# Algorithmes et Pensée Computationnelle

*Architecture des ordinateurs*

Le but de cette séance est de comprendre le fonctionnement d’un ordinateur. La série d’exercices sera axée autour de la conversion en base binaire, décimale ou hexadécimal, et du calcul de base en suivant le modèle Von Neumann.

# Conversions

**Question :** (*5 minutes*) **Conversion -**

1. Convertir le nombre 10 en base 2.
2. Convertir le nombre 45 en base 2.
3. Convertir le nombre 173 en base 2.

Vous pouvez utiliser la table des puissances de 2 pour vous aider.

| C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | **puissance** & & & & & &   
**valeur** & 32 & 16 & 8 & 4 & 2 & 1

Il existe 2 méthodes pour simplifier la conversion d’un nombre de la base 10 à la base 2. Ici le nombre 45 sera entièrement développé.  
**Méthode 1 :**  
Prenez votre nombre et divisez le par 2 en colonne. S’il y a un reste, notez le à coté, sinon notez 0. Répétez l’opération avec le résultat que vous venez d’obtenir, et ce jusqu’à arriver à 0. Pour lire votre nombre en binaire, prenez la suite de 0 et 1 correspondants aux différents restes, mais prenez les de bas en haut.

| C0.1 | C0.1 | **nombre** & **reste**  
& -  
& 1  
& 0  
& 1  
& 1  
& 0  
& 1

Ici, on obtient 101101  
**Méthode 2 :**  
Cette deuxième méthode consiste à utiliser la table des puissances de 2 que voici, ensuite on regarde la plus grande valeur de cette table qui peut être soustraite à notre nombre. Une fois trouvée, on la soustrait, on ajoute 1 sous la case correspondante et on répète l’opération avec le résultat obtenu. Répétez l’opération jusqu’à obtenir 0  
Ici, 45 - 32 = 13, puis 13- 8 = 5, puis 5-4 = 1, puis 1-1 = 0

| C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | **puissance** & & & & & &   
**valeur** & 32 & 16 & 8 & 4 & 2 & 1  
**binaire** & 1 & 0 & 1 & 1 & 0 & 1

On obtient donc 101101  
Les réponses sont les suivantes :

1. 10 = 1010
2. 45 = 101101
3. 173 = 10101101

**Question :** (*15 minutes*) **Conversion - , ,**

1. Convertir le nombre 40 en base 8.
2. Convertir le nombre 52 en base 3.
3. Convertir le nombre 254 en base 16.

S’inspirer de la solution de la question précédente.  
N’oubliez pas qu’en hexadécimal, A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14 et F vaut 15 !

Dans cette solution, la méthode 1 sera utilisée pour 52 en base 3, et la méthode 2 pour 254 en base 16.  
**Méthode 1 pour 52 (ici, nous sommes en base 3 donc on divise par 3) :**

| C0.1 | C0.1 | **nombre** & **reste**  
& -  
& 1  
& 2  
& 2  
& 1

Ici, on obtient 1221  
**Méthode 2 :**  
Ici, il faut prendre en compte le nombre de fois qu’on peut multiplier le nombre par la valeur de la puissance. Ici on a 256 qui est trop grand, il faut donc partir sur 16. On voit qu’on peut aller jusqu’à 15\*16 qui vaut 240. Donc on entre 15 sous la valeur 16 et on soustrait. 254-240 = 14. Il nous reste donc 14\*1, on met alors 14 sous la valeur 1.

| C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | **puissance** & & &   
**valeur** & 256 & 16 & 1  
**hexa** & 0 & 15 & 14

On obtient donc 15 14 (qui s’écrit FE)  
Les réponses sont les suivantes :

1. 40 = 50
2. 52 = 1221
3. 254 = FE

**Question :** (*15 minutes*) **Conversion - en**

1. Convertir le nombre 10110 en base 10.
2. Convertir le nombre 4321 en base 10.
3. Convertir le nombre ABC en base 10.

Ici utilisez les tables de puissances utilisées pour la méthode 2 (présentée dans la solution du premier exercice).  
N’oubliez pas qu’en hexadécimal, A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14 et F vaut 15 !

Reprenons la table des puissances utilisée précédemment (exemple avec ABC) :  
Ici, A vaut 10, B vaut 11 et C vaut 12

| C0.1 | C0.1 | C0.1 | C0.1 | C0.1 | **puissance** & & &   
**valeur** & 256 & 16 & 1  
**hexa** & 10 & 11 & 12

Ici on obtient donc 10\*256 + 11\*16 + 12\*1 = 2560 + 176 + 12 = 2748

1. Convertir le nombre 10110 = 22
2. Convertir le nombre 4321 = 586
3. Convertir le nombre ABC = 2748

# Arithmétique binaire

**Question :** (*15 minutes*) **Addition de nombres binaires**

1. Additionner 01010101 et 10101010
2. Additionner 01011111 et 10000001
3. Additionner 01110100 et 00011010

**Table d’addition binaire:**

| C0.1 | C0.1 | C0.1 | C0.1 | **a** & **b** & **s=a+b** & **r**  
& 0 & 0 & 0  
& 1 & 1 & 0  
& 0 & 1 & 0  
& 1 & 0 & 1

Il faut utiliser la table d’addition. Pour commencer, entrez a et b, puis commencez à additionner chaque ligne (si vous additionnez de haut en bas, n’oubliez pas d’entrer les nombres à additionner de bas en haut et de lire le résultat de bas en haut également, étant donné qu’on commence par additionner la fin des nombres, comme lors d’une addition en colonne). Si l’ addition vaut 1, le résultat est 1 et le reste est 0. Si l’addition vaut 2, le résultat est 0 et le reste est 1. Si l’addition vaut 3, le résultat est 1 et le reste est 1. N’oubliez pas d’ additionner également le reste à chaque ligne !  
Voici un exemple pour le deuxième point :

| C0.1 | C0.1 | C0.1 | C0.1 | **a** & **b** & **s=a+b** & **r**  
& 1 & 0 & 1  
& 0 & 0 & 1  
& 0 & 0 & 1  
& 0 & 0 & 1  
& 0 & 0 & 1  
& 0 & 1 & 0  
& 0 & 1 & 0  
& 1 & 1 & 0

On obtient donc 11100000

1. Additionner 01010101 + 10101010 = 11111111
2. Additionner 01011111 + 10000001 = 11100000
3. Additionner 01110100 + 00011010 = 10001110

**Question :** (*15 minutes*) **Soustraction de nombres binaires**  
Effectuer les opérations suivantes:

1. 01111111 - 01000000
2. 10000000 - 00000001
3. 10101010 - 01010101

**Table de soustraction binaire:**

| C0.1 | C0.1 | C0.1 | C0.1 | **a** & **b** & **s=a-b** & **r**  
& 0 & 0 & 0  
& 1 & 1 & 1  
& 0 & 0 & 0  
& 1 & 0 & 0

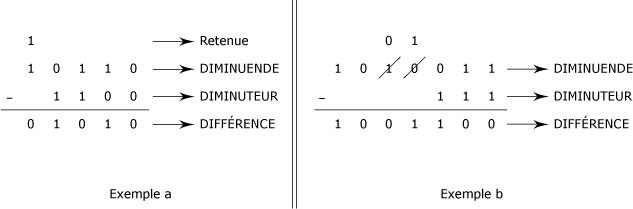
Il faut utiliser la table de soustraction. Pour commencer, entrez a et b, puis commencez à soustraire chaque ligne (si vous soustrayez de haut en bas, n’oubliez pas d’entrer les nombres à soustraire de bas en haut et de lire le résultat de bas en haut également, étant donné qu’on commence par soustraire la fin des nombres, comme lors d’une soustraction en colonne). Si la soustraction vaut 0, le résultat est 0 et le reste est 0. Si la soustraction vaut 1, le résultat est 1 et le reste est 0. Si la soustraction vaut -1, le résultat est 1 et le reste est 1. Si la soustraction vaut -2, le résultat est 0 et le reste est 1. N’oubliez pas de soustraire le reste à chaque fois !  
Voici un exemple pour le deuxième point :

| C0.1 | C0.1 | C0.1 | C0.1 | **a** & **b** & **s=a-b** & **r**  
& 1 & 1 & 1  
& 0 & 1 & 1  
& 0 & 1 & 1  
& 0 & 1 & 1  
& 0 & 1 & 1  
& 0 & 1 & 1  
& 0 & 1 & 1  
& 0 & 0 & 0

On obtient donc 1111111

1. 01111111 - 01000000 = 111111
2. 10000000 - 00000001 = 1111111
3. 10101010 - 01010101 = 1010101

**Exemple:**



Exemple de soustraction de nombres binaires

# Conversion et arithmétique

**Question :** (*20 minutes*) **Conversion et addition:**  
Effectuer les opérations suivantes:

1. 111101 + 110 = ...
2. 111111 + 000001 = ...
3. 127 + ABC = ...

Calculez à l’aide du tableau d’addition binaire ci-dessus ou une autre méthode que vous préférez.  
Additionnez les nombres lorsqu’ils sont dans la même base puis convertissez les dans la base souhaitée.  
Exemple de conversion : 1001000 = 72

Rappel: Les valeurs en hexadécimale (base 16)

Exemple de conversion : 3BF = 959

Exemple de conversion : 123 = ...  
   
 avec un reste de **3**  
 avec un reste de **7**  
 avec un reste de **1**

**123 = 173**

**6.1**  
Calcul du résultat en base 2:  
111101 + 110 = 1000011  
  
Conversion en base décimale:  
1000011 = 67  
  
Réponse :  
111101 + 110 = **67**

| C0.15 | C0.07 | C0.07 | C0.07 | C0.07 | C0.07 | C0.07 | C0.07 | C0.06 | Base & **1** & **0** & **0** & **0** & **0** & **1** & **1** &   
& & & & & & & &  
& & & & & & & &  
Équivalent & 1 x & 0 x & 0 x & 0 x & 0 x & 1 x & 1 x &  
Valeurs & 64 & 0 & 0 & 0 & 0 & 2 & 1 & = 67

**6.2**  
Calcul du résultat en base 2:  
111111 + 000001 = 1000000  
  
Conversion en base décimale:  
1000000 = 64  
  
Réponse :  
111111 + 000001 = **64**  
**6.3**  
ABC = 2748 (Conversion en base 10)

127 + 2748 = 2875 = 2875  
En commençant par zéro, augmentez 8 à des puissances entières de plus en plus grandes jusqu’à ce que le résultat dépasse 2875.

Déterminez les puissances de 8 qui seront utilisées pour placer les chiffres dans la représentation en base 8.  
   
Déterminez la valeur du premier chiffre en partant de la droite (correspondant à ) grâce au reste de la division entière.  
2875 / 8 = 359 avec un reste de **3**  
  
Divisez la partie numérique entière du quotient précédent, 359, par 8 et trouvez le reste. Le reste est le chiffre suivant (correspondant à ):  
359 / 8 = 44 avec un reste de **7**  
  
Ainsi de suite... la valeur pour :  
44 / 8 = 5 avec un reste de **4**  
  
La valeur pour :  
5 / 8 = 0 avec un reste de **5**  
2875 = **5473**

**Question :** (*20 minutes*) **Conversion et soustraction:**  
Effectuer les opérations suivantes:

1. 101010 - 010101 = ...
2. 64 - 001000 = ...
3. FFF - 127 = ...

Si besoin, référez-vous aux éléments des conseils précédents.

Calculez à l’aide du tableau de soustraction binaire ci-dessus ou une autre méthode que vous préférez.

Additionnez les nombres lorsqu’ils sont dans la même base puis convertissez les dans la base souhaitée.

Exemple de conversion : 1001000 = 72

Rappel: Les valeurs en hexadécimale (base 16)

| C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | C0.030 | **0** & **1** & **2** & **3** & **4** & **5** & **6** & **7** & **8** & **9** & **A** & **B** & **C** & **D** & **E** & **F**  
& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15

Exemple de conversion : 123 = ...  
   
 avec un reste de **1** (premier chiffre en partant de droite)  
 avec un reste de **1**  
 avec un reste de **0**  
 avec un reste de **1**  
 avec un reste de **1**  
 avec un reste de **1**  
 avec un reste de **1** (dernier chiffre en partant de droite)

123 = **1111011**

**7.1**  
Calcul du résultat en base 2:  
101010 - 010101 = 1000000  
  
Conversion en base décimale:  
1000000 = 64  
  
Réponse:  
101010 - 010101 = **64**

**7.2**  
Conversion de 001000 en base 10 :  
001000 = = 8  
Calcul du résultat en base 10:  
64 - 8 = 56  
  
Réponse :  
64 - 001000 = **56**

**7.3**  
Conversion de FFF en base 10:  
FFF = 4095  
  
Calcul en base décimale:  
4095 - 127 = 3968  
  
Convertir 3968 en base binaire :  
En commençant par zéro, augmentez 2 à des puissances entières de plus en plus grandes jusqu’à ce que le résultat dépasse 3968.  
Déterminez les puissances de 2 qui seront utilisées comme les places des chiffres dans la représentation en base 2 de 3968 :

Déterminez la valeur du premier chiffre en partant de la droite (correspondant à ) grâce au reste de la divsion entière.  
3968 / 2 = 1984 avec un reste de **0**  
Divisez la partie numérique entière du quotient précédent, 1984, par 2 et trouvez le reste. Le reste est le chiffre suivant (correspondant à ):  
1984 / 2 = 992 avec un reste de **0**  
Ainsi de suite... Pour   
992 / 2 = 496 avec un reste de **0**  
Pour   
496 / 2 = 258 avec un reste de **0**  
Pour   
258 / 2 = 124 avec un reste de **0**  
Pour   
124 / 2 = 62 avec un reste de **0**  
Pour   
62 / 2 = 31 avec un reste de **0**  
Pour   
31 / 2 = 15 avec un reste de **1**  
Pour   
15 / 2 = 7 avec un reste de **1**  
Pour   
7 / 2 = 3 avec un reste de **1**  
Pour   
3 / 2 = 1 avec un reste de **1**  
Pour   
1 / 2 = 0 avec un reste de **1**  
  
3968 = 111110000000  
  
Réponse:  
FFF - 127 = **111110000000**

# Modèle de Von Neumann

Dans cette section, nous allons simuler une opération d’addition dans le **modèle de Van Neumann**, il va vous être demandé à chaque étape (FDES) de donner la valeur des registres.  
**État d’origine:**  
A l’origine, notre Program Counter (PC) vaut 00100000.  
Dans la mémoire, les instructions sont les suivantes:

| C0.1 | C0.1 | **Adresse** & **Valeur**  
& 00100100  
& 10110110  
& 11101101

Les registres sont les suivantes:

| C0.1 | C0.1 | **Registre** & **Valeur**  
& 11100011  
& 01101100  
& 00100101  
& 00000000

Les opérations disponibles pour l’unité de contrôle sont les suivantes:

| C0.1 | C0.1 | **Numéro** & **Valeur**  
& MOV  
& XOR  
& ADD  
& SUB

**Question :** (*5 minutes*) **Fetch**  
À la fin de l’opération FETCH, quelles sont les valeurs du Program Counter et de l’Instruction Register?

Pour rappel, l’**unité de contrôle (Control Unit)** commande et contrôle le fonctionnement du système. Elle est chargée du **séquençage** des opérations. Après chaque opération FETCH, la valeur du **Program Counter** est incrémentée (valeur initiale + 1).

À la fin de l’opération Fetch, le Program Counter vaudra 00100001 tandis que l’Instruction Register vaudra 10110110, ce qui correspond à la valeur de l’adresse mémoire 00100000.

**Question :** (*5 minutes*) **Decode**

1. Quelle est la valeur de l’opération à exécuter?
2. Quelle est l’adresse du registre dans lequel le résultat doit être enregistré?
3. Quelle est la valeur du premier nombre de l’opération?
4. Quelle est la valeur du deuxième nombre de l’opération?

Pensez à décomposer la valeur de l’Instruction Register pour obtenir toutes les informations demandées.  
Les données issues de la décomposition de l’Instruction Register ne sont pas des valeurs brutes, mais des références. Trouvez les tables concordantes pour y récupérer les valeurs.

En décomposant l’Instruction Register (10110110), on obtient les données suivantes:

* 10, correspond à l’opération à effectuer,
* 11, correspond à **l’adresse** du registre où sera sauvegardé le résultat,
* 01, correspond à **l’adresse** du premier nombre,
* 10, correspond à **l’adresse** du deuxième nombre.

À partir de ces informations, on peut répondre aux questions posées:

1. Valeur de l’opération: ADD
2. Adresse du registre dans lequel le résultat doit être enregistré: 11
3. Premier nombre de l’opération: 01101100
4. Deuxième nombre de l’opération: 00100101

**Question :** (*5 minutes*) **Execute**  
Quel est résultat de l’opération?

Toutes les informations permettant d’effectuer l’opération se trouvent dans les données de l’Instruction Register.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| + | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|  | **1** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |