INF101
Introduction aux outils informatiques

Devoir 1

Présenté par :

JOSEPH Samuel Jonathan
DELIA Yves Anzt-ky

Hiver 2026

Exercice 1 : Composants d'une tour d'ordinateur de bureau

Les principaux composants que l'on trouve dans une tour d'ordinateur de bureau sont :

— **La carte mère (Motherboard)** : Le circuit imprimé principal qui connecte tous les composants.

— **Le processeur (CPU)** : Le cerveau de l'ordinateur qui exécute les instructions.

— **Le ventirad (Système de refroidissement)** : Dissipateur thermique et ventilateur pour le processeur.

— **La mémoire vive (RAM)** : Stocke temporairement les données en cours d'utilisation.

— **Le disque dur (HDD) ou disque SSD** : Pour le stockage permanent des données et du système d'exploitation.

— **Le bloc d'alimentation (PSU)** : Fournit l'électricité aux composants.

— **La carte graphique (GPU)** : Gère l'affichage à l'écran (peut être intégrée au CPU ou dédiée).

— **Le lecteur optique** (Optionnel, de plus en plus rare).

— **La carte réseau / Carte Wi-Fi** (Souvent intégrée à la carte mère).

— **La carte son** (Souvent intégrée à la carte mère).

FIGURE 1 – Vue interne des composants d'un ordinateur
Source : Image générée par IA (Amazon Titan Image Generator v2)

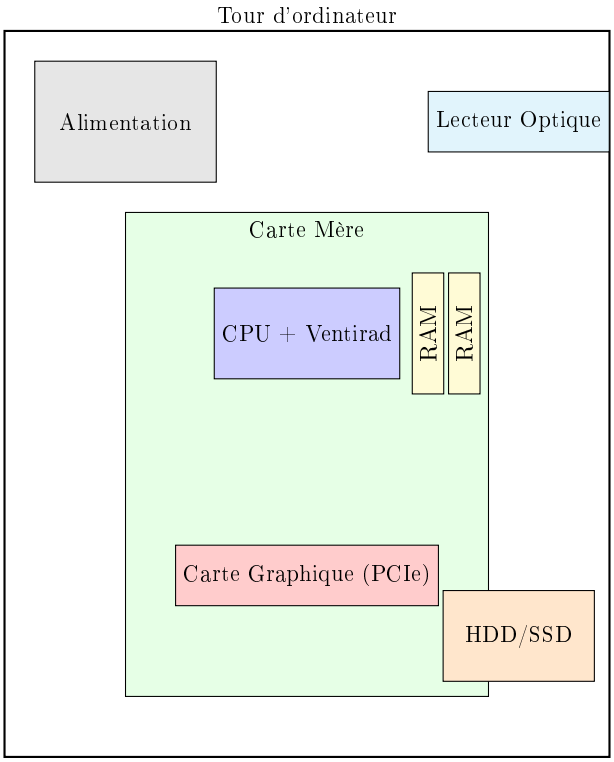

FIGURE 2 – Schéma simplifié de l'agencement interne d'une tour PC

Exercice 2 : Analyse du scénario et périphériques

Basé sur le scénario de l'étudiant de l'ISTEAH, voici les périphériques identifiés et leurs types :

FIGURE 3 – Illustration du poste de travail et des périphériques
Source : Image générée par IA (Amazon Titan Image Generator v2)

Périphérique	**Type**	**Justification dans le texte**
Clavier	Entrée	"Il saisit son texte à l'aide du clavier"
Souris	Entrée	"utilise la souris pour corriger certaines parties"
Écran	Sortie	"Il regarde alors une capsule vidéo" (Impliqué par le visionnage)
Écouteurs	Sortie	"il met ses écouteurs"
Clé USB	Entrée / Sortie	"copie la partie déjà réalisée du devoir sur une clé USB"
Imprimante	Sortie	"il imprime la partie déjà faite de son devoir"

Exercice 3 : Rôles des composants numériques (Architecture de Von Neumann)

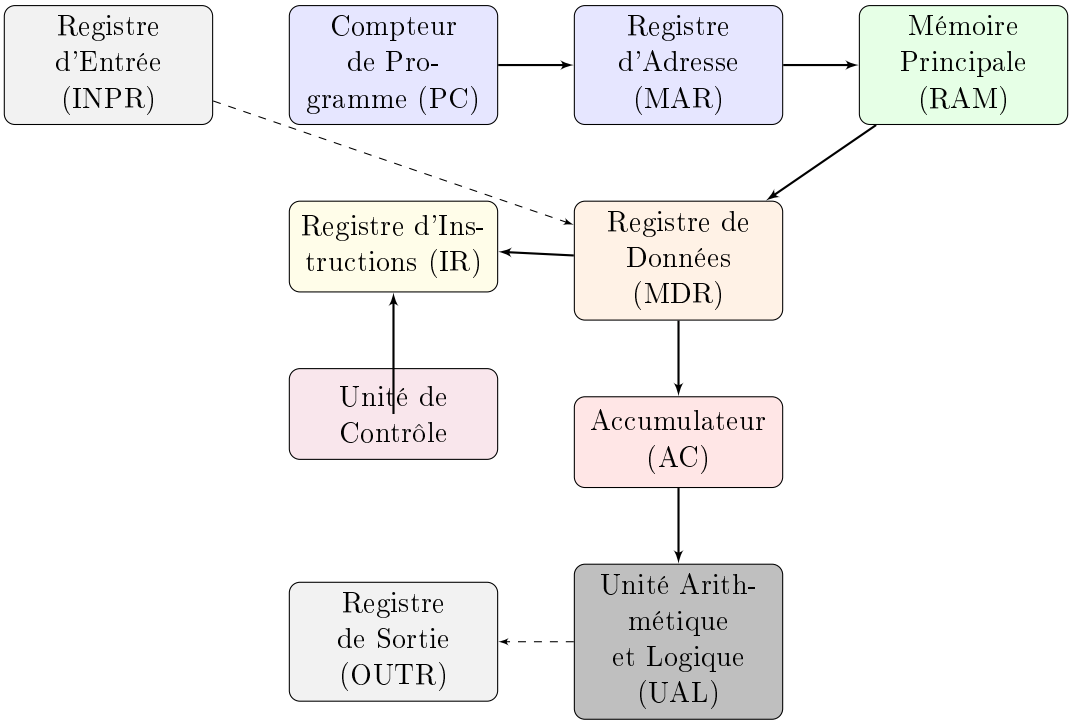

FIGURE 4 – Schéma conceptuel de l'Architecture de Von Neumann

Voici les définitions et rôles des composants numériques dans l'architecture de base d'un ordinateur :

Compteur de programme (PC)
C'est un registre qui contient l'adresse mémoire de la *prochaine* instruction à exécuter par le processeur. Il est incrémenté automatiquement après chaque lecture d'instruction.

Registre de données (MDR)
Ce registre stocke temporairement les données qui viennent d'être lues de la mémoire ou celles qui sont prêtes à être écrites dans la mémoire. Il agit comme un tampon entre le processeur et la mémoire centrale.

Accumulateur (AC)
C'est un registre spécial de l'unité arithmétique et logique ((UAL) utilisé pour stocker les résultats intermédiaires des opérations arithmétiques et logiques.

Registre d'instructions (IR)
Ce registre contient l'instruction qui est *actuellement* en cours d'exécution. Le processeur décode le contenu de ce registre pour savoir quelle opération effectuer.

Registre tampon
Un registre utilisé pour stocker temporairement des données lors de leur transfert entre deux unités fonctionnelles fonctionnant à des vitesses différentes (par exemple, entre le processeur et un périphérique).

Registre de sortie (OUTR)
Ce registre conserve les données traitées par le processeur avant qu'elles ne soient envoyées vers un périphérique de sortie (comme un écran ou une imprimante).

Registre d'entrée (INPR)
Ce registre reçoit et stocke temporairement les données provenant d'un périphérique d'entrée (comme un clavier) avant qu'elles ne soient traitées par le processeur.

Registre d'adresse (MAR)
Ce registre contient l'adresse de l'emplacement mémoire auquel le processeur veut accéder, que ce soit pour lire une instruction/donnée ou pour écrire une donnée.

Exercice 4 : Convertissez les nombres décimaux suivants en bases indiquées

a. 265 en Binaire

On divise par 2 successivement :

— $265 \div 2 = 132$, reste 1
— $132 \div 2 = 66$, reste 0
— $66 \div 2 = 33$, reste 0
— $33 \div 2 = 16$, reste 1
— $16 \div 2 = 8$, reste 0
— $8 \div 2 = 4$, reste 0
— $4 \div 2 = 2$, reste 0
— $2 \div 2 = 1$, reste 0
— $1 \div 2 = 0$, reste 1

Le resultat est donc $(265)_{10} = \mathbf{100001001_2}$, en faisant la notation du bas vers le haut

b. 839 à octal

On divise par 8 successivement :

— $839 \div 8 = 104$, reste 7
— $104 \div 8 = 13$, reste 0
— $13 \div 8 = 1$, reste 5
— $1 \div 8 = 0$, reste 1

Le resultat est donc $(839)_{10} = \mathbf{1507_8}$ en faisant la notation du bas vers le haut

c. 572 en hexadécimal

— $572 \div 16 = 35$, reste 12 (ce qui correspond à C)
— $35 \div 16 = 2$, reste 3
— $2 \div 16 = 0$, reste 2

$(572)_{10} = \mathbf{23C_{16}}$ **en faisant la notation du bas vers le haut**

Exercice 5 : Cycle d'instruction

Un cycle d'instruction est la séquence d'opérations effectuées par le processeur pour traiter une seule instruction. Il se décompose en trois phases : **Recherche** (Fetch), **Décodage** (Decode) et **Exécution** (Execute).

Prenons l'exemple suivant : l'instruction `ADD 200` est stockée à l'adresse mémoire `100`, et la case mémoire `200` contient la valeur `45`.

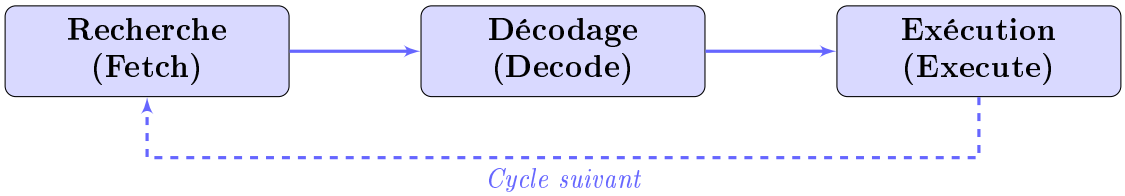

FIGURE 5 – Les trois phases du cycle d'instruction

Phase 1 – Recherche (Fetch)

Le processeur va chercher l'instruction en mémoire :

1. Le contenu du compteur de programme (PC) est copié dans le **registre d'adresse** (MAR) : MAR ← `100`
2. La mémoire renvoie le contenu de cette adresse dans le **registre de données** (MDR) : MDR ← `ADD 200`
3. Le contenu du MDR est transféré dans le **registre d'instruction** (IR) : IR ← `ADD 200`
4. Le compteur de programme est incrémenté : PC ← `101`

Phase 2 – Décodage (Decode)

1. L'unité de contrôle analyse le contenu de l'IR et identifie l'opération (`ADD`) et l'adresse de l'opérande (`200`).

Phase 3 – Exécution (Execute)

1. L'adresse de l'opérande est placée dans le **registre d'adresse** : MAR ← `200`
2. La valeur à cette adresse est lue dans le **registre de données** : MDR ← `45`
3. L'opération est effectuée dans le **registre du processeur** (accumulateur) : AC ← AC + 45

État des registres à chaque phase

Phase	**Reg. adr.** (MAR)	**Reg. mém.** (Mémoire)	**Reg. instr.** (IR)	**Reg. donn.** (MDR)	**Reg. proc.** (AC)
Recherche	100	Mém[100] lu	ADD 200	ADD 200	–
Décodage	–	–	ADD 200	–	–
Exécution	200	Mém[200] lu	ADD 200	45	AC + 45

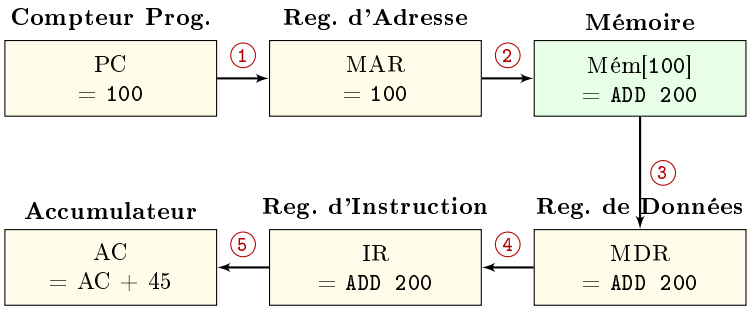

FIGURE 6 – Flux de données entre les registres lors du cycle d'instruction

Exercice 6 : Explicitez le mécanisme de conversion du décimal 41.6875 en binaire 101001.1011

Pour convertir un nombre décimal à virgule en binaire, on sépare le problème en deux parties : la partie entière (avant la virgule) et la partie fractionnaire (après la virgule).

1. La partie entière : 41

On utilise la méthode des divisions successives par 2. On note les restes de bas en haut.

- $41 \div 2 = 20$, reste 1
- $20 \div 2 = 10$, reste 0
- $10 \div 2 = 5$, reste 0
- $5 \div 2 = 2$, reste 1
- $2 \div 2 = 1$, reste 0
- $1 \div 2 = 0$, reste 1

On obtient : $(41)_{10} = \mathbf{101001_2}$

2. La partie fractionnaire : 0.6875

Ici, on utilise des multiplications successives par 2.

- $0.6875 \times 2 = \mathbf{1}.375 \rightarrow$ **on note 1, il reste** 0.375
- $0.375 \times 2 = \mathbf{0}.75 \rightarrow$ **on note 0, il reste** 0.75
- $0.75 \times 2 = \mathbf{1}.5 \rightarrow$ **on note 1, il reste** 0.5
- $0.5 \times 2 = \mathbf{1}.0 \rightarrow$ **on note 1, il reste** 0

Donc en lisant de haut en bas, on obtient : **.1011** On réunit les deux parties séparées par la virgule binaire :

$$101001 + .1011 = \mathbf{101001.1011_2}$$

Exercice 7 : Complément à $(r-1)$ en octal et hexadécimal

Rappel

Pour un nombre N en base r ayant n chiffres, le complément à $(r-1)$ est défini par :

$$(r^n - 1) - N$$

La méthode pratique consiste à soustraire **chaque chiffre** individuellement de $(r-1)$.

a) Complément à 7 d'un nombre octal

En base octale, $r = 8$ donc $r - 1 = 7$. Le complément à 7 s'obtient en **soustrayant chaque chiffre octal de 7**.

Exemple : Complément à 7 de $\mathbf{5274_8}$ $(n = 4$ chiffres)

$$C_7(5274_8) = (8^4 - 1) - 5274_8 = 7777_8 - 5274_8$$

Position	Chiffre 4	Chiffre 3	Chiffre 2	Chiffre 1
$r - 1 = 7$	7	7	7	7
N	5	2	7	4
Résultat	**2**	**5**	**0**	**3**

$$\boxed{C_7(5274_8) = 2503_8}$$

Vérification : $5274_8 + 2503_8 = 7777_8 = 8^4 - 1$ ✓

b) Complément à 15 (F) d'un nombre hexadécimal

En base hexadécimale, $r = 16$ donc $r - 1 = 15 = \mathrm{F}_{16}$. Le complément à F s'obtient en **soustrayant chaque chiffre hexadécimal de F (15)**.

Exemple : Complément à F de $\mathbf{3A7_{16}}$ $(n = 3$ chiffres)

$$C_F(3\mathrm{A}7_{16}) = (16^3 - 1) - 3\mathrm{A}7_{16} = \mathrm{FFF}_{16} - 3\mathrm{A}7_{16}$$

Position	Chiffre 3	Chiffre 2	Chiffre 1
$r - 1 = \mathrm{F}$ (15)	F (15)	F (15)	F (15)
N	3	A (10)	7
Soustraction	$15 - 3 = 12$	$15 - 10 = 5$	$15 - 7 = 8$
Résultat	**C**	**5**	**8**

$$\boxed{C_F(3\mathrm{A}7_{16}) = \mathrm{C}58_{16}}$$

Vérification : $3\mathrm{A}7_{16} + \mathrm{C}58_{16} = \mathrm{FFF}_{16} = 16^3 - 1$ ✓

Tableau récapitulatif par analogie

Base	r	$r-1$	**Méthode**	**Exemple**
Binaire	2	1	Inverser chaque bit ($0 \leftrightarrow 1$)	$C_1(1011001) = 0100110$
Octale	8	7	Soustraire chaque chiffre de 7	$C_7(5274_8) = 2503_8$
Décimale	10	9	Soustraire chaque chiffre de 9	$C_9(546700) = 453299$
Hexadécimale	16	F	Soustraire chaque chiffre de F	$C_F(3A7_{16}) = C58_{16}$

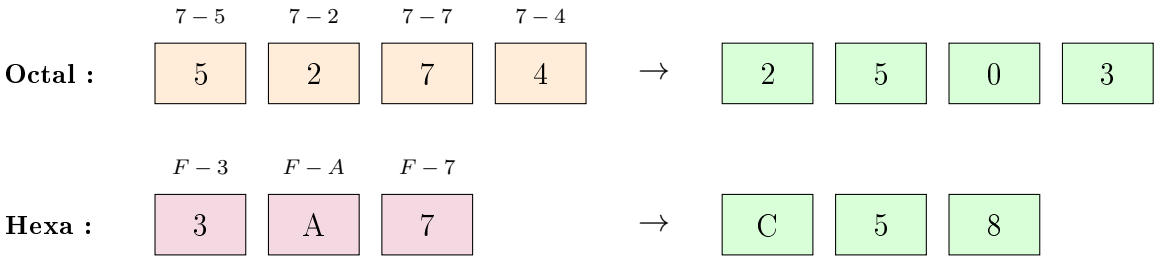

FIGURE 7 – Illustration du complément à $(r-1)$: chaque chiffre est soustrait de $(r-1)$

Exercice 8 : Configuration RAM et ROM de la carte mémoire

Analyse du bus d'adresses

Le bus d'adresses comporte **10 bits** (A10 à A1), ce qui permet d'adresser $2^{10} = 1024$ emplacements mémoire (adresses `0000` à `03FF`).

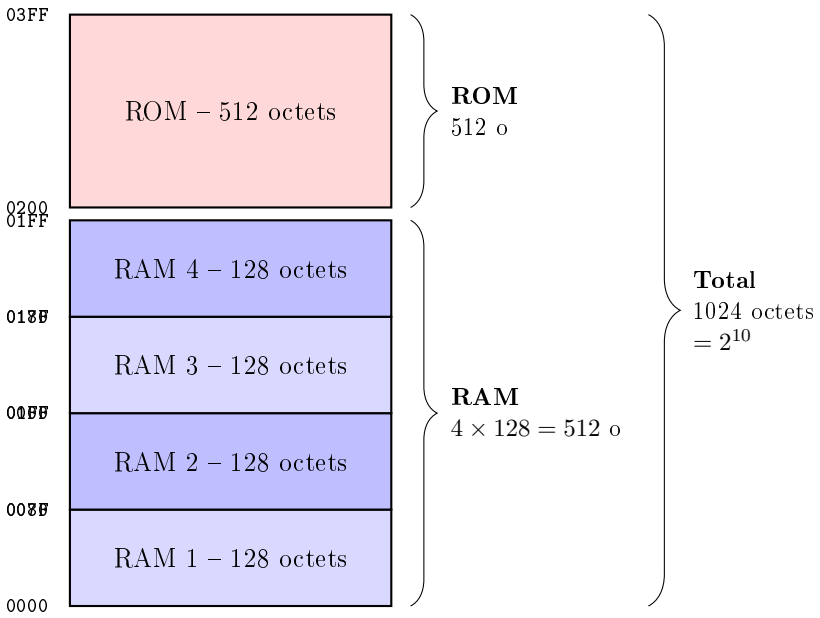

FIGURE 8 – Carte mémoire – répartition de l'espace d'adressage

Configuration de la RAM

La mémoire RAM est composée de **4 puces de 128 octets** chacune (2^7), pour un total de **512 octets** (adresses `0000` à `01FF`).

— **Adressage interne :** Chaque puce utilise les bits **A7 à A1** (7 bits) pour adresser ses 128 emplacements.

— **Sélection de puce :** Le bit **A10 = 0** identifie la zone RAM. Les bits **A9** et **A8** sélectionnent la puce active grâce à un décodeur 2 vers 4 :

A10	A9	A8	A7–A1	Composant sélectionné
0	0	0	x x x x x x x	RAM 1 (`0000`–`007F`)
0	0	1	x x x x x x x	RAM 2 (`0080`–`00FF`)
0	1	0	x x x x x x x	RAM 3 (`0100`–`017F`)
0	1	1	x x x x x x x	RAM 4 (`0180`–`01FF`)
1	x		x x x x x x x	ROM (`0200`–`03FF`)

Note : Les `x` représentent des bits « libres » utilisés pour l'adressage interne de chaque composant.

Configuration de la ROM

La mémoire ROM est composée d'**une seule puce de 512 octets** (2^9), occupant les adresses `0200` à `03FF`.

— **Adressage interne :** La puce utilise les bits **A9 à A1** (9 bits) pour adresser ses 512 emplacements.

— **Sélection de puce :** Le bit **A10 = 1** suffit à activer la ROM.

Logique de sélection des puces

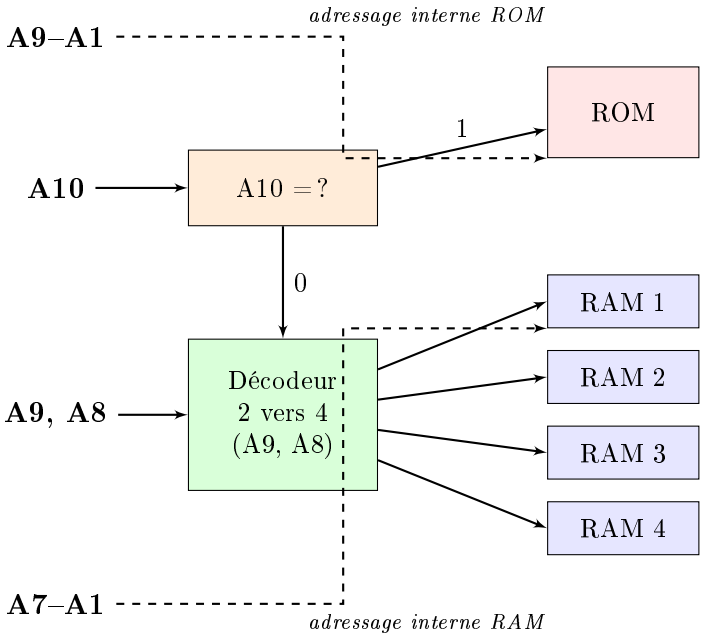

FIGURE 9 – Logique de sélection des puces mémoire

La mémoire totale adressable est donc : 512 (RAM) + 512 (ROM) = **1024 octets**, ce qui correspond bien aux 2^{10} combinaisons possibles du bus d'adresses de 10 bits.

Sources

1. M. Morris Mano, *Computer System Architecture*, 3e édition, Pearson, 1993.
2. M. Morris Mano, *Digital Design*, 5e édition, Pearson, 2012.
3. William Stallings, *Computer Organization and Architecture*, 10e édition, Pearson, 2016.
4. Andrew S. Tanenbaum, *Structured Computer Organization*, 6e édition, Pearson, 2012.
5. Notes de cours INF101 – Introduction aux outils informatiques, ISTEAH, Hiver 2026.
6. Images des exercices 1 et 2 générées par IA (Amazon Titan Image Generator v2).