveau** : une seule ligne Python correspond à de nombreuses instructions machine. Lorsque vous exécutez un programme Python, un autre programme (l'*interpréteur* Python, lui-même écrit en langage machine) le lit et fait exécuter par le processeur les instructions machine correspondantes.

Remarque : Vers la séquence sur le langage assembleur

Le langage machine, son écriture lisible appelée **langage assembleur**, et l'exécution pas à pas de petits programmes par le processeur feront l'objet d'une séquence à part entière.

4 Au-delà du modèle de von Neumann

4.1 Le goulot d'étranglement de von Neumann

Dans le modèle de von Neumann, chaque instruction et chaque donnée doit transiter entre la mémoire et le processeur par le bus. Or le processeur calcule beaucoup plus vite que la mémoire vive ne répond : sans précaution, il passerait l'essentiel de son temps à attendre. On parle de **goulot d'étranglement de von Neumann**. C'est pour atténuer ce problème que les processeurs disposent de **mémoires caches** (voir la partie 2).

4.2 Architectures monoprocesseur et multiprocesseur

Pendant des décennies, les performances des processeurs ont progressé principalement grâce à l'augmentation de leur fréquence. Depuis le milieu des années 2000, cette fréquence plafonne à quelques gigahertz : au-delà, le processeur consomme trop d'énergie et chauffe trop. Pour continuer à gagner en performances, les constructeurs ont alors multiplié les unités de calcul.

Définition 4.1: Architecture monoprocesseur, multiprocesseur

— Dans une architecture **monoprocesseur**, une seule unité de calcul exécute les instructions, une par une.
— Dans une architecture **multiprocesseur**, plusieurs unités de calcul exécutent des instructions **en même temps** (en parallèle). Aujourd'hui, ces unités sont le plus souvent regroupées sur une même puce : on parle de processeur **multicœur**, chaque **cœur** étant une unité de calcul complète.

Remarque : Plus de cœurs ne veut pas toujours dire plus rapide

Un programme écrit de manière purement séquentielle, comme tous ceux que nous avons écrits jusqu'ici en Python, n'utilise qu'un seul cœur : il ne va pas plus vite sur un processeur à 8 cœurs que sur un processeur à 1 cœur de même fréquence. Les cœurs supplémentaires servent à exécuter plusieurs programmes en même temps, ou des programmes spécialement conçus pour répartir leur travail entre plusieurs cœurs (calcul d'images, compression vidéo, etc.).

4.3 De quoi est faite l'UAL ?

À l'échelle la plus fine, un processeur est constitué de plusieurs milliards de **transistors**, de minuscules interrupteurs commandés électriquement. Assemblés, ils forment des **portes logiques**, qui réalisent les opérations booléennes *et*, *ou*, *non* sur des bits. En combinant ces portes, on obtient des **circuits combinatoires** capables, par exemple, d'additionner deux nombres écrits en binaire : c'est ainsi qu'est construite l'UAL.

Remarque : Vers la séquence sur les booléens

Les opérations booléennes, leurs tables de vérité et la construction d'un additionneur binaire à partir de portes logiques seront étudiées dans la séquence consacrée aux booléens.

Bilan

Dans cette séquence, nous avons appris à :
— distinguer le **matériel** du **logiciel**, et identifier les principaux composants d'un ordinateur (processeur, mémoire, carte mère, bus, périphériques) ;
— distinguer mémoires **volatiles** et **non volatiles**, et situer registres, cache, RAM et mémoire de masse dans la **hiérarchie des mémoires** ;
— décrire le modèle de **von Neumann** : UAL, unité de commande, mémoire contenant programmes *et* données, entrées-sorties ;
— expliquer le cycle **chercher – décoder – exécuter** et le rôle du **compteur ordinal** et du **registre d'instruction** ;
— définir le **langage machine** et le **jeu d'instructions** d'un processeur ;
— distinguer architectures **monoprocesseur** et **multiprocesseur**.